Fair Stable Marriage - Term Project

Social Computing - Fall 2018

Swapna Mukrappilly

Jason Trout

Zach Southwell

Joshua Musick

November 30, 2018

Abstract

We explore the problem of finding a stable matching that is 'fair' in terms of overall satisfaction for both men and women. We present a polynomial time solution for efficiently traversing the set of all stable matchings and present details about our implementation. Finally, we discuss the results of our implementation with problem sets of various sizes.

1 Introduction

1.1 Stable Marriage Problem

The Stable Marriage Problem is a matching between two groups, men and women, of equal size, where each person has an ordered preference list containing all members of the opposite group. For a matching, each member of the "proposing" group must be matched with a member of the "accepting" group. Furthermore, the matching must be *stable*, which means there can be no blocking pairs. A blocking pair is a man and woman that both prefer each other over their current match. As shown by Gale and Shapely [1], there is always at least one stable matching for any input.

The Gale-Shapely algorithm is effective in finding a stable matching. This algorithm, however, is biased in favor of one group, the "proposer". For the proposing group, each and every member is guaranteed to be matched with their best stable match. Alternately, each member of the "accepting" group is guaranteed to be matched with their worst possible stable match.

1.2 Fair Matching

The systematic bias of the Gale-Shapely algorithm is clearly not optimized for fairness. Fairness, however, must be defined for both groups and be measurable. We will define the function $score(p, m)$

to be the position of a person p's assigned match m in their preference list. With this definition, a lower individual score is a better outcome for that individual. For example, if a man M_1 has the preference list [W_3, W_1, W_4, W_2], and (M_1, W_4) are matched in a matching, then $score(M_1, W_4)$ is 2. Then, we can define the group score of group G as: $\sum score(p, m)\ \forall p \in G$. Using this measurement, consider two possible definitions of fair stable matching, balanced and optimal.

- **Balanced Fair Matching -** A matching that minimizes the difference of the group scores.
- **Optimal Fair Matching -** A matching that minimizes the sum of the group scores.

Fairness for this project was measured using the **Optimal Fair Matching** method. This method was choosen because the balanced method could very easily produce ”fair” matchings that give horrible group scores. The optimal method, starting from a stable matching, always looks for a matching that has a lower total score, which means someone in the matching is getting a better preference than before. This improvement has a finite limit and does not give a bias toward either group. It does have the potential to produce group scores that may be unbalanced, but overall satisfaction is better.

A simple example would be a balanced matching that gave a men's group score of 28 and a women's group score of 29; a difference of only 1. But suppose the optimal matching gave a men's group score of 22 and a women's group score of 25. In this situation, using the optimal matching, both groups are getting a better deal, even though it slightly favors the men.

2 Finding Stable Matchings

The most challenging part of finding the optimal matching is finding the set of all stable matchings. A naive approach to this problem is the brute force strategy. However, by applying some simple heuristics, one can reduce the problem space.

2.1 Brute Force Strategy

A brute force strategy includes finding all permutations of matchings, determine which are stable, calculating the ”fairness” of each matching, and keeping the matching with the lowest value. Clearly, as the input size grows, this solution quickly becomes intractable. However, for this project, initially we did use a modified version of this approach.

2.2 Eliminating Non-feasible Matchings

If we apply the Gale-Shapely algorithm to find the man-optimal and woman-optimal matchings, then we know the optimal and pessimal matches for each individual. For any individual i, any matching with a partner that falls outside the range of i's optimal and pessimal match is not a feasible pairing.

Furthermore, non-feasible matches for m_i to w_j can also be removed from m_i's feasible list, if m_i is not a feasible match for w_j. This is demonstrated by the removal of W_2 and W_4 from M_2's feasible preference list in Table 1.

The infeasible matchings are pruned from the preference list (left side) with the different types of pruning in different colors (right side) in Table 1. Red removals are from the man-optimal pruning, blue removals are from woman-optimal, and green removals are due to matches that are infeasible as a result of the previous two removals.

Original Preferences					Reduced Feasible Preferences				
M_1:	W_1	W_4	W_2	W_3	M_1:	W_1	W_4	~~W_2~~	~~W_3~~
M_2:	W_3	W_2	W_4	W_1	M_2:	W_3	~~W_2~~	~~W_4~~	W_1
M_3:	W_2	W_1	W_3	W_4	M_3:	W_2	~~W_1~~	~~W_3~~	~~W_4~~
M_4:	W_1	W_4	W_3	W_2	M_4:	~~W_1~~	W_4	W_3	~~W_2~~
W_1:	M_2	M_3	M_1	M_4	W_1:	M_2	~~M_3~~	M_1	~~M_4~~
W_2:	M_3	M_1	M_4	M_2	W_2:	M_3	~~M_1~~	~~M_4~~	~~M_2~~
W_3:	M_4	M_2	M_3	M_1	W_3:	M_4	M_2	~~M_3~~	~~M_1~~
W_4:	M_1	M_4	M_2	M_3	W_4:	M_1	M_4	~~M_2~~	~~M_3~~

Table 1: Original and Pruned Preference Lists

2.3 The Improved Brute-force Algorithm

If we apply the heuristic mentioned above, we can improve the brute force algorithm as follows:

1. Calculate the man-optimal stable matching with the Gale-Shapely algorithm.
2. Calculate the woman-optimal stable matching with the Gale-Shapely algorithm.
3. For each person, identify pairings that occur outside the range of the optimal and pessimal pairing as non-feasible.
4. For each non-feasible pairing found, remove the man from the woman's reduced preference list, and remove the woman from the man's reduced preference list.
5. Permute over all possible arrangements of the reduced preference lists. For each permutation:
   (a) Calculate whether the matching is stable.
   (b) For unstable matches, skip to the next permutation.
   (c) For stable matches, calculate the cumulative score of the match (using the full preference list) and record the lowest value.

While this is an improvement over the original algorithm, it's still an $O(2^n)$ run-time algorithm. We originally implemented this algorithm and most problem sets of 50 pairs did not complete in a reasonable amount of time.

2.4 Non-Feasible Prune Method

In conjunction with discovering the Gusfield/Irving [2] rotation method described below, we also came across a different method for pruning non-feasible pairs. This method uses the man-optimal / woman-optimal matching to identify all non-feasible matches for all men and women. Then the men and women feasible preference lists are updated. This method produced similar results to our heuristics, but with different performance times, as shown in Table 3.

3 Rotations

After implementing the brute-force approach above, we tried again using a strategy of match rotations detailed by Gusfield/Irving [2] to efficiently traverse the set of all stable matchings.

To explain this approach, let's define some functions which can be applied to any existing stable marriage:

- Define $currentMatch(p)$ to be the current match for person p
- Define $nextMatch(p)$ as follows: the first person in the preference list for person p, who prefers p over their current partner. Necessarily, $nextMatch(p)$ must follow $currentMatch(p)$ in p's preference list, otherwise the matching would not be stable. Of course, it is also possible that $nextMatch(p)$ does not exist. This is the case for any person matched with their pessimal match. As an example, looking at the problem set shown in table 3, we can use the reduced preference list to identify $nextMatch(p)$. For the man-optimal matching, $currentMatch(M_2) = W_3$, and $nextMatch(M_2) = W_1$.
- Define $consolationMatch(p)$ as the $currentMatch(nextMatch(p))$. In other words, the current match of $nextMatch(p)$. For our example shown in table 3, suppose we are taking the man-optimal match where $currentMatch(M_1) = W_1$, then $consolationMatch(M_1) = M_4$.
- Define a rotation in a matching as any ordered subset of matched pairs in a matching such that: for each matched pair i: $(0 \le i \le r)$, $consolationMatch(M_i) = M_{i+1}$ where r is the cardinality of the subset of matched pairs and $i + 1$ is taken modulo r. Using the example from table 3: the set of men has the following rotation: (M_1, M_4, M_2):
  $consolationMatch(M_1) = M_4$
  $consolationMatch(M_4) = M_2$
  $consolationMatch(M_2) = M_1$
- Let M be a stable matching and p be some rotation exposed in M. Then define M/p as the matching that results from re-assigning all men in the rotation to $nextMatch(M_i)$

Lemma 3.1 *If M is a stable matching and p is a rotation in M, then M/p is a stable matching.*

Proof: Suppose (m, w) is a blocking pair in M/p. All women in M/p either have the same partner in M or a new partner that they prefer more, so if w prefers m in M/p, then w prefers m in M. So m must be in the rotation p otherwise (m, w) would be a blocking pair in M. So m prefers w less than his matching in M, but more than his matching in M/p, but this violates the definition of the $nextMatch(m)$ function used to define a rotation.

3.1 The Set of All Stable Matchings as a Distributive Lattice

A stable matching can be expressed as a vector of $score(M_i, currentMatch(M_i))$. The set of stable matchings expressed in this way forms a distributive lattice. Consider the preference list in Table 2.

Men's Preferences						Women's Preferences					
M_1:	W_1	W_3	W_2	W_4	W_5	W_1:	M_3	M_2	M_4	M_5	M_1
M_2:	W_5	W_1	W_3	W_2	W_4	W_2:	M_2	M_5	M_4	M_1	M_3
M_3:	W_2	W_3	W_5	W_4	W_1	W_3:	M_5	M_3	M_1	M_4	M_2
M_4:	W_3	W_1	W_5	W_4	W_2	W_4:	M_4	M_1	M_3	M_5	M_2
M_5:	W_4	W_5	W_2	W_3	W_1	W_5:	M_4	M_2	M_3	M_5	M_1

Table 2: Sample Preference List

In this scenario, the man-optimal matching can be expressed as: $[0, 0, 0, 0, 0]$. The man-optimal matching has all men assigned to the woman who is their first choice. If we use the rotation algorithm described above, we can traverse the entire lattice of stable matchings from the man-optimal to the woman optimal. In this case, there are 8 possible stable matchings, shown in the Hasse diagram in Figure 1.

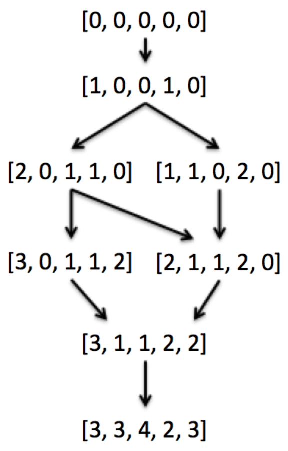
$$\begin{tikzcd}
 & {[0,0,0,0,0]} \arrow[d] & \\
 & {[1,0,0,1,0]} \arrow[dl] \arrow[dr] & \\
{[2,0,1,1,0]} \arrow[d] \arrow[drr] & & {[1,1,0,2,0]} \arrow[d] \\
{[3,0,1,1,2]} \arrow[dr] & & {[2,1,1,2,0]} \arrow[dl] \\
 & {[3,1,1,2,2]} \arrow[d] & \\
 & {[3,3,4,2,3]} &
\end{tikzcd}$$

Figure 1: Lattice Representation of all Stable Matches

3.2 Rotation algorithm

The algorithm using rotations to traverse through the lattice of all stable matchings is given below:

1. Use the Gale-Shapely algorithm to calculate the man-optimal stable matching
2. Use the Gale-Shapely algorithm to calculate the woman-optimal stable matching
3. Use the man-optimal and woman-optimal matchings to identify pairings that are not feasible:
    (a) For a man M, any woman W that occurs in M's preference list prior to M's optimal match or after M's pessimal match is not feasible.

   (b) Likewise, for a woman W, any man M that occurs in W's preference list prior to W's optimal match or after W's pessimal match is not feasible.

4. For each person, build a reduced preference list by taking the full preference list, then for each non-feasible pairing(M_i, W_j), remove M_i from the reduced preference list of W_j and remove W_j from the reduced preference list of M_i

5. Calculate the cumulative *score* for all people in the man-optimal matching.

6. Starting with the man-optimal matching, execute the following:

7. Identify any rotations in the given set. For each rotation found:

   (a) Take the matching produced by applying the rotation.

   (b) If the rotation has been visited already, return.

   (c) Calculate the cumulative *score* for all people in the matching. If the score is the lowest found so far, record the score.

   (d) Repeat step 7 with the rotated matching.

This algorithm was able to calculate optimal matching much more efficiently and handled larger data sets. Still, traversing all possible matchings becomes more and more computationally expensive as the number of pairs grow. The runtime to perform the rotation matching of large data sets is shown in Table 3.

4 Results

As noted above, the brute-force strategy was initially tested, but proved to be too inefficient even on small input data sets, therefore we did not do any performance testing of that method. However, we did find an alternate method for determining infeasible matchings in Gusfield Irving [2], in conjunction with with the rotation method. We implemented both and found that our *heuristics* and their *Non-Feasible Prune* method produced the same feasible preference list. The average runtime for each method, however, was not.

4.1 Runtime Performance

For performance testing, the runtime of Gale-Shapley(GS) (for men and women optimal), our Heuristics for pruning feasible matchings, the Non-Feasible Prune method, and the Rotation Matching algorithms were all measured. The results in Table 3 are averages for multiple runs of a set of different input preference lists for each input n size. It is easy to see that our heuristics took much longer to run than the non-feasible prune method. Larger inputs were tried, but due to memory constraints and algorithm implementation details, larger than 2000 would not run.

Input n	GS	Heuristic Prune	Non-Feasible Prune	Rotation Matching
10	0.035	0.148	0.025	0.07
100	2.05	4.539	2.66	25.01
1000	254	3230	908	64,147
2000	1336	27,382	4455	762,314

Table 3: Runtime Performance (in msec)

4.2 Fairness Performance

Using the "Optimal Fair Matching" method, we calculated the fairness of a number of different input sizes and randomized preference lists. Figure 2 shows average percentages of man and women scores for man-optimal, woman-optimal, and our fair solution. Each pair of bars represents the average ratio of man vs woman group scores in each solution for each input size n. The different color bars represent the input size n for how many man and women were in the matching. It is easy to notice, as n increases, so does the imbalance in both the man and woman optimal solutions. However, in the "Fair Solution", the men and women scores are approximately 50% for both groups.

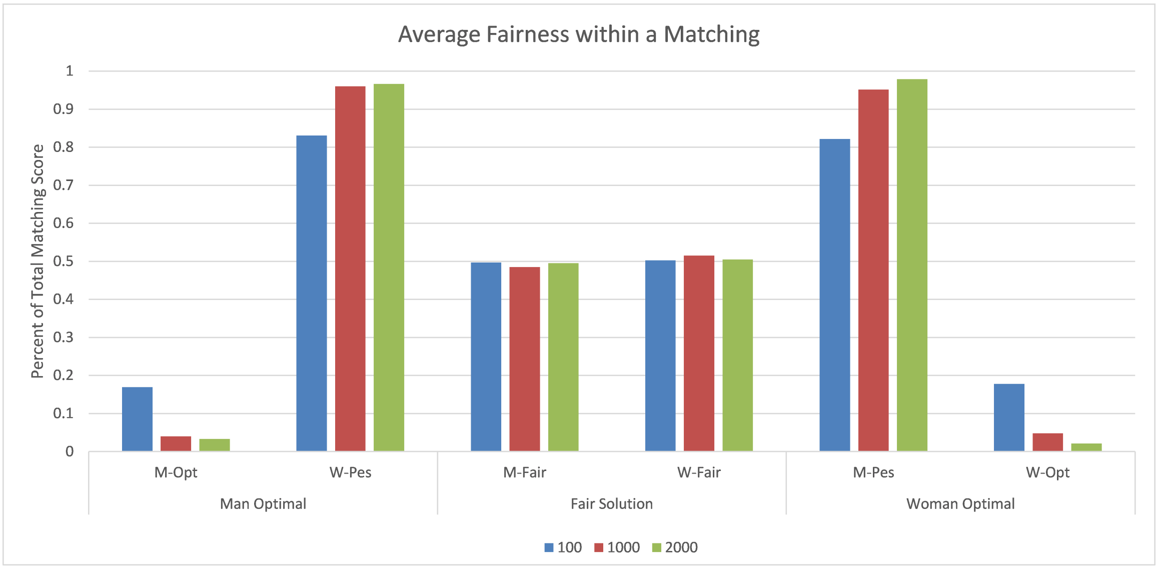

Figure 2: Average Fairness Comparison within a Matching

The fair matching is balanced for men and women, but should also be a good solution relative to something. Figure 3 shows the comparison of matchings for men and women relative to their pessimal matching. The left three sets of bars are the average scores for men relative to the man-pessimal matching. The three sets of bars on the right compare the women average scores. Since the man-pessimal (or woman-pessimal) matching is the worst score possible, it is a good gauge for how much better the other matchings are. As shown, the fair matching gives scores relatively close to the optimal score for both groups, not just one.

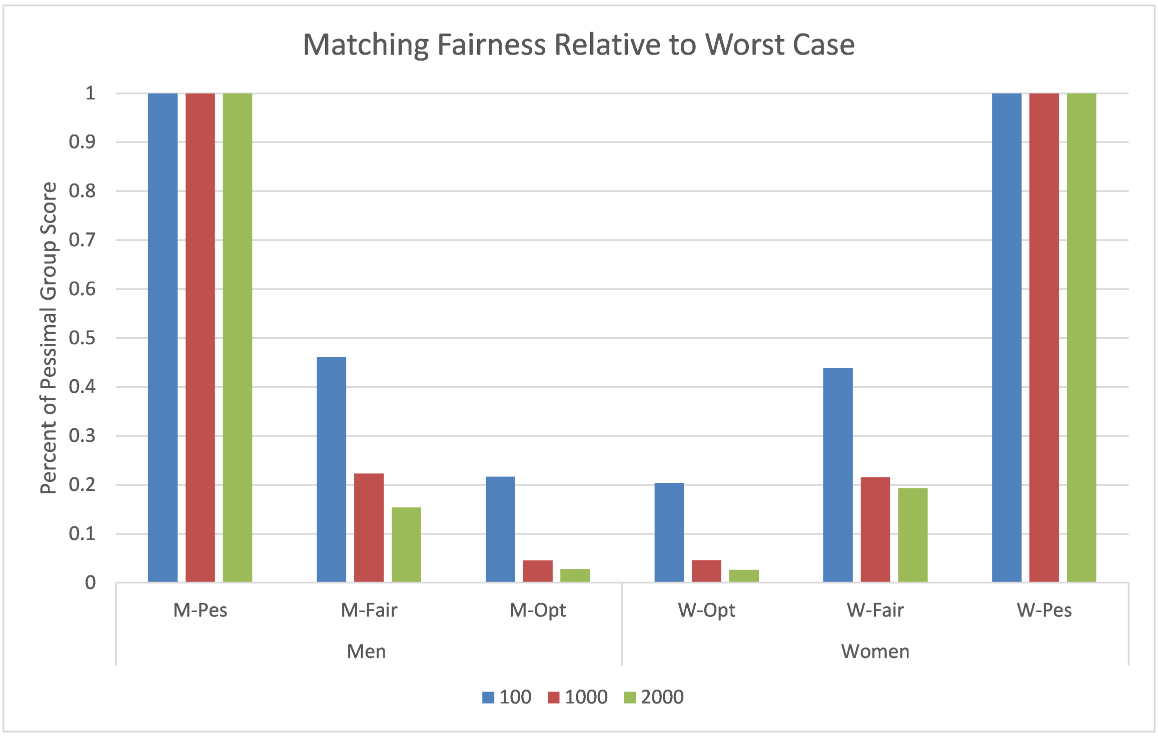

Figure 3: Fairness Compared to Worst Case

5 Conclusions

This paper demonstrated a reasonable solution to finding an optimal stable matching. This problem turns out to be more difficult than the problem of simply identifying a stable matching, which the Gale-Shapely algorithm solves. Instead, finding an optimal stable matching requires us to traverse the set of all stable matchings. Although we've shown that this can be done with a polynomial time algorithm ($O(n^4)$), run times become very large as the number of pairs grow, as demonstrated in section 4.1. For larger problem sets, it may be more desirable to find an *approximation* of the optimal solution.

The results of this analysis is heavily dependent upon input. If a carefully constructed preference list were given, the results could be skewed in a number of different ways. This potential issue was mitigated using many random preference lists as the basis for each set of runs.

References

[1] D. Gale and L. S. Shapley Jan 1962, *College Admissions and the Stability of Marriage, The American Mathematical Monthly, Vol. 69, No. 1 pp. 9-15.*

[2] Dan Gusfield and Robert W. Irving, *The Stable Marriage Problem: Structure and Algorithms 1989.*