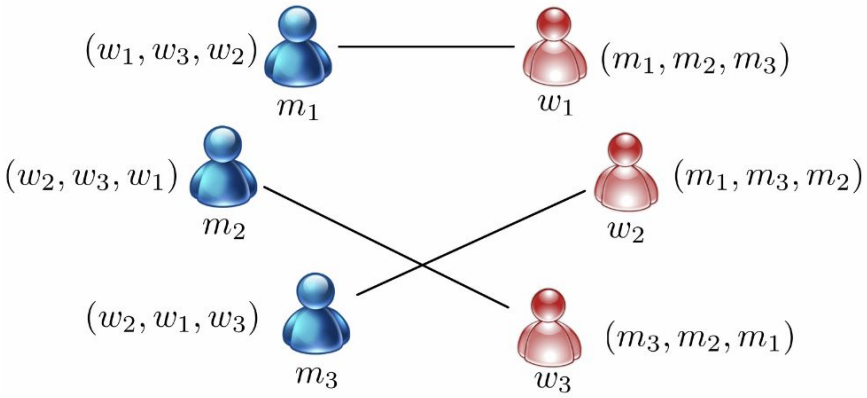
Fair Stable Marriage

By: Swapna Mukrappilly, Jason Trout, Zach Southwell, & Joshua Musick

Stable Marriage Problem

Given a ordered preference list of men and women, find a stable matching.



There should be no blocking pairs. A pair (m,w) is blocking in a matching M if they prefer each other to their current partners.

Gale-Shapley Algorithm

The Gale-Shapley Algorithm finds a stable match. However, it is strongly biased in favor of one gender / group, the proposing group.

- Each member in the “proposing” group is matched with their best stable match.
- Each member in the “accepting” group is matched with their worst stable match.

How do we find a matching that is fair for both genders / groups?

Fair Matching

Our heuristic for measuring fairness is Score. A Score (p,m) is defined as the position of a person p's assigned match m in their preference list.

M_1: W_1 W_4 W_2 W_3
M_2: W_3 W_2 W_4 W_1
M_3: W_2 W_1 W_3 W_4
M_4: W_1 W_4 W_3 W_2

W_1: M_2 M_3 M_1 M_4
W_2: M_3 M_1 M_4 M_2
W_3: M_4 M_2 M_3 M_1
W_4: M_1 M_4 M_2 M_3

$\text{Score}(\text{M\_1}, \text{W\_1}) = 0$
$\text{Score}(\text{M\_1}, \text{W\_2}) = 2$

A lower score is a better outcome for p. We will use this metric to measure satisfaction of an individual.

How Do We Define Fair Matching?

There are different approaches to finding a fair stable matching.

- Balanced Fair Matching - sum of the scores for all men is closest to the sum of all scores for women.
- Optimal fair matching - matching that minimizes the total cumulative score for all participants of the matching. We use this criteria.

Can be more favorable to one gender than the other, but this is the matching that produces the highest overall satisfaction.

Our Approaches to Finding Fair Matching

We tried three different approaches to find a fair stable match.

- Brute Force Solution
- Brute Force on a reduced problem set (remove non-feasible)
- Rotation Method (permutations to a stable matching)

Brute Force Solution

1. Compute all possible matchings
2. Determine which matchings are stable
3. If stable, calculate the cumulative score of the match
4. Save the matching with the lowest cumulative score

This approach is not feasible for large problem sets.

How to Reduce Possible Matchings

Remove non-feasible based on man-optimal matching (red)

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: Prune Based on Man-Optimal Matching

How to Reduce Possible Matchings

Remove non-feasible based on woman-optimal matching (blue)

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 2: Prune Based on Woman-Optimal Matching

How to Reduce Possible Matchings

Remove non-feasible based on non-mutually feasible pairs (green)

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 3: Pruned Based on Non-Mutual Feasible

Rotations

The brute force approach has us checking every possible matching to find the set of *stable* matchings. Clearly, the computation required to permute through every possible matching grows exponentially as the problem set increases.

Is there a more efficient way to find the set of all stable matchings?

We used a strategy detailed by Gusfield and Irving[1] to do this efficiently by identifying ‘rotations’ available in existing stable matchings.

To explain this strategy, we’ll first define some functions that will help explain the concept of a rotation.

Rotations: Definitions

currentMatch(p): Given a stable matching, the currently assigned match of person p.

nextMatch(p): Given a stable matching, the first person in the preference list for person p, who prefers p over their current partner.

For a reduced preference list, *nextMatch(p)* is just the next matching after *currentMatch(p)*. For the example shown in table 1:

- *currentMatch(M2)* is W_3
- *nextMatch(M2)* is W_1

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

Rotations: Definitions

consolationMatch(p): Given a stable matching, *currentMatch(nextMatch(p))*.

In our example:

- *consolationMatch(M_2)* is *M_1*

To see why this is the case:

- *nextMatch(M_2)* is *W_1*
- *currentMatch(W_1)* is *M_1*

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

Rotations: Definitions

rotation: any ordered subset of matched pairs in a matching such that: for each matched pair i : $(0 \leq i \leq r)$, $consolationMatch(M_i) = M_{i+1}$ where r is the cardinality of the subset of matched pairs and $i + 1$ is taken modulo r.

Using the same example from table 1: the set of men has the following rotation: (M_1, M_4, M_2):

$consolationMatch\ (M1) = M_4$
$consolationMatch\ (M4) = M_2$
$consolationMatch\ (M2) = M_1$

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

Rotations: Properties

Given some stable matching M and a rotation p, let M/p be the matching that results from re-assigning all the men in the rotation with $nextMatch(M_i)$

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

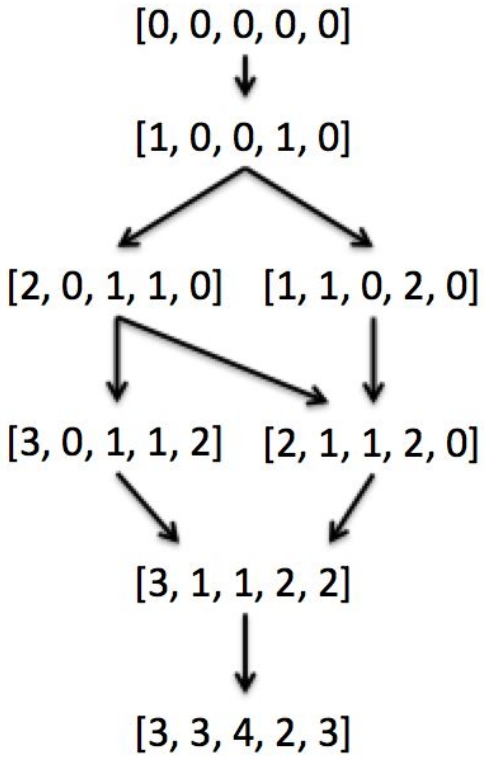
Proof: Suppose (m,w) is a blocking pair in M/p. All women in M/p either have the same partner in M or a new partner that they prefer more, so if w prefers m in M/p, then w prefers m in M. So m must be in the rotation p, otherwise (m,w) would be a blocking pair in M. So m prefers w less than his matching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

The set of stable matchings as a distributive lattice

A stable matching can be expressed as a vector of *score*(M_i). The set of stable matchings expressed in this way forms a distributive lattice. Consider the problem set shown in Table 2. This particular set has 8 possible stable matching configurations.

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



Rotations: Algorithm

By starting with the man-optimal stable matching and recursively finding rotations in the set of men, or similarly by starting with the woman-optimal stable matching and finding rotations in the set of women, the set of all stable matchings can be traversed in polynomial time.

Results

- We initially tested with the brute-force strategy, but this strategy proved too inefficient even with a small number of inputs; therefore, no performance testing was done for this method.
- With the rotation method in conjunction with a method for determining unfeasible matchings from Gusfield and Irving, we determined that the feasible preference list produced using our heuristics and the Non-Feasible Prune was the same; however, the run time performance of generating that list was not the same.

Results: Runtime Performance

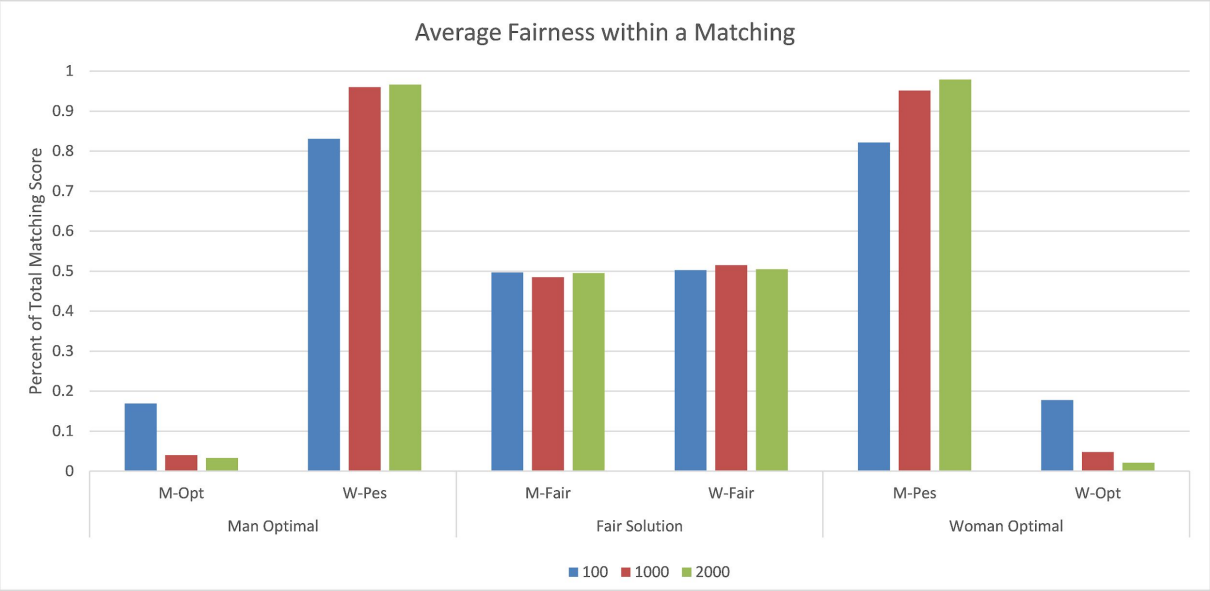
The table below shows averages of multiple runs of a set of different input preference lists for each input n size. Our heuristics took much longer to run than the non-feasible prune method.

Input (n)	Gale-Shapley	Prune Heuristics	Non-Feasible Prune	Rotation Matching
10	0.035	0.15	0.025	0.07
100	2.05	4.54	2.66	25.01
1000	254	3230	908	64,147
2000	1336	27,382	4455	762,314

Runtime Performance : Time given in msec

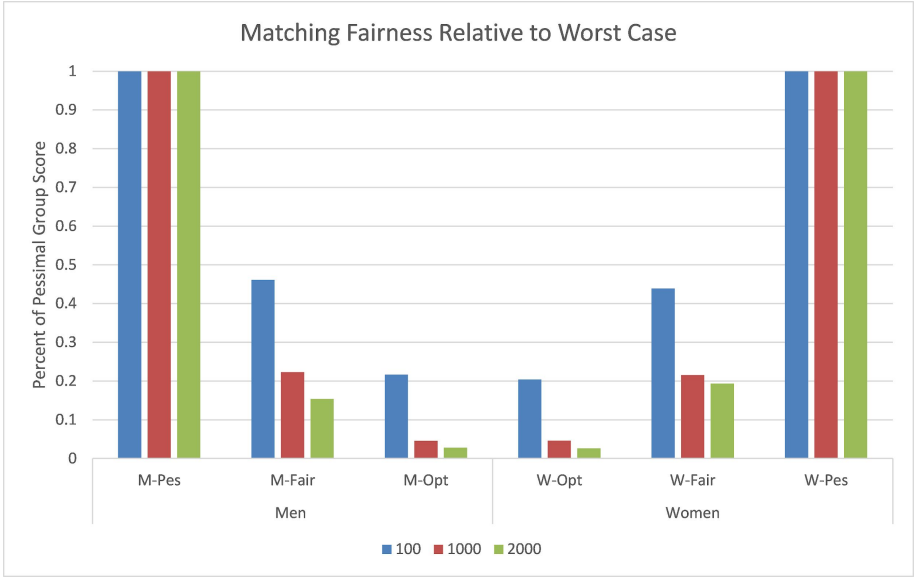
Results: Fairness Performance

Using the “Optimal Fair Matching” method, we calculated the fairness of a number of different input sizes and randomized preference lists. This chart shows the average man and woman scores based on man-optimal, woman-optimal, and our fair solution. Each pair of bars is the ratio of men and women scores for the input size n.



Results: Fairness Performance

Our optimal fair matching solution seems to be a balanced solution, but we need to compare good solutions to something. Here, we compare matchings relative to their pessimal matchings. Since the pessimal matching represents the worst-score possible, it provides a good gauge for how better the other matchings are.



Conclusion

- Our project demonstrates a reasonable solution to finding an optimal stable matching.
- Optimal stable matching is a more difficult problem than simply determining a stable matching--which is what the Gale-Shapely algorithm solves.
- Optimal stable matching can be solved with a $O(n^4)$ algorithm, but for large problem sets, finding an approximation of the optimal solution is more practical.

Questions?

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.