THE STABLE MARRIAGE PROBLEM

— Optimizing Different Criteria Using Genetic Algorithms

Master's (one year) thesis in Informatics (15 credits)

Ioannis Damianidis
s101248@student.hb.se

AUTUMN 2010:MI18

HÖGSKOLAN I BORÅS
INSTITUTIONEN FÖR DATA- OCH AFFÄRSVETENSKAP

Title: The Stable Marriage Problem – Optimizing Different Criteria Using Genetic Algorithms

Year: 2010

Author/s: Ioannis Damianidis

Supervisor: Ulf Johansson

Abstract

"*The Stable marriage problem (SMP) is basically the problem of finding a stable matching between two sets of persons, the men and the women, where each person in every group has a list containing every person that belongs to other group ordered by preference. The first ones to discover a stable solution for the problem were D. Gale and G.S. Shapley. Today the problem and most of its variations have been studied by many researchers, and for most of them polynomial time algorithms do not exist. Lately genetic algorithms have been used to solve such problems and have often produced better solutions than specialized polynomial algorithms. In this thesis we study and show that the Stable marriage problem has a number of important real-world applications. It the experimentation, we model the original problem and one of its variations and show the benefits of using genetic algorithms for solving the SMP.*"

Keywords: Stable Marriage problem, Genetic Algorithm, Maximum egalitarian happiness matching, maximizing criteria.

Table of Contents

1 Introduction

1.1 The standard SMP

The stable marriage problem is basically the problem of finding a stable matching between two sets of persons, the men and the women, where each person in every group has a list containing every person that belongs to other group ordered by preference. The problem of finding a stable matching was stated in a paper published in 1962 by Gale and Shapley (1962, p. 11), and it can be defined as follows:

"A certain community consists of n men and women. Each person ranks those of the opposite sex in accordance with his or her preferences for a marriage partner. We seek a satisfactory way of marrying off all members in the community. ... we call a set of marriages unstable (and here the suitability of the term is quite clear) if under it there are a man and a woman who are not married to each other but prefer each other to their actual mates."

The stable marriage problem (SMP) has been studied and researched by a large number of scholars throughout the previous decades. The first ones to discover a stable solution for the problem were D. Gale and G.S. Shapley (1962) by the introduction of their algorithm, called the GSS algorithm, in their paper "College Admissions and the Stability of Marriage". Since their findings, many other aspects and variations of the problem have been examined and solved. Also a multitude of modifications and implementations of their algorithm exist.

This is how the GSS algorithm (Gale and Shapley, 1962) functions: One of the sets (usually the men) is the applicants, that is the ones who propose marriage. We start with the first man who proposes to his most preferred woman. She has to accept since she is single and they become engaged. They are not matched until the end of the procedure. We could say that the woman keeps the man in mind in case no-one better comes up. This is one of the assumptions of the algorithm. Then the second man proposes to his most preferred woman. Now there are two possibilities: She is either engaged or single. If she is single they become engaged, but if on the other hand she is engaged, she checks her list and compares her fiancée with the proposer. If she prefers the new man more than her fiancée then she breaks her engagement and creates a new engagement with the new man. If not then the man proposes to his next most preferred woman and the cycle repeats. The algorithm assures that every girl will get a proposal in the end and it proves that there is always a stable set of marriages. It ends when every woman has gotten at least one proposal and since we have n men that propose to n women, the algorithm's complexity is $O(n^2)$ (Dubins and Freedman, 1981). In figure 1, below you can see the algorithm where m = men and w = women.

```
1.    assign each person to be free;
2.    while some man m is free and m has a nonempty list loop
3.          w := first woman on m's list; m proposes to w
4.          if m is not on w's preference list then
5.                delete w from m's preference list;
6.                goto line 3
7.          end if
8.          if some man p is engaged to w then
9.                assign p to be free;
10.         end if
11.         assign m and w to be engaged to each other;
12.         for each each successor p of m on w's list loop
13.               delete p from w's list;
14.               delete w from p's list;
15.         end loop;
16.   end loop;
```

Figure 1: The Gale Shapley algorithm

The purpose of this thesis is two-fold. In the first part, we will analyse the SMP and its real-world applications, and the variations of the problem. Later, we will present our own implementation through the use of the genetic algorithm(GA), and evaluate our results according to our objectives.

1.2 Problem Statement

The GSS algorithm always finds a stable matching. In a stable matching, there is no man-woman pair that would prefer each other compared to their current matches. However there are some problems with the original algorithm. First of all, it produces either the man optimal or the woman optimal result, depending on which group is proposing. "*A stable marriage is called optimal if every applicant is at least as well off under it as under any other stable assignment.* " (Gale and Shapley, 1962, p. 10).

The problem with the man optimal result is that it is also the result where every woman gets her worst choice partner, meaning the marriages are stable but all the women are highly unsatisfied with their partners, and vice versa for the woman-optimal matching.

Since 1962, a great amount of articles and books have been written about the stable marriage problem. Nowadays, there are variations of the SMP which have been proven to be NP-hard (Iwama, Manlove, Miyazaki and Morita, 1999; Manlove, Irving, Iwama and Miyazaki, 2002). NP-hard stands for 'non-deterministic polynomial-time hard' which means, that the SMP belongs to a category of problems for which it has not been proven yet if polynomial time algorithms exist for solving them. Therefore, approximation algorithms were suggested to deal with the NP-hard variations of the problem. An approximation algorithm is an algorithm that because no polynomial time solution can be found, it settles for an optimal solution, of a specific optimality, of a subset of the problem.

In contrast with approximation algorithms, the genetic algorithm is a heuristic algorithm which finds a solution of good quality but of unknown optimality. Genetic algorithms are a category of evolutionary algorithms that function in a procedure similar to the one used in natural evolution to find solutions to optimization problems. They usually start from a population of random solutions and then rate the individuals using a fitness function. The best individuals are combined then together to create better solutions until an acceptable result is reached. They also allow us to specify the properties of the solution we are looking for, by modifying the fitness function. This is why we decided to use a genetic algorithm for our experiments because in that way we can specify if we want solutions that are just stable or also have a better happiness for the women than the GSS result, or have other different properties according to the criteria we wish to apply.

GAs are especially equipped for dealing with NP-hard problems (Grefenstette, Gopal, Rosmaita and Van Gucht, 1985). The reason because genetic algorithms are good at this is because they can go through a large number of solutions quite fast and locate good solutions. One example of a problem that falls in the same category and has been successfully solved with the genetic algorithm, is the travelling salesman problem where we seek to find the shortest way of travelling between a number of cities.

An interesting variation of the SMP is the Stable Marriage with Correlated lists where lists are affected by beauty, which means that certain individuals that are considered prettier are higher in everyone's list.

1.3 Objectives

The objectives behind this thesis are basically two:

1. First of all, our goal is to find and describe the most important instances of the SMP and its variants in real-life.

2. Our next objective is by studying the original algorithm by Gale and Shapley. to create a solution based on a genetic algorithm that focuses on finding solutions that may not be stable but have better happiness values and are more fair than the GSS solution, concerning the difference between happiness between men and women. Also, we will evaluate how correlated lists affect the solutions.

1.4 Main contribution

Our contribution is the evaluation of how correlation affects the results in the GA results. It is clear that as the level of correlation increases, the properties of the solutions also change.

1.5 Overall method

In order to achieve our goals we followed the following strategy. Concerning the first part of the research, we did a thorough examination and literature study of previous work on the

subject, and identified SMP applications and modifications that exist.

For the second part, we decided to use the genetic algorithm to create an implementation of the problem in MATLAB. After that we created a suitable fitness function and in addition an implementation of the GSS algorithm. Then a number of experiments was conducted for the original SMP and for the Stable Marriage with Correlated lists. Our plan was to compare the two implementations and judge them in accordance to certain criteria, in order to show how efficient the GA is for finding solutions to the Stable marriage problem, and how these solutions change as correlation increases.

2 Background Theory

In chapter 2 we provide all the instances and variations of the SMP we managed to locate through literature study. After that we describe some basic elements of the genetic algorithm and finally we present other attempts to combine the two subjects.

2.1 Stable Marriage problem variations

There are a few variations of the SMP that can be combined together of course to create even more. Here we present those that are the most important and have been studied more over the last 50 years.

2.1.1 Stable marriage with Ties

Perhaps the simplest generalization of the SMP is the SMT, the Stable marriage problem with Ties, or with indifference (Iwama, Manlove, Miyazaki and Morita, 1999; Irving, Manlove and Scott, 2000). The difference about it is that the preference lists include ties, in the sense that a person finds two of the persons of the opposite sex equally preferable and they both occupy the same position in his/her preference list. The notion of stability then changes. In order for us to move on, we need to present the notions of weak strong and super stability as mentioned by Manlove (1999, p. 3) :

"*A matching M is weakly stable if there is no couple (x; y), each of whom strictly prefers the other to his/her partner in M. Also, a matching M is strongly stable if there is no couple (x; y) such that x strictly prefers y to his/her partner in M, and y either strictly prefers x to his/her partner in M or is indifferent between them. Finally, a matching M is super-stable if there is no couple (x; y), each of whom either strictly prefers the other to his/her partner in M or is indifferent between them.*"

Out of the three stability notions weak-stability is the most important (Manlove, et al., 2002) and it is shown that a weak stable matching always exists. In order to tackle the SMT the solution is quite simple.

"By breaking the ties arbitrarily, an instance I of SMT becomes an instance I' of SM, and it is clear that a stable matching for I' is a weakly stable matching for I. Thus a weakly stable matching for I may be found in O(n^2) time, using the Gale/Shapley algorithm" (Manlove, 1999, p. 2).

The way however, that ties are broken has an impact on the size of the stable matchings that the algorithm produces, and it is impossible to identify the most suitable way (Irving and Manlove, 2007). Because every instance of SMT actually includes many different instances depending on how we choose to treat it, we have many different sizes of stable matchings according to the way we break ties (Irving, Manlove and Scott, 2008). Nevertheless, Irving (1994) produced an algorithm that finds solutions that include no strong or super-stable matching.

The SMT is usually studied in combination with the next variation, that of Incomplete Lists, so a few more algorithmic solutions about it are mentioned in chapter 2.1.2.

2.1.2 Stable marriage with Incomplete Lists

Known as SMI (Stable Marriage with Incomplete lists) it is a variation of the original problem that is more realistic. In this case, we have the situation where a woman might declare that one or more men are unacceptable for her, meaning she would under no circumstances accept a proposal from them even if she were single. A stable matching exists in this variation but it does not always contain all persons, some might remain "single" (Gale and Sotomayor, 1985).

You can observe below the differences between the preference lists for the women, for the SMP with a population of 4 men and women, for the four variations of the problem. SMTI stands for Stable marriage with Ties and Incomplete lists which is the combination of the two problems.

W_x is the preference list of woman number x (likewise for men) and the brackets in figure 2 symbolize ties in the lists. The SMI and SMTI lists contain only the acceptable individuals.

Standard problem	**SMT**
w_1: m_1 m_3 m_4 m_2	w_1: m_1 [m_3 m_4] m_2
w_2: m_4 w_1 m_3 m_2	w_2: m_4 m_1 [m_3 m_2]
w_3: m_1 m_2 m_3 m_4	w_3: m_1 m_2 m_3 m_4
w_4: m_2 m_3 m_4 m_1	w_4: [m_2 m_3 m_4] m_1

SMI	**SMTI**
w_1: m_1	w_1: m_1 [m_3 m_4]
w_2: m_4 m_3 m_2	w_2: [m_3 m_2]
w_3: m_1 m_2	w_3: m_1
w_4: m_2 m_3 m_4 m_1	w_4: [m_2 m_3 m_4] m_1

Figure 2: Preference lists for SMP

By combining SMT and SMI we get the SMTI(Stable marriage problem with Ties and Incomplete Lists), which was first addressed by Ronn (1986). Manlove (1999) also provided three algorithms, one for each notion of stability, that were based on Irving's work (1994). Irving's (1994) contribution was algorithms for the stable marriage problem with complete lists and ties. The application of both incomplete lists and ties causes the problem to become NP-hard (Iwama, Manlove, Miyazaki and Morita, 1999). For the SMTI a number of approximation algorithms exist (Iwama, Miyazaki and Yamauchi, 2007; Irving and Manlove, 2007). In the SMTI, the stable matchings that exist, are not always of the same size. So the algorithms must also focus on finding a maximum stable matching, also known as MAX SMTI (Iwama, Manlove, Miyazaki and Morita, 1999). In comparison with the SMP, the SMTI also has at least one stable matching where the algorithm's complexity is O(a), By a we symbolize the acceptable pairs that exist (Gusfield and Irving, 1989). Alternative solutions include the use of local search solutions, that try to improve a matching by moving from a solution with n stable pairs to one with n+1 until an optimal one is reached (Marx and Schlotter, 2010; Gelain, et al., 2010).

2.1.3 Agents

Agents refer to the existence of individuals, whose purpose is to match the men with the women and each of them has in their possession the preference lists of one or more participants. The pairs are created after negotiations between the respective agents. The preference lists of each person therefore are hidden. Each agent only has possession of the information of his own client, and knows nothing about the way other participants have ranked him/her. It can be said that if the participants chose to act as agents to themselves this variation could be treated as the original problem. The main focus in this situation for every participant is to act by keeping his preferences private. This problem is known as the Distributed Stable Marriage Problem (DisSM) (Brito and Meseguer, 2008).

2.1.4 The sex-equal stable marriage

It is clearly evident that the SMP could never be implemented to solve the problem of divorces in modern society. The reason for that is that each man or woman would prefer the matching in which their happiness is maximized. The algorithms which have as a goal to achieve stable solutions are diametrically opposed to the individual happiness of any participant (Caldarelli and Capocci, 2000).

We adopt the idea that a person's happiness or his regret cost, relates to the rank of his/her partner has on his/her preference list (Dzierzawa and Omero, 2000). In the following simple problem for 3 men and women, the regret cost for man 1 if he picks woman 3 is 3 since that is her rank, and woman 3 has a regret cost of 2:

Men's preference lists	Women's preference lists
1: 1 2 3	1: 3 1 2
2: 3 2 1	2: 3 2 1
3: 2 3 1	3: 3 1 2

Figure 3: Regret costs

Therefore, if our aim is to maximize the total happiness for both the men and women, we could achieve it through the minimization of a function. We bring into attention that in the Gale Shapley algorithm even though the happiness of the men is maximum, there might be another stable matching with a better total happiness.

The sex-equal stable marriage problem (Gusfield and Irving, 1989), or the sex fair problem, can be defined like this: If M is a stable matching between n men and women and the position of a woman w in a man's preference list is $p_m(w)$, and for the woman $p_w(m)$ respectively, we can define the happiness cost h(M), and the egalitarian happiness cost eh(M), for any instance of SMP, with formulas (1) and (2) (Iwama and Miyazaki, 2008).

Happiness cost

$$h(M) = \sum p_m(w) + \sum p_w(m) \quad (1)$$

Egalitarian cost

$$eh(M) = \sum p_m(w) - \sum p_w(m) \quad (2)$$

We can define the happiness per person if we divide the above numbers with the number of participants. The formulas 3,4 and 5 give us the happiness per person(hpp), for men or women, the happiness per couple(hpc) and the egalitarian happiness per couple(ehc) (Caldarelli and Capocci, 2000). Where N is the number of men/women.

Happiness per person

$$hpp(M) = 1/N(\sum p_m(w)) \quad (3)$$

Happiness per couple

$$hpc(M) = 1/N(\sum p_m(w) + \sum p_w(m)) \quad (4)$$

Egalitarian happiness per couple

$$ehc(M) = 1/N(\sum p_m(w) - \sum p_w(m)) \quad (5)$$

Back in 1987, Gusfield (1987) presented three polynomial time algorithms using graph theory to solve the SMP, and proved how to construct a tree graph, representing the SMP. One of those algorithms found the minimum regret stable matching for women. The problem of finding a sex-equal stable matching is NP-hard (Yanagisawa, 1993), and polynomial algorithms where suggested by Gusfield and Irving (1989). Approximation algorithms also exist for the sex-equal problem (Iwama, 2007).

Apart from those forms of happiness and their respective maximum matchings, lately there has been research around finding a lexicographic maximum matching. A lexicographic maximum stable matching is defined as follows (Irving, Manlove and Scott, 2008, p. 2):

"A lexicographic maximum stable matching is one in which the maximum number of people obtain their 1rst-choice partner, and subject to this condition, the maximum number obtain their second-choice partner, and so on."

2.1.5 Stable matchings

We already mentioned that at least one stable matching exists in every SMP instance (Gale and Shapley, 1962). Knuth (1976) wondered if it is possible to know the number of stable matchings that exist if we know the number of men and women. The answer was given by Irving and Leather (1986), by an algorithm that calculated, that if n is a power of 2, we can calculate the number of stable matchings that exist. In SMTI, or in the hospital/residents problem with couples, that is described later, a stable matching does not always exist. However, other solutions that are not stable but have a maximum number of stable pairs can be found in those situations. This constitutes, a good judging criteria, concerning the quality of the solutions. Also, if total stability is not a fundamental requirement then solutions with the maximum number of couples, are frequently more desirable, especially for a large number of participants. In these cases the matches with the largest possible number of stable matchings are given priority (Biro, Manlove and Mittal, 2010).

2.1.6 Quantitative lists

In contrast to the SMP where every man only assigns a rank to every woman, SMP with Quantitative lists, or SMQ, includes values that show how much a man prefers a woman compared to another one. SMQ can be handled in the same way and can be solved with traditional SMP algorithms (Gusfield, 1987; Irving and Gusfield, 1989). Lately a new side of the problem was studied where the notion of *a* -stability was introduced, where *a* symbolizes the preference value (Pini, Roshi, Venable and Walsh, 2010).

2.1.7 Beauty and distance

Caldarelli and Capocci (2000) created a more realistic model by introducing the notions of beauty and distance. It is obvious, that in the real world, people's opinions about beauty tend to be similar, so those tendencies should be reflected in the preference lists of each gender. It is highly unlikely for example, in an instance of 1000 men and women that someone would rank a woman 1st and someone else would rank the same woman last. This instance of the

problem also known as SM with Correlated lists, is of great significance because it is a closer representation of reality than the standard SMP where the preference lists are created randomly. Also in the real world, people usually find partners that live geographically close to them. By applying those two principles in their model Caldarelli and Capocci (2000) came to the conclusion that:

"*It is interesting to note, however, that even if the more beautiful players have by far a larger satisfaction in their matching with respect to the others, the general dissatisfaction in the system increases. As a matter of fact, when the concept of "most beautiful" in the world tends to be the same for everyone it becomes more and more difficult to make more people happy. However, the presence of beauty transforms in a fairer way the GS algorithm that now tends to give the same results regardless the sex.*" (Caldarelli and Capocci, 2000, p. 4).

The above results can be easily explained. In the case of uncorrelated lists, it is easier to satisfy the majority of the participants because their interests, in most cases, do not coincide. So in extreme cases like in the man-optimal matching it could happen that every man is matched with his first choice, if all the first choices are different. When beauty is applied however, it is certain that the opinions of people and their preference lists will start to look more and more alike. Therefore, it is evident that more and more people would have to settle for a lower ranking partner. Thus the overall satisfaction decreases compared to original SMP. Nevertheless, the happiness for the women increases since now the more beautiful women would get more proposals and it is more likely that they would get more preferable partners.

In order to introduce beauty into the SMP we can use formula number (6), that Caldarelli and Capocci (2000) used:

$$S = n + U*I \qquad (6)$$

The preference lists in this situation are created as follows. Each man gives a score S to each woman. S is consisted of n and UI. In this case, n is a random number between 0 and 1. This reflects the man's individual opinion of the woman. The second number I belongs again to (0,1), but is the same for every man. U is a value used to weigh the contribution of I in every preference list. In that way it is certain that every man will have a different value for the same woman w, but we ensure that the scores tend towards a specific value according to the value of I. For larger values of U, beauty plays greater part, and is considered more important, and the larger U becomes the more the lists become identical for everyone. It is also noted that there is a gap between the happiness of the more preferable and "uglier" people, that grows as U increases. If U = 0 then beauty plays no part whatsoever and we have an instance of the original SMP. Then all the women are sorted according to their scores in a descending order and the preference lists are created. The women's preference lists are created in the same way.

In contrast, the introduction of distance, calculated in the scores in a similar way as beauty, did not yield different results in comparison with the classic algorithm, because the criterion of beauty had a much larger impact on the creation of the lists. Distance did not give an

advantage to every woman or man that was considered beautiful, but rather every participant received a set of “neighbours” that had an advantage in comparison with the rest people for that specific preference list. So the lists' nature became more random in contrast with beauty. In a subsequent paper (Caldarelli, Capocci and Laureti, 2001) it was considered that the participants had incomplete information about the other sex. It was thought that distance separated people in such a way, that some of the participants had access or were acquainted to only a fraction of the population of the other sex. So the preference lists contained subsets of the population different for each man/woman, and the problem now started to look like SMI. The results from such studies implied that, the competition for more beautiful partners was not so fierce as before, due to lack of knowledge and lack of alternative partners to choose from. So as a result, the more the number of people that were missing from the preference lists increased, the better total happiness tended to be.

Beauty and distance are terms borrowed from the real world. They do not apply only to the marriage example. We could say that beauty is the attribute that the other side finds attractive in every instance. For example, in the student-hospitals representation which is examined in chapet 2.2.1, beauty has different meanings for each side. For the students, beauty could be the sum of all their qualifications, since that is what hospitals would find attractive in a medical student, and for the hospitals beauty could represent how well-known or prestigious a hospital is. Distance could also have some significance, since some students might prefer hospitals situated close to their homes or in specific cities where they would like to reside in.

2.2 Real world applications

There are some instances of the stable marriage problem like the firm-workers problem (Roth and Sotomayor,1992), that are exact copies of the SMP with only the terminology changing. The firm-workers problem is a basic representation of the job market, where every company wants to hire the best workers and every worker wants to be hired at the company he prefers most, and is of course an instance of SMP. The original problem is however, significant since all other methodologies for solving the more complex variations are based on the GSS algorithm.

We will not mention any more real-world applications that are similar to the firm-workers example, but we will present those instances that include different aspects, in comparison with the original problem and how researchers have addressed them in order to deal with their characteristic properties.

2.2.1 College admissions and the hospital residents problem

Perhaps the most practical and the most well-known application of SMP is college admissions. The same problem is also known as the hospitals/medical students matching problem (Roth and Sotomayor, 1992). It was mentioned for the first time in the paper by Gale and Shapley (1962). Here we have the notion that each college can admit a number of students, so more than one student can be admitted in the same college. The problem that led to the use of SMP in this situation, was that in the mid-40's medical students, when they received offers from hospitals, they used to wait in case a better offer would present itself

from a hospital more to their liking. The situation would end up in unhappy students who accepted their first offers only to regret it or in unhappy hospitals when students did not keep their earlier commitments.

The problem was resolved through an initiative called the National Internships Matching Problem (Roth and Sotomayor, 1990) which in 1951 solved the problem long before Gale and Shapley came up with stable matchings. Same initiatives are still used around the world for the particular problem such as USA, Canada and Scotland (Irving, Manlove and Scott, 2000). These organisations have a policy of producing the hospital/college optimal matching. Obviously, the algorithm solves the problem but there is one alteration required. Since hospitals/colleges take in a number of students we need to have as many clones of every hospital as their quota (Dubins and Freedman, 1981). All the clones have the same preference rank as the original hospital. Then we have a case of matching many to one.

In the same paper (Dubins and Freedman, 1981) it was proven that the original algorithm always resulted in the student-optimal result and it was also shown that by deliberating handing in false preferences a student, cannot improve his position, that is to receive a better matching, provided that all the other students are truthful about their own. If a group of students applies the same strategy, some might get better matchings but not all of them. Therefore, it is more profitable, for the proposers to always hand in their real preferences.

However, the above statement is true for the proposing side only. It was proven by Dubins and Freedman (1981) and Gale and Sotomayor (1985), that if a woman states a falsified preference list, then under some situations she could yield better results. It was proven that:

"*If there is more than one stable matching, then there is at least one woman who will be better off by falsifying, assuming the others tell the truth.*" (Gale and Sotomayor, 1985, p. 5)

In the category of cheating we could also include a situation studied by Irving (2008), where after the allocation of hospitals, two students realised that they would prefer to have one another's hospital, and exchanged their positions. This did not happen because the matching was not stable but because the students noticed that they were better off trading places, although the hospitals were quite unhappy about it. Then those two students formed a man-blocking pair.

If we generalise the original problem and consider the case where the number of students is more than the available college positions, then there is a number of students that will remain "single". Those students will be the same in every stable matching (Gale and Sotomayor, 1985).

In the case where the hospital-residents problem might be an instance of SMT there are some facts that need to be taken into account. This situation is very plausible since hospitals have a large number of applicants on a national level, and there might often be applicants with the same qualifications, or from the applicants point of view indifference between choices is certain to come up. Irving, Manlove and Scott (2000), the existence of an algorithm that

produces a weak stable matching which has the largest number of pairs as possible, and they also noticed the problem where a student might convince a hospital to prefer him in the expense of another student, when the hospital is indifferent between the two of them.

This system has been in use since the 1950's in America. However in the 1970's, it was noted that there was a number of students that negotiated their positions outside the system. This was due to the fact that many medical students used to get married during their college years so they naturally preferred to be residents at the same hospital (Dean, Goemans and Immorlica, 2006). So because the system could not facilitate their needs, it was to their benefit not to participate in the NRMP(National Residents Matching Program). In order to deal with such cases the NRMP changed its algorithm and strategy many times (Roth and Sotomayor, 1990; Roth and Peranson, 1999) and allowed couples to have a common preference list, however in such cases the problem is NP-hard and stable matchings might not exist (Ronn, 1990). In addition, because in different instances of the hospital/residents problem with couples, depending on which stable matching we choose, the number of students that are admitted changes, so selecting one matching over the other could affect the future of some students (Aldershof and Carducci, 1996).

2.2.2 The sailors-boats problem

In a similar way to the hospitals/medical students problem, we have the problem of assigning sailors to boats in the U.S, Navy. Sailors are given new assignments every few years and they are required to hand in a preference list. The Navy is responsible for arranging these assignments, and it must do so in a way that the cost of making those assignments is minimized or kept within bounds of the budget. The cost of re-educating and training the sailors in ways suitable for their new duties is also a cost parameter. But it is also imperative that the Navy takes into consideration not only the stability of the matching, but also the happiness of the sailors and the commanders of the boats, because in the boat optimal case, the happiness of the sailors is minimal, which would certainly result in a drop in moral (Garrett et al., 2005), or vice versa in the sailor optimal case. The whole process has been computerized with the sailors selecting their preferred positions through an online pool.

The whole process can be reduced to an instance of the SMTI, but it does not take into consideration the implementation costs, so that a stable matching might not be affordable, or a sailor might not be qualified for a job. Therefore, we have another case of a NP-hard problem.

2.2.3 The stable room-mates problem

Mentioned in the original paper (Gale and Shapley, 1962) as well as by Irving (1985), the room-mates problem describes a matching problem where a set of people have to pick someone to be their room-mate, and therefore the sex of the people involved is of no concern so we can have only one set of people with preference lists. The significant difference mentioned by Gale and Shapley (1962) is that for this problem even when we have complete lists, there are cases where a stable matching is not possible. Below, in figure 4 we have the

example they gave:

Person	Preference List
1	2 3 4
2	3 1 4
3	2 1 4
4	arbitrary

Figure 4: Instance with no stable solutions

It is evidently shown in figure 4, that anyone would prefer someone else other than n. 4 so this instance will always be unstable.

If we introduce incomplete lists and ties in the problem, we get the SRT and the SRTI. Ronn (1986; 1990) studied the problem and showed that the existence of ties in preference lists makes the SRT NP-complete. Irving and Manlove (2002) studied the problem and proposed an approximation algorithm.

2.2.4 Application in router technology

MUFCA (Most Urgent Cell First) is an algorithm presented by Balaji Prabhakar and Nick McKeown (1999) for solving the problem of creating a switch used in LAN switches and routers that while it falls on the category of switches that combine input and output queuing they act however in the same way as an output-queued switch. In order for it to handle inputs and outputs, it uses the Gale Shapley algorithm (1962). Each input in MUFCA has a urgency value and according to that value, the preference lists for each switch are created, and then inputs are matched with the outputs. This resulted in a speed-up of the process by four times.

2.2.5 Stable allocation problem

The stable allocation problem is basically a matching of many to many. It has many every day applications (Dean, Goemans and Immorlica, 2006). The matching of network clients to specific servers is one of them. In order to improve performance it is better to assign servers to clients that are closer geographically. Also the assignment of teaching assistants in universities to courses is another example. One difference with the standard SMP is that costs or weights that must be satisfied are assigned to each agent in the problem. The Gale-Shapley algorithm can be extended and it has been proven that it solves this problem in the same way as the SMP and it provides the man-optimal solution (Dean, Goemans and Immorlica, 2006).

2.3 The genetic algorithm

In chapter 2.3 we describe the basic genetic algorithm and its main components; i.e., selection, reproduction and fitness.

2.3.1 Definition and history

Genetics algorithms are a category of evolutionary algorithms. A definition of genetic algorithms was given by Goldberg (1989, p. 1):

"*Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance.*"

Research on genetic algorithms started in the 1950's, when following the work of Alex Fraser (1957), Hans-Joachim Bremermann introduced the notions of mutation and selection and grounded the basics of genetic algorithms (1962). Interest in this new technique rose in the 1970's and 1980's mainly due to the book written by John Holland (1975) *Adaptation in Natural and Artificial Systems.* Soon applications of GA in artificial intelligence (Schwefel, 1977, 1981) arose, and finally research led to the production of genetic algorithm products. Genetic algorithms are especially efficient at finding solutions to optimization problems and search problems. An optimization problem consists of a pool of solutions out of which we need to find the best possible one. As mentioned in chapter 1, some variations of SMP are NP-hard optimization problems.

2.3.2 The genetic algorithm

Following the patterns of living organisms and natural evolution, genetic algorithms solve mathematical problems. In the beginning, the problem that needed solution, had to be represented as a bit string of 0's and 1's. Nowadays, and in our problem, other representations are available. Each string represents an individual or a chromosome, and a population is a collection of those chromosomes. In the travelling salesman problem, we face the problem of finding the shortest route for a salesman to travel between a number of cities. With 4 cities we could represent the problem by an ordered string such as this one [2,3,4,1] where each element of the string represents a city and the order of the elements defines the order with which they are going to be visited.

The first step of the GA is to create an initial population. There are no restrictions concerning the initialization, although one could try to begin with a set of solutions that one knows are better, or it can be done randomly. Once a number of those solutions is created, it is easy to discern that parts of some chromosomes, or maybe whole ones, will be better than the others. This evaluation takes place through the use of a fitness function.

A fitness function is an objective function that is used to measure the quality of a chromosome. Then through the use of a selection function, those chromosomes that are of

better quality or fittest, are selected. In nature that would be equivalent to “survival of the fittest”. It is imperative that the fitness function is related to the goal we wish to achieve. In the travelling salesman problem again, a possible fitness function would be a function that calculates the total distance travelled and selects the chromosomes with the minimum value.

More than one fitness functions could be used for the same problem, depending on the value we wish to minimize. The complexity of the GA is $O(n^3)$ (Goldberg, 1989), where n equals the number elements in a string. The fitness function is the key component of the GA since it must capture the correct optimization criterion and, at the same time, it must be fairly straightforward to calculate the fitness values. Because of that it requires a large amount of computation time for complicated problems, and the GA has been criticised hardly because of that.

By isolating only the best chromosomes a reproduction strategy is needed after that. There are many to choose from but the ones mostly used are crossover and mutation.

Crossover is a reproduction technique that requires two chromosomes, the “parents”. By combining the two parents in a specific way, a number of new chromosomes is acquired, called “children”. That can be achieved in a variety of ways. The most common is one or two point crossover. A position is selected in both chromosomes and then they are split them into two pieces and the pieces in the two parents are swapped.

We then evaluate the children through the fitness function and select the fittest and reject the weakest, so that we have a new population. In most of the cases, the new generation will be more fit than the previous one since it will be descended from mostly the fittest chromosomes. For our problem these two techniques are unsuitable, so we applied a crossover function called cyclic crossover, which is described in chapter 3.

Mutation is another term borrowed from natural evolution. Mutation only requires one and not two parents. As the word implies, by applying mutation to a chromosome we randomly change an element of the string sequence. In problems such as ours and the travelling salesman, where we have restrictions over the appearance of each number, this technique can not be applied but there are other ones such as inversion of string, or the swapping of two elements. The reason behind using mutation along with crossover is because by selecting the fittest children every time we have the risk of our results becoming localized, and the population chromosomes' becoming identical to one another. Therefore, by adding mutation, we avoid that risk. If we wish to further reduce that risk, we can add a probability value to each member of the population. Therefore, even the weak chromosomes might have an opportunity to be selected, adding in that way to the diversity of the new population. Unfit chromosomes will remain in the population for a few generations, thanks to mutation, in a similar way to natural selection. Nevertheless, if the rate of mutation is too high we run the risk of losing some fit solutions.

The procedure of evaluating each chromosome and then selecting the most optimal ones as a reproduction pool, is repeated until the algorithm is terminated. There are a few termination

reasons, and the most common are, reaching a previously set number of generations, reaching a satisfactory solution, achieving a fitness value or running out of computation time. There are no guarantees that the GA will always find the best solution, because the algorithm is heuristic. Because the algorithm tends to go for fit solutions in the short-term future, it might be led towards a solution that seems at the time to be fitter and lose fittest solutions. So, another algorithm, for example a linear algorithm might sometimes provide a better solution than the GA. It is still an open question which category of algorithms performs better.

2.3.3 Selection

A number of methods exist for selecting the most appropriate parents for reproduction. The majority of them evaluate the quality of the chromosomes using the values of the fitness function. A description of the most common ones follows.

- **Stochastic uniform** places all individuals in a line where each individual occupies a part of the line depdning on how fit it is. The algorithm goes through the line in a number of steps equal to the number of parents. The individual on which the algorithm lands after each step is then selected as a parent.

- **Roulette** is similar to a wheel where the area of each individual is proportional to its fitness value. The better the fitness value, the bigger the area is. The algorithm then randomly selects one of the sections with a probability equal to the area it occupies.

- **Tournament** selects each parent by choosing two individuals from the population at random and then it selects the best one by comparing their fitness values.

Other settings of the GA that were taken into account where the elite count, which is the number of chromosomes from the previous generation that survive for the next generation while the rest of them are thrown away. Of significance also are the crossover and mutation rate, that calculate what percentage of the children are created by crossover and what percentage by mutation.

2.4 Related work

We have already mentioned that genetic algorithms excel at handling NP-hard optimization problems. In Roth's and Vande Vate's work (1990) a sequence of steps was described for finding a stable matching that is similar to the process the GA uses. They used the following strategy, themselves following the example of Knuth (1976).

Beginning from a random matching, Roth and Vande Vate (1990) located the first blocking pair for the matching. A blocking pair is a pair of people that prefer each other to their current partners, and because of them the matching is unstable (Roth and Vande Vate, 1990). A way to measure how unstable a matching is, is by counting the blocking pairs. By altering the matching in a way that the two people are matched together one can come to a new matching, which might be stable or might contain more blocking pairs. By continuing in the same way,

a stable matching will eventually be found. The GA basically follows the same pattern, but it changes more than one pair in every generation.

Aldershof and Carducci (1999), modelled a coding of the SMP and the hospital couples problem, through the use of a GA they created, and their goal was to solve the hospital/residents couples problem. They achieved that goal by representing the problem in the form of a bit matrix X of dimensions h and p, where h is the number of hospital positions, and p the number of students. If a student and a hospital are matched then the value of of X(h,p) = 1, otherwise it is 0. That matrix was then translated into a string where every position represented a hospital, and the number located in that position represented the student that was assigned to that hospital. They represented the rest of the problem in the form of inequalities. If all those inequalities are satisfied by a chromosome X then that chromosome constitutes a stable marriage. Moreover, in their initialization and reproduction functions they ensured that their population produces acceptable pairs (Aldershof and Carducci, 1999), that is pairs that have each other on their preference lists.

As far as the couples problem is concerned, they separated the applicants set into three categories. Those who are single applicants, and two sets of men and women for the couples that create a married couple, which then is matched to a hospital. More inequalities (Aldershof and Carducci, 1999) were then introduced for this problem.

For fitness function, they simply created a function that adds the number of inequalities that were satisfied. This helped them locate solutions for the couples problem where a stable matching might not exist, but it is the best available solution might be located anyway, even though it might lack stability. For a mating function they used cyclic crossover, which is described in chapter 3.1.3.

As far as mutation was concerned, they chose to use a function that randomly or after finding an unstable pair in a chromosome, performs the change. Mutation is of importance, because simply changing a random number in the string may result in an illegal chromosome.

Their results were finding all matches in the SMP, and a student-optimal matching but no hospital-optimal matching in the couples problem. Also, it appeared that singles get more satisfactory results than couples, and couples had a larger probability of being unmatched in the end of the algorithm, which can simply be explained by the fact that, it is easier to satisfy the pre-requisites of a single than of two persons. They also emphasized on the need for an algorithm with specific criteria for deciding among matchings.

The sex-fair problem was studied through the use of a GA (Nakamura, Onaga, Kyan and Silva, 2002). The SMP was then translated into a graph problem as shown in the picture below and the effectiveness of the GA was confirmed. The representation of the problem was the same as before (Aldershof and Carducci, 1999).

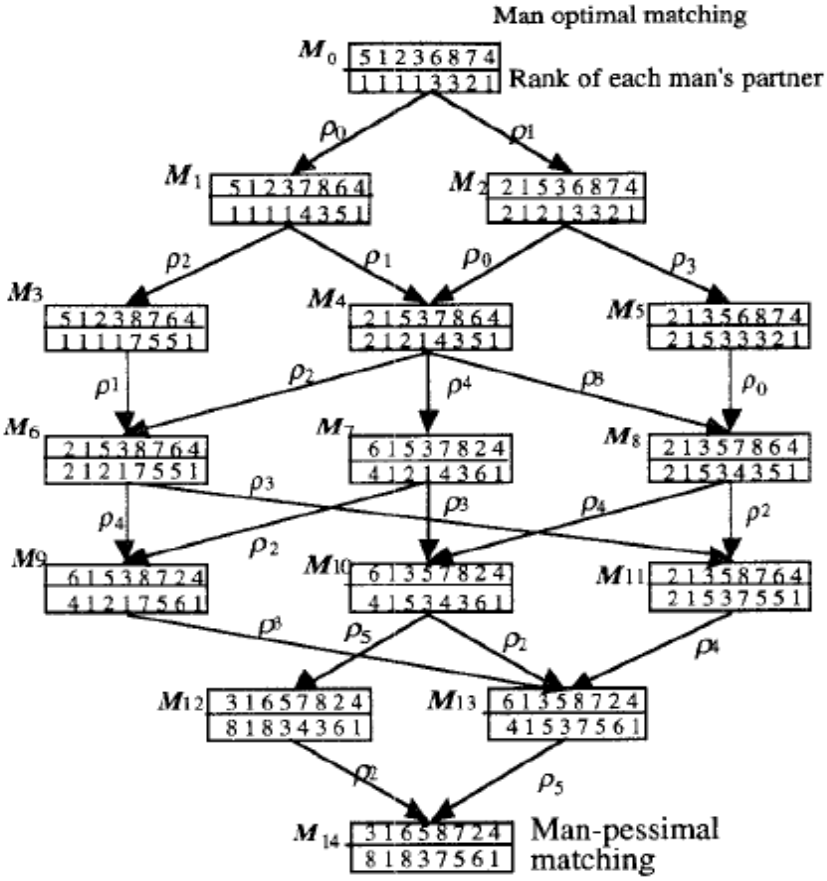

Figure 5: Graph representation of SMP

The sailor-boat problem has also been transformed into a GA representation (Garrett, et al., 2005). The goal was to minimize the cost that the U.S. Navy must pay for making the assignments, and also maximize the number of sailors that are assigned jobs. However, a sailor can only apply for a specific set of jobs that he is qualified for. So creating an initial population as well as applying mutation functions, is harder in this instance of the problem. In order to bypass this problem, Garret et al. (2005) connected each sailor with the set of jobs he is allowed to be assigned to, beforehand. That way it was assure that no illegitimate matching would occur. They also applied a function for looking into the probability that a position might be assigned twice. Uniform crossover (Sywerda, 1989) was used as a reproduction function. Their research resulted in an algorithm that greatly improved the selection process.

In the chapter 3 we present the initialization function, the reproduction function and the fitness function, we have implemented with the help of MATLAB's genetic algorithm optimization tool, as well as the standard Gale-Shapley algorithm (1962).

We tested our implemented algorithms on the following two problems: the original stable

marriage problem with random lists and the stable marriage problem with correlated lists where 'beauty' affects the preferences of every participant. The reason why we chose the correlated lists problem was because it gives a closer representation of a real life problem and because as mentioned before this is the main contribution of this thesis. Preferences are never totally random but there always more popular individuals.

3 Method

3.1 Representation

In chapter 3.1 we are going to present all the details about the functions we created for the GA implementation.

3.1.1 Population

To begin with the standard bit-string representation of the first GAs is not particularly easy to use in the SMP. Our choice of chromosome type was a vector of integers, who represent the men matched to the women..For example, the vector [3, 1, 2] can be translated as the 1st woman is matched to the 3rd man, the 2nd woman to the 1st man and the 3rd woman to the 2nd man.

3.1.2 Initialization

There are two parameters that the GA must have before we start our experiments. The population vectors and the preference lists. The preference lists are created once and stay the same through the iterations. We chose to create both these elements randomly. One might argue that this would require a number of more iterations in the beginning until we receive good results, unless we are lucky, but we can sidestep that with a good population size to ensure we get more good results.

3.1.3 Crossover

The two crossover functions that we described in chapter 2.3.2 are not suitable for the SMP. If we chose to implement any of the two we would come to disastrous results. Let us take for example, the two chromosomes of length 8 [3, 8, 1, 2, 5, 7, 4, 6] and [2, 7, 8, 4, 6, 3, 5, 1]. If we used one-point crossover after position 3 we would come with the children: [2, 7, 8, 2, 5, 7, 4, 6] [3, 8, 1, 4, 6, 3, 5, 1], which contain the same number twice and thus destroy the chromosomes rendering them into not valid representations of the SMP. In addition, if our aim is to represent an instance with incomplete lists, we must make certain that our crossover function not only creates appropriate descendants, but also that the matchings are acceptable.

The solution to the above problem is to use cyclic crossover (Aldershof and Carducci, 1999). Cyclic crossover operates as follows: Two children are created, so that in each of them an integer is placed only in the position it occupied in the parents. By accepting that rule, acceptability of the matchings is assured. We randomly select, a position in the first parent.

We copy the integer x_1 in that position in the same position in the child. Now, if we look at the same position in the second parent, because of the way the function works, we must place the integer x_2 that lies there, who is called the opposing *gene*, in the position it has in the first parent (since x_2 has taken the place it had in parent 2). So we copy x_2 in the position it has in the 1^{st} parent. Then we get a new opposing gene which we also copy in the position it holds in the 1^{st} parent. We continue in such fashion, until we find a number that already exists in the child. The genes that have not been copied so far, we copy them into the 2^{nd} child (which has been empty until now). We now have two half-empty chromosomes. We complete them with the missing genes in the same position they have in the 2^{nd} parent. The whole process is better described step-by-step, in figure 6.

Parent 1: [3, 2, 5, 1, 4]
Parent 2: [1, 3, 4, 2, 5]
starting point is position **2**

Step	Parents	Children
1	**Parent 1**: [3, **2**, 5, 1, 4] **Parent 2**: [1, **3**, 4, 2, 5]	**Child 1** : [, 2, , ,] **Child 2** : []
2	**Parent 1**: [**3**, 2, 5, 1, 4] **Parent 2:** [**1**, 3, 4, 2, 5]	**Child 1** : [3, 2, , ,] **Child 2** : []
3	**Parent 1**: [3, 2, 5, **1**, 4] **Parent 2**: [1, 3, 4, **2**, 5]	**Child 1 :** [3, 2, , 1,] **Child 2** : []
4	**Parent 1**: [3, 2, **5**, 1, **4**] **Parent 2**: [1, 3, 4, 2, 5]	**Child 1 :** [3, 2, , 1,] **Child 2** : [, , 5, , 4]

Final children

Child 1 : [3, 2, 4, 1, 5]
Child 2 : [1, 3, 5, 2, 4]

Figure 6: Cyclic crossover example

Starting from position 2 in Parent 1, we place “2” into Child 1 in the second position (Step 1). The corresponding gene is number “3” in Parent 2. It occupies position 1 in Parent 1 and we copy it into Child 1 (Step 2). The next gene is “1”. After copying it into Child 1 (Step 3), we have the next gene is “2” which already exists in Child 1, so we are obligated to stop. We copy the remaining genes from Parent 1 into Child 2 (Step 4). Then we fill in the gaps in both children with the genes in the positions they hold in Parent 2 (final children).

One disadvantage of this method is that, although it guarantees the reproduction of acceptable children, is that the children due to the strict positioning rule, clones of the same chromosome are created. Especially if the generating function creates chromosomes randomly. The use of a slightly increased rate of mutation helps us avoid that danger and ensure diversity.

3.1.4 Mutation

Applying mutation has the same difficulties as crossover. Because of its random nature it has the capability to disturb the stability and structure of the chromosome and make it unacceptable. For our problem we chose to use two functions where one of them is chosen randomly each time mutation is used. In the first function we randomly swap the positions of 5% of the elements of the vector, and and not just randomly change one gene, thus keeping the chromosome acceptable. In the second function we locate all the unstable pairs in an individual and then select one of them randomly and swap it with another random element.

3.1.5 Fitness function

The values of the fitness function depend on the kind of problem we wish to maximize. In chapter 2.1.4 we mentioned how happiness is defined in the SMP.

The fitness function we considered at first, consisted of the sum of the number of stable pairs plus the value of happiness. In the problem of maximizing the man or woman happiness we used a function where each person's partner is given a value of their rank in the preference list. That way first choices get the lowest values and last choices the highest. This is both convenient and inconvenient, because the GA tries to solve any given problem by minimizing the fitness function values. So for example, in the original problem the matchings with the highest amount of stable matchings is the individual with the highest fitness value and therefore it will be rejected.

In contrast, when we are interested about happiness scores, the individual with the lowest happiness score is considered the most fit, and that matches perfectly our chosen representation of happiness, which is according to formula (4). Individuals with their first options like in optimal cases will get the best scores.

The same applies for egalitarian function costs. To solve the stable pairs representation problem, we inverted the value (matching with 9 stable matchings would get the score 2 and vice versa). That way the stable pairs score must be minimized, exactly like the others. We also added three constants S, H and E to weight the contribution of every optimality criterion to the fitness value. In conclusion, our final fitness formula for one individual x was formula (7):

$$F(x) = S*sta\_pairs(x) + H*hap(x) + E*ehap(x) \qquad (7)$$

This function value contains stable pairs, happiness for every person, and egalitarian happiness. The values A, B, C represent how much every criterion affects the final fitness

score. The more we increase the value of one of these three the more significant is any increase or decrease of that criterion , meaning the fitness function focuses more on minimizing that specific one. By setting S=H=E=1 we acknowledge that all three criteria play the same role and we search for a solution which compromises for a stable matching that has good happiness and egalitarian happiness values. By setting H=E=0 we can research only for stable matchings. By setting E=0 we would obtain better happiness solution. Finally, for H=0 the algorithm returns a better egalitarian solution.

3.1.6 Implemented algorithms for comparison

We have implemented the Gale-Shapley algorithm (1962), as a means of comparison with our own algorithm. It always yields the man-optimal solution, so we know beforehand that unless the man-optimal solution is the same with the happiness solution it always gives us results that are not happiness optimal. So we are quite sure that our GA implementation will outperform the GSS.

3.2 Experiments

3.2.1 Algorithm settings

First, we took the Gale-Shapley algorithm and noted the produced results, and then ran the GA algorithm experimenting with different sets of values for every setting. The quality of the solutions was then compared and studied. Because of the nature of the GA, it is not guaranteed that it always succeeds in finding all the possible stable matchings, but in most cases it always finds at least one solution of good quality.

The matter of settings was not of great importance in our study. Our primary goal was to show that the GA can deal with the SMP and its variants with ease and produce satisfying results. We experimented with 3 different n sizes of 10, 25, 50 in both variations. For the same n size we also experimented with different weight values for stability in contrast with maximum happiness, to observe the differences in the solutions. We applied each combination of S,H,E on 5 problems and then found the average value.

After that we wished to have a correlated lists example where with higher correlation so that we could see the differences between the solutions depending on the correlation value.

The GA settings for every experiment had to be modified because for larger values of n we need a larger population size and a larger iteration size so that we do not miss any solutions. All the settings for our experiments are mentioned here on table 1:

n	10	25	50
Population size	100	1000	4000
Number of Iterations/Running time	150	1000/13 minutes	20000/57 minutes
Selection function	Stohastic Uniform		
Crossover/Mutation fraction	0.1/0.9		

Table 1: Genetic algorithm settings

The last setting states that 10% of the children come from crossover and 90% from mutation.

3.2.2 Original Stable marriage

We used a problem of small size to test our algorithm with an instance of the original problem at first. We present here and comment the results for the problem of size n = 10 to see if our algorithm manages to solve the typical problem and find all the stable matchings.

By cross-checking our results with the results of the GSS algorithm, Our goal was not to just reach a stable matching, but by altering our fitness function and running more experiments for different S, H and E values, to find solutions with better happiness and egalitarian happiness values. That way we searched for results that would be sex-fair but also of high happiness for both sexes, by sacrificing stability.

The reasons for choosing these values was to find out and compare the solutions for different combinations of the criteria. We wished to use a function that maximizes more than one criterion at the same time. We then examined how the absence of one or more of the criteria affects the solutions. Our purpose was to compare the solutions where one of the S,H, or E are equal to zero with the first setting where they are all present.

In tables 2,3 and 4, you can see the GSS results. Hap(m) represents the happiness of men, hap(w) the happiness of women, hap(m+w) the overall happiness and ehap the egalitarian happiness.

Stable pairs	Hap(m)	Hap(w)	Hap(m +w)	Ehap
10	3.2	2.7	5.9	0.5
10	1.7	4.3	6	2.6
10	2.9	3.2	6.1	0.3
10	1.8	4.1	5.9	2.3
10	2	3.8	5.8	1.8
Average values				
10	**2.1**	**3.85**	**5.95**	**1.75**

Table 2: GSS results for n =10 for SMP

Stable pairs	Hap(m)	Hap(w)	Hap(m +w)	Ehap
25	3.8	6.56	10.36	2.76
25	2.88	7.8	10.68	4.92
25	2.24	10.36	12.6	8.12
25	4.2	5.48	9.68	1.28
25	2.44	8.08	10.52	5.64
Average values				
25	**2.94**	**7.93**	**10.87**	**4.99**

Table 3: GSS results for n =25 for SMP

Stable pairs	Hap(m)	Hap(w)	Hap(m +w)	Ehap
50	3.12	13.1	16.22	9.98
50	4.16	12.74	16.9	8.58
50	3.56	16.24	19.8	12.68
50	3.94	12.5	16.44	8.56
50	4.76	11.12	15.88	6.36
Average values				
50	**3.61**	**14.03**	**17.64**	**10.41**

Table 4: GSS results for n =50 for SMP

In tables 5,6, and 7 we have the GA results with their average values for every S,H,E combination that we experimented with for all the n sizes.

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
10	3.2	2.7	5.9	0.5
10	2.6	2.1	4.7	0.5
10	2.9	3.2	6.1	0.3
9	3.1	3.3	6.4	0.2
10	3	2.5	5.5	0.5
Average values				
9.8	**2.96**	**2.76**	**5.72**	**0.4**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
10	3.2	2.7	5.9	0.5
10	2.6	2.1	4.7	0.5
10	3.2	3	6.2	0.2
9	3.3	3.3	6.6	0
10	3	2.5	5.5	0.5
Average values				
9.8	**3.06**	**2.72**	**5.78**	**0.34**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
10	3.2	2.7	5.9	0.5
10	2.6	2.1	4.7	0.5
10	2.9	3.2	6.1	0.3
10	2.8	3.4	6.2	0.6
10	3.8	1.5	5.3	2.3
Average values				
10	**3.06**	**2.58**	**5.64**	**0.84**
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
9	2.8	2.7	5.5	0.1
8	2	2.3	4.3	0.3
7	2.9	3	5.9	0.1
6	3	3.1	6.1	0.1
9	2.6	2.3	4.9	0.3
Average values				
7.8	**2.66**	**2.68**	**5.34**	**0.18**

Table 5: Genetic Algorithm results for n =10 for original SMP

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
24	5.2	4.56	9.76	0.64
24	4.88	4.24	9.12	0.64
23	5.32	5.24	10.56	0.08
23	4.72	4.48	9.2	0.24
23	4.6	5.16	9.76	0.56
Average values				
23.4	**4.94**	**4.74**	**9.68**	**0.43**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
22	5.68	5.72	11.4	0.04
25	4.6	4.32	8.92	0.28
23	5.52	5.24	10.76	0.28
25	4.84	4.44	9.28	0.4
25	4.68	4.76	9.44	0.08
Average values				
24	**5.06**	**4.9**	**9.96**	**0.22**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
25	6.68	3.28	9.96	3.4
23	4.68	4.72	9.4	0.04
24	7.2	3.68	10.88	3.52
22	4.92	3.48	8.4	1.44
25	4.68	4.76	9.44	0.08
Average values				
23.8	**5.63**	**3.98**	**9.62**	**1.7**
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
19	4.76	4.76	9.52	0
17	4.76	4.8	9.56	0.04
17	5.16	5.2	10.36	0.04
20	4.32	4.08	8.4	0.24
19	4.64	4.68	9.32	0.04
Average values				
18.4	**4.73**	**4.7**	**9.43**	**0.07**

Table 6: Genetic Algorithm results for n =25 for original SMP

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
41	8.32	8.36	16.68	0.04
38	8.44	8.56	17	0.12
41	9.08	8.88	17.96	0.2
43	7.52	7.24	14.76	0.28
41	9.1	8.98	18.08	0.12
Average values				
40.8	**8.49**	**8.4**	**16.9**	**0.09**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
41	11.22	11.2	22.42	0.02
38	12.38	12.44	24.82	0.06
41	11.38	11.46	22.84	0.08
39	11.36	11.32	22.68	0.04
38	12.26	12.26	24.52	0
Average values				
39.4	**11.72**	**11.74**	**23.46**	**0.02**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
43	9.96	5.48	15.44	4.48
43	7.38	6.32	13.7	1.06
43	11.4	5.8	17.2	5.6
42	9.94	6.36	16.3	3.58
43	8.64	6.36	15	2.28
Average values				
42.8	**9.46**	**6.06**	**15.53**	**3.4**
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
31	6.96	7.06	14.02	0.1
34	7.36	7.44	14.8	0.08
27	7.04	7.08	14.12	0.04
33	6.88	6.88	13.76	0
30	7.6	7.5	15.1	0.1
Average values				
31	**7.17**	**7.19**	**14.36**	**0.02**

Table 7: Genetic Algorithm results for n =50 for original SMP

3.2.3 Stable Marriage with Correlated Lists

The major difference in the SMP with Correlated lists is in the way the preference lists are created. We used formula (6) to produce the preference lists. You can see the GSS results on tables 8,9, and 10. The value of correlation for this problem was U = 1.

Stable pairs	Hap(m)	Hap(w)	Hap(m +w)	Ehap
10	3.2	2.7	5.9	0.5
10	1.7	4.3	6	2.6
10	2.9	3.2	6.1	0.3
10	1.8	4.1	5.9	2.3
10	2	3.8	5.8	1.8
Average values				
10	2.1	3.85	5.95	1.75

Table 8: GSS results for n =10 for SM with Corr. lists

Stable pairs	Hap(m)	Hap(w)	Hap(m +w)	Ehap
25	3.8	6.56	10.36	2.76
25	2.88	7.8	10.68	4.92
25	2.24	10.36	12.6	8.12
25	4.2	5.48	9.68	1.28
25	2.44	8.08	10.52	5.64
Average values				
25	**2.94**	**7.93**	**10.87**	**4.99**

Table 9: GSS results for n =25 for SM with Corr. lists

Stable pairs	Hap(m)	Hap(w)	Hap(m +w)	Ehap
50	50	14.12	16.92	31.04
50	50	14.84	12.94	27.78
50	50	14.66	14.44	29.1
50	50	13.86	13.58	27.44
50	50	13.76	14.42	28.18
Average values				
50	**14.54**	**14.77**	**29.31**	**0.23**

Table 10: GSS results for n =50 for SM with Corr. lists

Here are our experimental results for the new problem on tables 11,12 and 13.

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
9	3.8	4.1	7.9	0.3
9	3.4	4.2	7.6	0.8
10	3.7	4	7.7	0.3
10	4.3	4.1	8.4	0.2
10	4.3	3.6	7.9	0.7
Average values				
9.6	**3.9**	**4**	**7.9**	**0.46**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
9	4.6	4.4	9	0.2
9	3.9	4.5	8.4	0.6
10	3.7	4	7.7	0.3
10	4.3	4.1	8.4	0.2
10	4.3	3.6	7.9	0.7
Average values				
9.6	**4.16**	**4.12**	**8.28**	**0.4**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
9	3.8	4.1	7.9	0.3
10	2.8	4.7	7.5	1.9
10	3.7	4	7.7	0.3
10	3.5	4.6	8.1	1.1
10	4.3	3.6	7.9	0.7
Average values				
9.8	**3.62**	**4.2**	**7.82**	**0.86**
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
6	3.8	3.8	7.2	0
5	3.7	3.7	7.4	0
7	3.5	3.5	7	0
7	3.8	3.7	7.5	0.1
6	3.4	3.4	6.8	0
Average values				

6.2	**3.64**	**3.62**	**7.18**	**0.02**

Table 11: Genetic Algorithm results for n =10 for SM with Corr. lists

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
21	7.28	7.28	14.56	0
25	8	7.8	15.8	0.2
25	8.4	8.24	16.64	0.16
21	8	7.92	15.92	0.08
24	8.08	7.76	15.84	0.32
Average values				
23.2	**7.95**	**7.8**	**15.75**	**0.15**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
21	9.16	9.16	18.32	0
23	8.44	8.56	17	0.12
20	9	9	18	0
23	8.68	8.68	17.36	0
22	8.28	8.28	16.56	0
Average values				
21.8	**8.71**	**8.74**	**17.45**	**0.02**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
21	8.36	6.76	15.12	1.6
23	9.52	6.36	15.88	3.16
24	9.36	7.04	16.4	2.32
23	8.04	7.36	15.4	0.68
24	8.6	6.92	15.52	1.68
Average values				
23	**8.78**	**6.89**	**15.66**	**1.89**
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
12	6.88	6.88	13.76	0
10	7.2	7.16	14.36	0.04
8	7.4	7.4	14.8	0
11	7.2	7.2	14.4	0

11	7.12	7.08	14.2	0.04
Average values				
10.4	**7.16**	**7.14**	**14.3**	**0.02**

Table 12: Genetic Algorithm results for n =25 for SM with Corr. lists

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
38	14.02	14.1	28.12	0.08
41	14.34	13.88	28.22	0.46
36	15.04	14.96	30	0.08
37	14.62	14.58	29.2	0.04
40	14.64	14.68	29.32	0.04
Average values				
38.4	**14.53**	**14.44**	**28.97**	**0.09**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
37	18.14	18.14	36.28	0
31	19.12	19.12	38.24	0
34	17.46	17.46	34.92	0
34	18.58	18.58	37.16	0
30	19.86	19.86	39.72	0
Average values				
33.2	**18.63**	**18.63**	**37.26**	**0**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
38	15.26	12.96	28.22	2.3
37	15.36	11.36	26.72	4
36	15.06	13.08	28.14	1.98
38	14.26	12.88	27.14	1.38
37	12.94	14.6	27.54	1.66
Average values				
37.2	**14.58**	**12.98**	**27.55**	**1.6**
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
13	12.7	12.7	25.4	0
20	12.6	12.6	25.2	0

18	12.62	12.58	25.2	0.04
23	12.98	13	25.98	0.02
15	12.2	12.18	24.38	0.02
Average values				
17.8	**12.62**	**12.61**	**25.23**	**0.01**

Table 13: Genetic Algorithm results for n =50 for SM with Corr. lists

In order to compare how the results change with the increase of correlation we have also run experiments with the same settings for n = 25 and U = 1.5. Here are the results on table 14.

S = 1 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
23	9.28	9.16	18.44	0.12
23	9.52	9.2	18.72	0.32
23	9.4	8.24	17.64	1.16
22	9	8.72	17.72	0.28
22	9.36	9.4	18.76	0.04
Average values				
22.6	**9.31**	**8.94**	**18.26**	**0.38**
S = 1 H = 0 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
24	9.32	9.4	18.72	0.08
24	9.96	9.52	19.48	0.44
22	9.96	9.92	19.88	0.04
21	10.56	10.56	21.12	0
22	9.6	9.6	19.2	0
Average values				
22.6	**9.88**	**9.8**	**19.68**	**0.11**
S = 1 H = 1 E = 0				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
23	9	8.32	17.32	0.68
23	10.72	8.48	19.2	2.24
22	9.88	7.52	17.4	2.36
22	9.4	8.28	17.68	1.12
23	8.52	9.52	18.04	1
Average values				

22.6	9.5	8.42	17.93	1.48
S = 0 H = 1 E = 1				
Stable pairs	**Hap(m)**	**Hap(w)**	**Hap(m +w)**	**Ehap**
10	8.04	8	16.04	0.04
10	8	8.04	16.04	0.04
14	8.36	8.36	16.72	0
12	8.6	8.6	17.2	0
10	8.4	8.4	16.8	0
Average values				
11.2	**8.28**	**8.28**	**16.56**	**0.02**

Table 14: Genetic Algorithm results for correlated lists for n =25 with higher correlation

4 Analysis

4.1 Comparison of GA results with GSS results

To begin with we shall first compare the GSS with the GA results. On tables 15, 16 and 17, we have the average GSS results and the average GA results for comparison for the original problem.

Algorithm	Stable pairs	Hap(m + w)	Ehap
GSS	10	5.95	1.75
S =1 H = 1 E = 1	9.8	5.72	0.4
S =1 H = 0 E = 1	9.8	5.78	0.34
S =1 H = 1 E = 0	10	5.64	0.84
S =0 H = 1 E = 1	7.8	5.34	0.18

Table 15: Average GSS and GA results n =10 for original SMP

Algorithm	Stable pairs	Hap(m + w)	Ehap
GSS	25	10.87	4.99
S =1 H = 1 E = 1	23.4	9.68	0.43
S =1 H = 0 E = 1	24	9.96	0.22
S =1 H = 1 E = 0	23.8	9.62	1.7
S =0 H = 1 E = 1	18.4	9.43	0.07

Table 16: Average GSS and GA results n =25 for original SMP

Algorithm	Stable pairs	Hap(m + w)	Ehap
GSS	50	17.64	10.41
S =1 H = 1 E = 1	40.8	16.9	0.09
S =1 H = 0 E = 1	39.4	23.46	0.02
S =1 H = 1 E = 0	42.8	15.53	3.4
S =0 H = 1 E = 1	31	14.36	0.02

Table 17: Average GSS and GA results n =50 for original SMP

The purpose with our results was to minimize the values of happiness and egalitarian happiness. Minimized values mean better results.

By taking a look at the results, if we look at the values for all 3 criteria we can see that the GA results for happiness and egalitarian happiness are better compared to the GSS results. For egalitarian happiness especially the results are very close to optimality which is 0. There is however a small drop in the number of stable pairs. This goes in accordance with the theory since every person tries to get as best a matching as possible and that is at the expense of stability (Caldarelli and Capocci, 2000). This can also be seen at the last set of results for all n where S=0, H=1, E=1. Actually, for this setting the results are much better as far as happiness and egalitarian happiness are concerned. This is also the reason we used this combination of settings even though it doesn't take into account the number of stable pairs in the fitness function. On all other combinations of weight values, we can observe that we get a high improvement of happiness and egalitarian happiness at the cost of a few stable pairs. This proves that the GA can get better results with different properties if we use a fitness function containing more than one criteria.

Moreover, if we compare the solutions to each other, we can see that when either H or E are absent from the fitness function, we then get a greater improvement in the value of the remaining criterion. This is quite evident from the results. That way we have either a generally good solution for all criteria (S=H=E=1), or have a better happiness (E=0) or a better egalitarian happiness solution (H=0).

It can be noticed in the results for n =50, that the number of stable pairs drops more than in the previous experiments. That is due to the fact that the GA can not find all stable pairs in a reasonable running time for higher problem sizes. However, in this research stability was not considered more important than the other two criteria. It was considered more important to get improved values for the other two criteria than finding a stable solution. Stability restrictions can be relaxed in the real world also to satisfy other needs, like in the navy case where the U.S. Navy can block a matching if they think it is unprofitable economically even though the result would be stable. On tables 18, 19 and 20, we present the results for the correlated lists problem.

Algorithm	Stable pairs	Hap(m + w)	Ehap
GSS	10	7.8	1
S =1 H = 1 E = 1	9.6	7.9	0.7
S =1 H = 0 E = 1	9.6	8.28	0.4
S =1 H = 1 E = 0	9.8	7.82	0.86
S =0 H = 1 E = 1	6.2	7.18	0.02

Table 18: Average GSS and GA results n =10 for SM with Corr. lists

Algorithm	Stable pairs	Hap(m + w)	Ehap
GSS	25	16.26	1.3
S =1 H = 1 E = 1	23.2	15.75	0.15
S =1 H = 0 E = 1	21.8	17.45	0.02
S =1 H = 1 E = 0	23	15.66	1.89
S =0 H = 1 E = 1	10.4	14.3	0.02

Table 19: Average GSS and GA results n =25 for SM with Corr. lists

Algorithm	Stable pairs	Hap(m + w)	Ehap
GSS	50	29.31	0.23
S =1 H = 1 E = 1	38.4	28.97	0.09
S =1 H = 0 E = 1	33.2	37.26	0
S =1 H = 1 E = 0	37.2	27.55	1.6
S =0 H = 1 E = 1	17.8	25.23	0.01

Table 20: Average GSS and GA results n =50 for SM with Corr. lists

As it is seen, all the above comments apply here as for the original problem. The GA managed to find better solutions concerning happiness and egalitarian happiness by sacrificing some stable pairs.

4.2 Comparison of GA results between the two variations

In the second part of our results' analysis we will compare the solutions between original SMP and correlated lists SMP for both low and high correlation for n = 25. Here they are on table 21.

Criteria	Stable pairs	Hap(m + w)	Ehap
Original SMP			
S =1 H = 1 E = 1	23.4	9.68	0.43
S =1 H = 0 E = 1	24	9.96	0.22
S =1 H = 1 E = 0	23.8	9.62	1.7
S =0 H = 1 E = 1	18.4	9.43	0.07
Correlated SMP U = 1			
S =1 H = 1 E = 1	23.2	15.75	0.15
S =1 H = 0 E = 1	21.8	17.45	0.02
S =1 H = 1 E = 0	23	15.66	1.89
S =0 H = 1 E = 1	10.4	14.3	0.02
Correlated SMP U = 1.5			
S =1 H = 1 E = 1	22.6	18.26	0.38
S =1 H = 0 E = 1	22.6	19.68	0.11
S =1 H = 1 E = 0	22.6	17.93	1.48
S =0 H = 1 E = 1	11.2	16.56	0.02

Table 21: Comparison between SMP and SM with Corr. lists results

The first observation that can be made is that in the correlated results happiness is worse compared to the SMP results, and also, for higher correlation it has almost a doubled value. This is quite logical, since now everyone's preferences are more similar, so it is harder for everyone to be just as satisfied as when the lists are created more randomly.

Secondly, egalitarian happiness in both correlated problems is better than in the original problem. This was expected and according to theory (Caldarelli and Capocci, 2000), is due to the fact that the men are going to be less satisfied now because they will get worse matches than with random lists, and in addition, the more popular women will get better results since they will receive more proposals. Those two facts combined lead to the improvement of the egalitarian happiness value.

However, egalitarian happiness when U =1.5 is worse than what it is for U=1, when E=1. This is attributed to the fact that there is a fewer number of stable matchings now. Since everyone's preferences are more similar it is only logical that it is harder to find stable matchings. So when we try to optimize egalitarian happiness along with stability, the results are less fair because there are not so many stable or almost stable solutions to choose from.

5 Conclusions

In this chapter we show how our results are in accord with our objectives.

5.1 Objectives

The first objective we had set was answered with the help of literature study. We saw that matching problems that are instances of SMP exist in important parts of society like in education and in the military, or in technological areas like in router technology. We also saw that most instances are not straightforward representations of the SMP.

Since it has been studied extensively, it is not surprising that a variety of algorithms exist for the SMP. Most of them focus only on stable matchings. But the Navy example shows that stable solutions are not always the best choice. And the human factor of satisfaction led people to trying to find ways of fooling the system, to improve their situation.

So it is obvious, that different approaches to the problem would benefit the participants, especially if they are willing to compromise on the matter of stability.

Let us not forget that in real world situations stability might not play such an important role, otherwise people would choose to follow the SMP to find their partners. These facts combined with the more realistic Correlated lists example, can lead us to the conclusion that maximum egalitarian solutions of the problem might be more preferable than just stable ones.

That brings us to our second objective. We implemented the SMP and the SM with Correlated lists, and we came up with a function that, through the use of weight constants it tried to improve at least two criteria at the same time. Our results, were satisfying and yielded better results than the GSS, but that was achieved by sacrificing a number of stable pairs. Especially for egalitarian happiness many of our results were much closer to optimality than the GSS results.

We also showed that correlation affects happiness in a negative way, while egalitarian happiness improves for a small value of correlation, but then decreases as correlation increases.

It is clear that the benefits of using GA implementations for solving the SMP do not lie on finding optimal solutions, like in any polynomial algorithm, but on the combination of many criteria that are important to the 'matchmaker', in order for him to create solutions that focus on increasing the participants' happiness.

6 Summary

We have studied the stable marriage problem and have given a detailed description of its variations and its real world instances. We have shown the implications and what changes those real world instances have and what modifications were made by other researchers to handle them. Afterwards, we created an genetic algorithm representation for the original Stable Marriage Problem and also for an instance of the Stable Marriage Problem with Correlated lists, something that has not been done before. Through the use of a genetic algorithm representation we showed that we can get better happiness matches and especially

egalitarian happiness matches by relaxing the stability restrictions. We came to the conclusion that while correlation leads to more fair solutions, if it increases highly, egalitarian happiness starts to decrease.

The significance of the SMP is clearly proven by all its variations and applications in real life. Along with that every algorithm that manages to improve the certain aspects of the solutions and not just stability should be beneficial.

The GA manages to give an alternative point of view on the situation by providing with solutions that can raise the system's happiness by holding stability on high levels.

7 Appendix

Gale Shapley Algorithm

```
function marriage = DS_SMP(n,listm,listw)
      freem = (1:n);
      statusw = zeros(n,1);
      marriage = zeros(1,n);
      while (~isempty(freem))
        man = listm(freem(1),:);
        for j = 1:length(man)
            if (statusw(man(j)) == 1)
               a = listw(man(j),:);
               if (find(a == freem(1)) < find(a == marriage(man(j))))
                  freem(length(freem) + 1) = marriage(man(j));
                  marriage(man(j)) = freem(1);
                  freem(1) = [];
                  break
                  end
               else
                     statusw(man(j)) = 1;
                     marriage(man(j)) = freem(1);
                     freem(1) = [];
                     break
               end
        end
      end
return
```

Fitness Calculating Function

```
function ehap= marriage_fitness(x,listm,listw, mp)
[m n] = size(x);
scores = zeros(m,1); %number of stable pairs
satw = zeros(m,1);
satm = zeros(m,1);
hapm = zeros(m,1);    %men happiness
hapw = zeros(m,1);    %women happiness
hap = zeros(m,1);     %total happiness
ehap = zeros(m,1);    %egaliterian happiness
```

```
A = 1;
B = 1;
C = 1;
idx = 1:n;
es = 0;

for j = 1:m
    p = x(j,:);
    if A>0
        for k = 1:n
            w = listw(k,:);% for each womans list of preferences
            i = 1;
            s = 1;
            while w(i) ~= p(k)
                if (mp(k,i) < idx(listm(w(i),:) == (idx(p == w(i)))))
                    s=0;
                    break
                end
                i = i +1;
           end
           if s==1
              scores(j)= scores(j) +1;
           end
        end
        scores(j) = n +1 - scores(j);
    end

    if B>0 || C>0
        for i = 1:n
            sw = idx(listw(i,:) == p(i));
            satw(j) = satw(j) + sw;
        end

        for i = 1:n
            sm = idx(listm(p(i),:) == i);
            satm(j) = satm(j) + sm;
        end

        hap(j) =   (satw(j) + satm(j))/n;
        hapm(j) =  A*scores(j) + B*satm(j);
        hapw(j) =  A*scores(j) + B*satw(j);
        es = (satm(j) - satw(j))/n;
        if es<0
            es = es*(-1);
        end
    end

    ehap(j) = A*scores(j) + B*hap(j) + C*(es) ;
end
```

Initialization Function

```
function x = init(NVARS,marriagefnc,options)
```

```
totalPopulationSize = sum(options.PopulationSize);
n = NVARS;
x = zeros(totalPopulationSize,n);

for i = 1:totalPopulationSize
      x(i,:) = randperm(n);
end
```

Crossover Function

```
function xoverKids  = crossover(parents,options,NVARS, ...
    marriagefnc,ehap,x)

    Kids = length(parents)/2 - 1;
    xoverKids = zeros(Kids,NVARS);
    index = 1;

    for i=1:Kids

        child1 = zeros(NVARS,1);
        child2 = zeros(NVARS,1);
        parent1 = x(parents(index),:);
        parent2 = x(parents(index+1),:);
        index = index + 2;

        p = ceil((length(parent1) -1) * rand);
        while isempty(find(child1 == parent1(p), 1))
            child1(p) = parent1(p);
            child2(p) = parent2(p);
            p = find(parent1 == parent2(p),1);
        end

        p = find(child1 == 0);
        for j = 1:length(p)
            child1(p(j)) = parent2(p(j));
            child2(p(j)) = parent1(p(j));
        end

        xoverKids(i,:) = child1;
        xoverKids(i+1,:)= child2;

    end
```

Mutation Function

```
function mutationChildren = mutate_permutation(parents,options,NVARS, ...
    marriagefnc, state,ehap,x,listm,listw,mp)

idx = 1:NVARS;
mutationChildren = zeros(length(parents),NVARS);
for i=1:length(parents)
    parent = x(parents(i),:);
    child = parent;
    d = rand;
```

```
    uns = zeros(NVARS,1);
    q=1;
    if (d > 0.5)
       for k = 1:NVARS
           w = listw(k,:);% for each womans list of preferences
           j = 1;
           s = 1;
           while w(j) ~= parent(k)
                if (mp(k,j)<idx(listm(w(j),:)==(idx(parent ==w(j)))))
                    s=0;
                    break
                end
                j = j +1;
           end

           if s==0
               uns(q) = k;
               q = q+1;
           end
       end
       sp = find(uns);
       if ~isempty(sp)
           p1 = ceil(length(sp) * rand);
           p2 = ceil(length(parent) * rand);
           child(sp(p1)) = parent(p2);
           child(p2) = parent(sp(p1));
       end
    else
       for j = 1:ceil(NVARS/20)
           par = child;
           p = ceil(length(parent) * rand(1,2));
           child(p(1)) = par(p(2));
           child(p(2)) = par(p(1));
       end
    end
    mutationChildren(i,:) = child;
end
```

8 References

Aldershof B. and Carducci O. M., (1999), Stable marriage and genetic algorithms: a fertile union. J. Heuristics (Netherlands), 5(1):29–46.

Biro, P. and Manlove, D.F. and Mittal, S., (2010), *Size versus stability in the marriage problem.* Theoretical Computer Science, 411 (16-18). pp. 1828-1841. ISSN 0304-3975

Brito I. and Meseguer P., (2005), Distributed stable marriage problem. *Sixth Inter*national Workshop on Distributed Constraint Reasoning (DCR) at the Nineteenth International Joint Conference on Arti¯cial Intelligence.

Bremermann, H.J., (1962), *Optimization through evolution and recombination* In: Self-Organizing systems 1962, edited M.C. Yovitts et al., Spartan Books, Washington, D.C. pp. 93–106.

Caldarelli G. and Capocci A., (2000), Beauty and distance in the stable marriage problem. Physica A: Statistical Mechanics and its Applications, 300:325-331.

Caldarelli G., Capocci A., Laureti P. , (2001), Sex-oriented stable matchings of the marriage problem with correlated and incomplete information, Physica A: Statistical Mechanics and its Applications vol. 299, issn 0378-4371, p. 268—272.

Dean B.C. , Immorlica N., and Goemans M.X., (2006), The unsplittable stable marriage problem. In Proceedings of the 4th IFIP International Conference on Theoretical Computer Science.

Dubins L. E., and Freedman D. A., (1981), "Machiavelli and the Gale-Shapley Algorithm," American Mathematical Monthly, 88: 485-494.

Dzierzawa M. and Omero M., (2002), Statistics of stable marriages. Physica A, 0007321:1-7.

Fraser A., (1957). "Simulation of genetic systems by automatic digital computers. I. Introduction". *Aust. J. Biol. Sci.* 10: 484–491.

Gale D., Shapley L., (1962), College admissions and the stability of marriage, Amer. Math. Monthly 69 9–15.

Gale D. and Sotomayor M., (1985), "Some remarks on the stable matching problem," Discrete Applied Mathematics, Vol.11, pp.223-232.

Gale D. and Sotomayor M., (1985), Ms Machiavelli and the stable matching problem. *Amer. Math. Monthly* 92 261–268.

Garrett J. D. , Vannucci J. ,Silva R. , Dasgupta D. , and Simien J. , (2005), Genetic algorithms for the sailor assignment problem. In Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-05). ACM press.

Goldberg D.E., (1989), *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Grefenstette J., Gopal R., Rosmaita B., and Van Gucht D., (1985). Genetic Algorithms for the Traveling Salesman Problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, John J. Grefenstette (Ed.). L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 160-168.

Gusfield D., (1987), Three fast algorithms for four problems in stable marriage. SIAM J.Comput., p.111–128.

Gusfield D. and Irving R.W., (1989), The Stable Marriage Problem: Structure and Algorithms. MIT Press.

Holland, J.H., (1992), *Genetic algorithms*, Scientific American Vol. 267, p. 66-72.

Irving R.W., (1985), An efficient algorithm for the "stable roommates" problem, J. Algorithms 6 (1985) 577–595.

Irving R. W. and Leather P. , (1986), "The complexity of counting stable marriages," *SIAM J. Comput.*, Vol.15, pp. 655–667.

Irving R.W., Leather P., Gusfield D., (1987), An efficient algorithm for the "optimal" stable marriage. J. ACM, p. 532–543.

Irving R. W., (1994), Stable marriage and indifference. In *Selected papers of the conference on Combinatorial Optimization* (CO89), Martin Dyer, Chris Potts, and Les Proll (Eds.). North-Holland Publishing Co.,

Amsterdam, The Netherlands, The Netherlands, 261-272.

Irving R. W., and Manlove D. F., (2002), The stable roommates problem with ties. *Journal of Algorithms* 43(1):85-105.

Irving, R.W. and Manlove, D., (2007), An 8/5 approximation algorithm for a hard variant of stable marriage. In, Fleischer, R. and Trippen, G., Eds. *Proceedings of COCOON 2007, 13th Annual International Computing and Combinatorics Conference, 16-19 July 2007* Lecture Notes in Computer Science Vol 4598, pages pp. 548-558, Banff, Canada.

Irving R. W., (2008), Stable matching problems with exchange restrictions. Journal of Combinatorial Optimization.

Irving, R.W, and Manlove, D.F. and Scott, S., (2008), *The stable marriage problem with master preference lists.* Discrete Applied Mathematics, 156(15). pp. 2959-2977. ISSN 0166-218X

Iwama K. , Manlove D. , Miyazaki S. , and Morita Y., (1999), Stable marriage with incomplete lists and ties. In Proceedings of ICALP '99: the 26th International Colloquium on Automata, Languages, and Programming, volume 1644 of Lecture Notes in Computer Science, pages 443-452. Springer-Verlag.

Iwama K., Miyazaki S., Yamauchi N., (2007), A 1.875-approximation algorithm for the stable marriage problem *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* pp. 288–297.

Iwama K. and Miyazaki S., (2008), A survey of the stable marriage problem and its variants. In ICKS '08: Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008), pages 131–136, Washington, DC, USA,. IEEE Computer Society.

Kato A., (1993), "Complexity of the sex-equal stable marriage problem," *Japan Journal of Industrial and Applied Mathematics (JJIAM)*, Vol. 10, pp. 1–19.

Knuth D. E. , (1976), "Mariages Stables," Les Presses del'Universit´e Montr´eal, (Translated and corrected edition, "Stable Marriage and Its Relation to Other Combinatorial Problems," CRM Proceedings and Lecture Notes, Vol. 10, American mathematical Society, 1997.

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

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

Manlove, D., (2002), The structure of stable marriage with indifference. Discrete Applied Mathematics 122(1-3):167-181.

Marx D. and Schlotter I., (2010), Parameterized complexity and local search approaches for the stable marriage problem with ties, In Algorithmica vol. 58, Springer, p. 170-187.

Nakamura M., Onaga K., Kyan S. and Silva M.,(2002), Genetic algorithm for sex-fair stable marriage problem, IEEE International Symposium on Circuits and Systems, 1995.vol. 1, pp 509-512,ISBN 0780325702.

Pini M.S, Rossi F., Venable K.B. and Walsh T., (2010), Stable marriage problems with quantitative preferences, Arxiv preprint arXiv:1007.5120.

Prabhakar B. , and McKeown N., (1999), "On the speedup required for combined input and output queued switching," Computer Systems Lab, Stanford Univ., Stanford, CA, Tech. Rep. CSL-TR-97-738.

Roth A.E. and Vande Vate J.H., (1990), Random Paths to Stability in Two-Sided Matching, Econometrica Vol. 58, No. 6 , pp. 1475-1480

Roth E., and Sotomayor M.A.O., (1992), Two-sided matching: A study in game-theoretic modeling and analysis, Cambridge Univ Press,ISBN 0521437881.

Roth E. and Peranson E., (1999), The redesign of the matching market for american physicians: Some engineering aspects of economic design. American Economic Review, 89:748–780.

Ronn E., (1986), On the complexity of stable matchings with and without ties. PhD thesis, Yale University.

Ronn E., (1990), NP-complete stable matching problems. Journal of Algorithms, 11:285–304.

Schwefel, H. P. , (1977), *Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie*. Basel; Stuttgart: Birkhäuser. ISBN 3764308761.

Schwefel, H.P., (1981), *Numerical optimization of computer models (Translation of 1977 Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie*. Chichester; New York: Wiley. ISBN 0471099880.

Sywerda G., (1989), *Uniform crossover in genetic algorithms*. In Proceedings of the third international conference on Genetic algorithms, J. David Schaffer (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2-9.

Yanagisawa H., (2007), Approximation Algorithms for Stable Marriage Problems PhD Thesis.

University of Borås is a modern university in the city center. We give courses in business administration and informatics, library and information science, fashion and textiles, behavioral sciences and teacher education, engineering and health sciences.

In the **School of Business and Informatics (IDA),** we have focused on the students' future needs. Therefore we have created programs in which employability is a key word. Subject integration and contextualization are other important concepts. The department has a closeness, both between students and teachers as well as between industry and education.

Our **courses in business administration** give students the opportunity to learn more about different businesses and governments and how governance and organization of these activities take place. They may also learn about society development and organizations' adaptation to the outside world. They have the opportunity to improve their ability to analyze, develop and control activities, whether they want to engage in auditing, management or marketing.

Among our **IT courses**, there's always something for those who want to design the future of IT-based communications, analyze the needs and demands on organizations' information to design their content structures, integrating IT and business development, developing their ability to analyze and design business processes or focus on programming and development of good use of IT in enterprises and organizations.

The **research** in the school is well recognized and oriented towards professionalism as well as design and development. The overall research profile is Business-IT-Services which combine knowledge and skills in informatics as well as in business administration. The research is profession-oriented, which is reflected in the research, in many cases conducted on action research-based grounds, with businesses and government organizations at local, national and international arenas. The research design and professional orientation is manifested also in InnovationLab, which is the department's and university's unit for research-supporting system development.

UNIVERSITY OF BORÅS
SCHOOL OF BUSINESS AND INFORMATICS

Visiting address: Järnvägsgatan 5 · Postal address: Allégatan 1, SE-501 90 Borås
Phone: + 46 33 435 40 00 · E-mail: inst.ida@hb.se · Web: www.hb.se/ida