Stable Matchings with Additional Objectives

Craig T. Lennon
Department of Mathematical Sciences, West Point, Thayer Hall, West Point, NY 10996, USA
craigtlennon@gmail.com

In the stable matching problem, n men and n women each list all members of the opposite sex in order of individual preference as a marriage partner. A complete matching is stable if no man and woman mutually prefer each other to their partners in the matching. In the sailor assignment problem, sailors and employers list one another in order of preference, but there is a monetary cost associated with moving a given sailor to a given employer, and a training value measuring a variety of factors relating to the Navy's objective evaluation of how a sailor might fit into a given position. In this paper, we consider a version of the stable matching problem with multiple objectives, like in the sailor assignment problem. Given n applicants and n employers, each with complete preference lists, we want to find matchings which are stable, respect the aggregate preferences of the applicants and employers, and minimize / maximize any additional objectives. We develop a genetic algorithm which searches among the lattice of stable matchings, design and execute an experiment to improve it, and then compare its performance to a single variable optimization algorithm which optimizes each objective function independently.

Key words: stable matching; sailor assignment problem; genetic algorithm; design; programming: assignment, programming: multiple criteria, optimization
History:

---

1. Introduction

In the stable matching problem, n men and n women each list all members of the opposite sex in order of individual preference as a marriage partner. A complete matching is stable if no man and woman mutually prefer each other to their partners in the matching, so stable matchings are efficient in the sense that the matching cannot be rearranged without some person being less happy. Because of this efficiency, and because of the ease of finding a stable matching via the Gale-Shapley algorithm (Gale and Shapely 1962), stable matchings have been used for such varied purposes as pairing medical interns with hospitals (Gusfield

and Irving 1989) and children with schools (Abdulkadiroglu et al. 2005). Stability is a local measure of satisfaction, and does not take account of the aggregate preferences of either side of the matching process. So, for example, while stability means that no man and woman could switch to their mutual advantage, it is possible that an unstable matching might make men and (or) women better off on average. By design, stability takes account only of the preferences of the men and women, and ignores all additional factors associated with the matching (e.g. relocation costs in the interns and hospitals matching).

Additional factors are taken into account in the sailor assignment problem. In this problem, sailors and employers list one another in order of preference. Additionally, there is a monetary cost associated with moving a given sailor to a given employer, and a training value measuring a variety of factors relating to the Navy's objective evaluation of how a sailor might fit into a given position. There are four objective functions measuring how well an assignment respects the aggregate preferences of sailors and employers, how high the cost is, and how high the training value is. The assignments need not be stable, or even complete matchings (even if the number of sailors is equal to the number of employers). Preference lists are far from complete, with an individual sailor listing a few of the thousands of possible employers. Multi-objective evolutionary algorithms have been applied to this problem as part of the Navy's genoSAP program. The sparsity of the preference lists leaves a relatively small number of matchings to explore, and mutation and crossover operations are easier to implement when matchings need not be complete. Thus a simple evolutionary algorithm has been developed which can search throughout the solution space in reasonable time (Dasgupta et al. 2008). If, by contrast, we assumed that preference lists were complete, we would have to consider all possible matchings, and this is a vast space for an evolutionary algorithm to search. If we also required that matchings be complete (assuming an equal number of sailors and employers), the mutation and crossover operations of (Dasgupta et al. 2008) would not work.

In this paper, we consider a version of the stable matching problem with multiple objectives, like in the sailor assignment problem. Given n applicants and n employers, each with complete preference lists, we want to find matchings which are stable, respect the aggregate preferences of the applicants and employers, and minimize / maximize any additional objectives. To be precise, let each applicant list his preferences for employers, and let the employers do the same, so that we have two complete sets of preference lists. We represent an assignment of applicants to employers by a permutation σ of $1, \ldots, n$. We say applicant

i gives ranking k to an assignment σ if $\sigma(i)$ is applicant i's kth choice of employer (and employers rank assignments in a similar fashion). We also have some finite number of matrices of quantifiable but incomparable costs and benefits. In deference to the sailor assignment problem, we will consider as an example the two objectives of low monetary cost and high training value. Therefore, let $C = \{c_{i,j}\}$ and $T = \{t_{i,j}\}$ $(1 \leq i, j \leq n)$ represent the cost and training value of assigning applicant i to employer j.

We can measure the aggregate happiness of the applicants and employers by

$$R_a(\sigma) = \sum_{j=1}^{n} \text{ applicant } j\text{'s ranking of } \sigma(j),$$

and

$$R_e(\sigma) = \sum_{i=1}^{n} \text{ employer } i\text{'s ranking of } \sigma^{-1}(i).$$

We can measure the additional objectives with these objective functions:

$$C(\sigma) = \sum_{j=1}^{n} c_{j,\sigma(j)} \qquad \text{and} \qquad T(\sigma) = \sum_{j=1}^{n} t_{j,\sigma(j)}.$$

We want the matching to be stable for efficiency purposes, and we want to minimize $R_a, R_e, C, -T$. It is unlikely that we would find a matching simultaneously minimizing all objective functions, so our goal is to find some number of stable matchings representing Pareto optimal solutions (solutions so that no objective function may be improved without others being worse). In section 2, we explain why the added restriction of stability simplifies this problem, and present an evolutionary algorithm for its solution. In section 3 we refine the algorithm through testing, and in section 4, we compare the results of this algorithm with the results of single objective optimization on a simulated version of the problem. We also consider how closely the Pareto front of a set of stable matchings approximates the Pareto front we would find if we removed the stability restriction. In section 5, we discuss how the problem changes under the assumptions of correlated or dependant preferences and costs.

2. Algorithm concept

The challenge of the sailor assignment problem with complete preference lists is the large search space ($n!$ possible matchings), but the set of stable matchings is a more limited space, and the structure of this set suggests an algorithm for its exploration. Since we are dealing with unknown preference lists, we will assume that applicant and employer preference lists

are chosen uniformly at random from among all $n!^{2n}$ possible preference lists. The random variable $\mathcal{S}_n$ will represent the number of stable matchings in our randomly chosen instance. Gale and Shapley (1962) showed that there is always at least one stable matching. Later, Knuth (1976) discovered a $2n$-dimensional integral formula for $\mathrm{E}[\mathcal{S}_n]$, the expected value of $\mathcal{S}_n$, and conjectured that this formula could be used to find an asymptotic estimate for $\mathrm{E}[\mathcal{S}_n]$. Confirming Knuth's conjecture, Pittel (1989) used the integral formula to show that

$$\mathrm{E}[\mathcal{S}_n] = (1 + o(1))e^{-1} n \ln n, \quad n \to \infty. \tag{1}$$

The second moment of $\mathcal{S}_n$ has also been sharply estimated (Lennon 2009), and found to be

$$\mathrm{E}\left[\mathcal{S}_n^2\right] = (1 + o(1)) \left(\frac{1}{e^2} + \frac{1}{2e^3}\right) n^2 \ln^2 n. \tag{2}$$

Combined with Cantelli's inequality, relations (1), (2) imply that

$$\mathrm{P}\left(\mathcal{S}_n \geq \epsilon \mathrm{E}[\mathcal{S}_n]\right) \geq \frac{(1-\epsilon)^2}{(1-\epsilon)^2 + (2e)^{-1}} > .84$$

for ϵ sufficiently small. Hence the fraction of problem instances with roughly $cn \ln n$ solutions is 0.84, at least. Further, it is known that with high probability, the number of stable matchings is at least $\sqrt{n/\log n}$ (Pittel 1992).

There are instances with much larger numbers of stable matchings (Benjamin et al 1995, Gusfield and Irving 1989, Irving and Leather 1986, Knuth 1976), but from Chebychev's inequality we know that it is unlikely that the number of stable matchings is of an order much larger than $n \log n$. Thus we have evidence to suggest that a randomly chosen instance should have on the order of $n \log n$ stable matchings when n is large. This is a reasonably sized search space for a evolutionary algorithm. The specific design of such an algorithm is suggested by the structure of the set of stable matchings.

We can define a partial order on the set of stable matchings which turns this set into a distributive lattice (Gusfield and Irving 1989). In this partial order, we consider matching M_1 to be above matching M_2 if every applicant prefers his partner in M_1 to his partner in M_2. The Gale-Shapley algorithm finds the top matching in this lattice by requiring applicants to propose to employers, proposing to their most preferred first, and moving down their preference lists when rejected. The employers wait for proposals, holding a proposal from an applicant until a more preferred applicant proposes, at which time they accept the preferred applicant's proposal and reject the applicant to whom they were engaged. The rounds of

proposals continue until a stable matching is found. In order to move down the lattice, we simply break one marriage in the matching, forcing some applicant to propose to the next employer on his list. This sets off a new round of proposals and rejections, which will end in a different stable matching unless an applicant is rejected by every employer left on his list, in which case it ends in failure (Knuth 1990). Since the stable matching lattice is distributive, every pair of matchings has an infimum and supremum. The infimum of two stable matchings M_1, M_2 is the stable matching in which every applicant is assigned the employer he prefers least of his partners in M_1 and M_2. The supremum for M_1, M_2 is the stable matching in which each applicant is assigned the employer he likes best out of M_1, M_2.

In the context of multi-objective optimization, a matching M_1 is dominated by matching M_2 if M_2 is at least as good as M_1 in every objective function and strictly better in at least one objective function. A typical multi-objective evolutionary algorithm develops an initial population, and then evolves that population via mutation and crossover (mating) for some number of generations, producing a set of mutually non-dominating solutions which are an approximation for the Pareto front of the solution space (Kalyanmoy 2001). The structure of the stable marriage lattice suggests breaking a marriage as a mutation operator, and taking the infimum and supremum of two matchings as a mating operator. We might suppose that to explore a large part of the stable matching lattice, the product of the population size and number of generations should be somewhere between $\sqrt{n/\log n}$ and $n \log n$ when n is large. This number may be reduced to save time.

The complexity of the algorithm described above is on the order of the product of population size, number of generations, and the number of operations required to find a stable matching. The latter time is of order at most n^2 times the number of operations required for an applicant to propose to an employer, since each applicant can be rejected at most once by each employer in an attempt to find a stable matching. Thus this algorithm's complexity is

$$O\left(\text{population size} \times \text{number of generations} \times n^2 \times \#\text{ operations required for a proposal}\right).$$

The relevant factor in a proposal is the length of time for the employer to look up the position of the proposing applicant on her preference list, which is $O(n)$, making the complexity of order n^3 multiplied by the product of the number of generations and population size. Note that algorithms already exist for finding various types of stable matchings, for example a stable matching in which the average satisfaction of all persons is maximized (in $O(n^4)$ com-

plexity (Irving et al. 1987), but these algorithms do not take into account factors other than applicant and employer satisfaction. With the general outline of our algorithm suggested by the structure of the stable matching lattice, we now proceed to fill in the details of the algorithm, and to refine the algorithm based on experimentation.

3. Design details and trail runs

From our knowledge of the structure of the stable matching set, we conclude that our algorithm should have the general form shown in figure 1. Note that the initial population

---

1. **Input applicant and employer preferences, cost and training matrices.**
2. **Develop an initial population of stable matchings**
3. **For a specified number of generations, evolve the population by:**
   (a) record non-dominated matchings, reducing this set to a specified size if required
   (b) chose an even number of matchings to pair for mating, and replace each pair with their inf and sup
   (c) break a marriage in any remaining matchings, replacing the broken matching with the new one (if one is found)
4. **Output a list of non-dominated matchings of specified size.**

Figure 1: stable matching evolutionary algorithm

could be developed by breadth or depth first search methods, or by some combination of the two. In breadth first, we start at the top of the lattice with matching M_0, the matching found by Gale-Shapely with the applicants proposing. We then break a marriage to produce a matching M_1, and then repeatedly return to M_0, breaking different marriages to find different stable matchings M_2, M_3, and so on, producing an initial population close to the top of the lattice. Alternatively, we could break a marriage in M_0 to get M_1, then break a marriage in M_1 to get M_2, and so on, marching down a chain towards the infimum of the lattice. This would be a depth first search. We could also compromise by following a chain in depth first fashion for a certain number of steps before returning to the top of the lattice to start again.

We can also choose between randomly picking a marriage to break and choosing one based on a deterministic criteria. For example, we could always break the marriage with the highest cost or lowest training value. If choosing based on a deterministic criteria, like cost for example, we need to worry about what will happen if the highest cost matching contains an applicant with only one possible partner in a stable matching (the average number of such applicants is asymptotic to $\log^2 n$ (Pittel et al. 2007)). Therefore we will allow several attempts to break a marriage should the first one result in no stable matching.

Considering step 3a of figure 1, we must decide how to reduce the size of the non dominated set. This could be done by deleting matchings at random, or by a diversity preserving method such as some type of clustering algorithm, for example by using k-means clustering on objective function values to choose a diverse population of k non-dominated matchings. We must also decide which proportion of the population should be mated as opposed to mutated. To provide evidence for these decisions, we designed an experiment to test the performance of the algorithm with different settings of these parameters.

3.1. Test case generation

We generated ten test cases, each consisting of an instance of preferences, costs and training values for 200 applicants and 200 employers. The preference lists of applicants and employers are independent and uniformly random. In deference to the sailor assignment research of (Dasgupta 2008), we generated training and cost data by the methods used in that research. The training values of assigning applicant i to employer j ($1 \leq i, j \leq n$) are uniform (0.7,1.02) random variables, and the costs of assigning applicant i to employer j were determined by a random variable with the cumulative distribution shown in the appendix in figure A-1 (Garrett 2009).

For each test case, we collected every non-dominated matching found by every run of the algorithm and deleted those matchings which were now dominated. Our first response variable was what percentage of this list was discovered by a run with a given combination of factors, and our second response variable was the lowest cost matching discovered during a run. We consider our first response (non-dom %) as our measure of how well the algorithm covers the Pareto front for a given combination of factors. The second response (minimum cost) measures how effective it is to break a matching based on cost as opposed to breaking it at random. After repeating the experiment on ten test cases for every factor combination,

we averaged the response variables (for each combination of factors) across the ten test cases. These are the results which we will now analyze.

3.2. Experiment results

The left two columns of the table 1 are the non-dominated % and minimum cost averaged over 10 runs, while the next seven columns are the levels of the factors: development type (dev), population size (pop), number of generations (num), percentage mated (mate), number of attempts to break (att), break marriages based on cost (bc), and the list reduction method (shrink). Since our focus is on the main effects of the parameters, we began by inspecting plots of each of the variables against our two response variables.

Table 1: Experiment Results

% non-dom	min cost	dev	pop	num	mate	att	bc	shrink
48%	1286783	depth	20	12	40%	2	true	k-means
24%	1306695	depth	9	13	60%	1	false	k-means
29%	1311912	depth	12	2	30%	4	true	k-means
27%	1301222	depth	14	5	90%	3	false	k-means
47%	1293966	breadth	19	7	20%	2	false	k-means
19%	1310939	breadth	10	6	80%	1	true	k-means
20%	1309570	breadth	8	15	40%	5	false	k-means
36%	1297550	breadth	18	12	90%	4	true	k-means
34%	1305374	breadth	13	8	50%	3	true	k-means
16%	1313552	breadth	5	4	60%	4	false	UAR
35%	1296049	breadth	16	3	50%	5	true	UAR
36%	1298096	breadth	13	14	70%	3	false	UAR
34%	1296587	breadth	11	11	10%	3	true	UAR
19%	1303906	depth	6	9	80%	4	true	UAR
44%	1287593	depth	15	10	30%	5	false	UAR
32%	1299354	depth	17	1	70%	2	true	UAR
22%	1305470	depth	7	5	20%	2	false	UAR

In these plots, there was no clearly discernable relationship between the responses and any variable other than population size, which showed a clear linear relation. The correlation between the non-dom % and population size ($r = 0.89$) much outweighs the correlation between non-dom % and the number of generations ($r = 0.17$). This size of this difference is somewhat surprising, as we expected the number of generations to have an important influence on the diversity of the non-dominated set. One possibility is that the typical stable matching lattice is short and wide, with relatively few steps in a chain between the applicant

and employer optimal matchings. In previous tests with numbers of generations above 20, we found that a large proportion of the last generation was the employer optimal matching. In one trial, every member of the 21st generation was the employer optimal matching. Upon reaching such a stage, further evolution brings no benefit, and the number of generations would not matter. It could be that depth first search tends to find the employer optimal matching and other matchings close to it from which initial population exploration is limited. Regardless of the reason, it appears that there is more benefit in a large population than in a large number of generations.

We also observe that there was no improvement in the minimum cost variable when we break marriages based on cost. In earlier trials we did with 50 applicants, we did see a small improvement, but it seems to vanish with 200 applicants. This seems reasonable, since one would expect that as the number of applicants increases, the rounds of proposals resulting from breaking a marriage will involve greater numbers of people changing partners, and breaking marriages based on cost only helps for the first marriage broken. Thus breaking marriages based on cost will only be valuable when the population size is small, or when there are some extremely high cost values. In fact, there is a nearly linear relationship between minimum cost and non-dom % ($r = -.80$), suggesting that the best way to get low cost is by finding a large non-dominated set, which one does by choosing a large population size.

Our recommendation based on the tests conducted above would be to use a population size as large as possible given running time considerations, with some low number of generations and depth first search, deleting matchings from the non-dominated list uniformly at random. Use a local improvement operator for breaking matchings only when the distribution of the variable being improved has some extremely high values. After such a run, one can see if the last generation is largely composed of the employer optimal matching, or is still diverse, and use this as a criteria upon which to decide if more generations of evolution are desirable.

4. Comparison with single variable optimization

We chose to use 200 applicants and employers, and to generate one test case by the same method used in the previous section. We picked a population size of 20, a non-dominated set size of 40, and evolved the population for 5 generations, developing the initial population via depth first search. We chose to break marriages randomly during mutation, and allowed

3 attempts to break a marriage. We mated 50% of the population.

The matchings found by our algorithm appear to good compromises between applicant and employer preferences, as one might expect from the stability requirement. As one might also expect, the single variable optimization program can significantly outperform us with regard either to minimizing cost or to maximizing training value. It can also achieve better results with regard either to minimizing applicant rank or employer rank, but here it's myopic focus does not bring as much of an advantage.

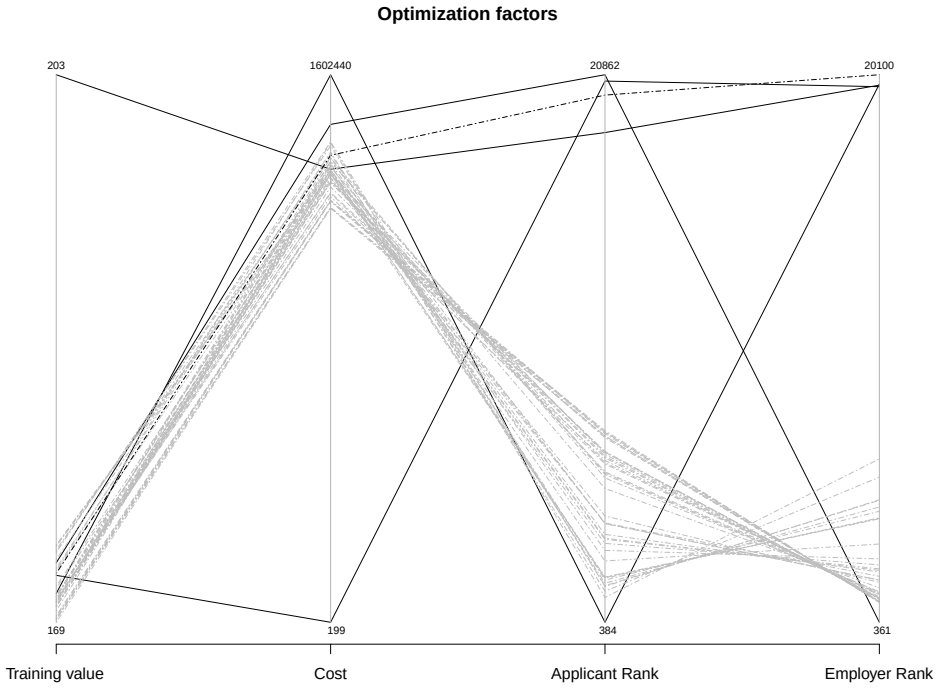

Figure 2: Single and multi-objective optimization

As another method of comparison, we might consider what the values of the rankings, the cost and training value "should " be for some randomly chosen matching. The expected value of the applicant ranking of this matching is the number of applicants multiplied by the average applicant ranking: $200 * 100.5 = 20,100$. Likewise, the expected value is $20,100$ for the employer ranking, $200 * .86 = 172$ for the training value, and $200 * 6,833 = 1,366,600$ for the cost (the average cost is $\$6,833$ (Garrett 2009)). Indeed, the values of those variables not optimized by optmatch are close to these values, as are the cost and training values for the stable matchings. This occurs because preference lists, costs, and training values were

generated independently, with all cost and training variables independent and identically distributed (i.i.d), so whenever a matching is chosen without regard to an objective, we expect the other objectives to be close to their mean. The advantage of using the stable matching evolutionary algorithm is that many different matchings are found, so one can hope to choose one with a particularly low cost and (or) high training value. As a note of caution, the stable matchings may have substantial overlap, so the spread of the cost and training values will probably not be as large as if we were to choose an equal number of matchings uniformly at random.

The assumption that preference lists, cost, and training value are all independent, with preferences chosen uniformly randomly and cost and training value i.i.d., is unlikely to hold true. The difficulty with performing tests without these assumptions is that there are many ways in which the variables in question could be related. Any size subset of applicants could have any degree of correlation between their preferences, and the same is true for employers. Costs and training values could be correlated among themselves, could be dependent on one another, or could be correlated with preferences. In the sailor assignment problem, for example, one might expect that employers are likely to want a sailor when training value is high, as the high training value could indicate that the sailor is appropriate for the position in question. We would expect sailors to be broken down into subsets depending on their career path, with similarity between the preferences of sailors in each set, and a similar situation might exist for employers. With such vague knowledge of the exact structure of the dependencies, we are best served by testing the algorithm under the assumptions of independence, and then providing intuition as to how dependency or similarity of preferences might change the performance of the algorithm, as we will do in the next section.

5. Performance under differing assumptions

The strength of the stable matching evolutionary algorithm is in finding a population of matchings which are good compromises between applicant and employer preferences. To the extent that other variables are correlated with or dependent on these preferences, the stability restriction will be a proxy for these other variables. If, in the sailor assignment problem, good employer rankings imply high training value, for example, our algorithm will be selecting for high training value. If sailors want jobs where there is a high cost to the assignment, we will also be selecting for high cost. The more interesting question is how

the stable matching algorithm performs when we change the assumption that applicant and employer preferences are independent and uniformly random.

If the applicant and employer preferences are independent (or nearly so) of the other objectives, then we expect that a large number of stable matchings will provide more possible solutions, and thus better opportunity to find good values for the other objectives. So we turn to the question of how changing our assumptions changes the number of stable matchings. First we consider the extreme case of similar preference lists.

Lemma 5.1 *When every applicant has an identical preference list and every employer has an identical preference list, there is exactly one stable matching.*

PROOF. Let the applicants propose to their partners in the order in which they are preferred by the employers, so that the applicant listed kth by all employers chooses his partner kth. Since each applicant has the same preference list, he will be rejected by his first $k-1$ choices, who all hold better proposals, and accepted by his kth choice, who has no partner. The applicant who is listed kth on every employer's preference list will be married to the employer who is listed kth on every applicant's preference list. But this is the same result we would get if employers proposed, meaning the applicant and employer optimal matchings are the same, and there is only one stable matching. □

Identical preference lists are the extreme of similarity, but it is less certain what the extreme of different preference lists is. A Latin square is one candidate. A Latin square is a $n \times n$ matrix where every number $1 \ldots n$ appears exactly once in every row and exactly once in every column. When a certain Latin square construction is used to create preference lists, there are exponentially large numbers of stable matchings (Benjamin 1995). We hoped that there might be a simple relationship between the number of stable matchings and the similarity of the preference lists, with greater similarity implying fewer stable matchings, and greater difference corresponding to more stable matchings. No simple relationship of this kind holds, however, as we now show.

Lemma 5.2 *Suppose for $1 \leq i \leq n$, applicant i ranks employer i within his first i choices, and only prefers some subset of employers $1, \ldots, i-1$ to employer i. Suppose also that employer i ranks applicant i within her first i choices, and only prefers some subset of applicants $1, \ldots, i-1$ to applicant i. Then there is exactly one stable matching.*

The proof is similar to that of lemma 5.1.

Evidently we can decrease the similarity of the preference lists quite a bit without increasing the number of stable matchings, at least under some conditions. But while we we can significantly alter the similarity of applicant and employer preferences without increasing the number of stable matchings, we do not believe that we can change the dependance between sailor and employer preferences without changing the number of stable matchings. We think it reasonable to suppose that the expected number of stable matchings will generally increase as the difference in ranking between the applicant and employer optimal stable matchings increases. In other words, greater mutual attraction between applicants and employers might mean fewer stable matchings. If true, then one could, for a given preference list, find the applicant and employer optimal matchings by having applicants and employers propose, and then judge how many stable matchings one might expect to find based on the difference in the ranks between these two matchings.

The effect of similarity of preferences on the Pareto front is even less certain. We can say something about its shape when preference lists are identical, but not when they differ.

Proposition 5.3 *When every applicant has an identical preference list and every employer has an identical preference list, every matching is Pareto optimal with respect to applicant and employer ranks, and has the exact same ranks.*

This proposition, in combination with lemma 5.1, suggests that a stable matching evolutionary algorithm may produce poor results when preferences are extremely similar.

6. Conclusions

To closely approximate the Pareto front of a multi-objective optimization problem, we search for a set of non-dominated matchings which is large, diverse, and close to the Pareto front. The last of these three criteria is hard to measure, since we cannot find the actual Pareto front, but based on our comparison with single objective optimization, we have reason to suspect that stable matchings are often close to Pareto optimal with respect to applicant and employer preference (not considering other objectives). The stability criteria may limit the diversity of the population, and likely misses matchings which are extreme with respect to cost or training value, suggesting that this optimization method is of best use when applicant and employer preference are the primary considerations, and the other objectives are

regarded as desirable, but secondary. In this case, this algorithm can produce good results, but they will vary with the preference lists. Specifically, we expect a larger and more diverse non-dominated set when there are more stable matchings to choose from, a situation which seems more likely to occur when there is not strong mutual attraction between subsets of applicants and employers.

Appendix: Cost CDF and Optimization Results

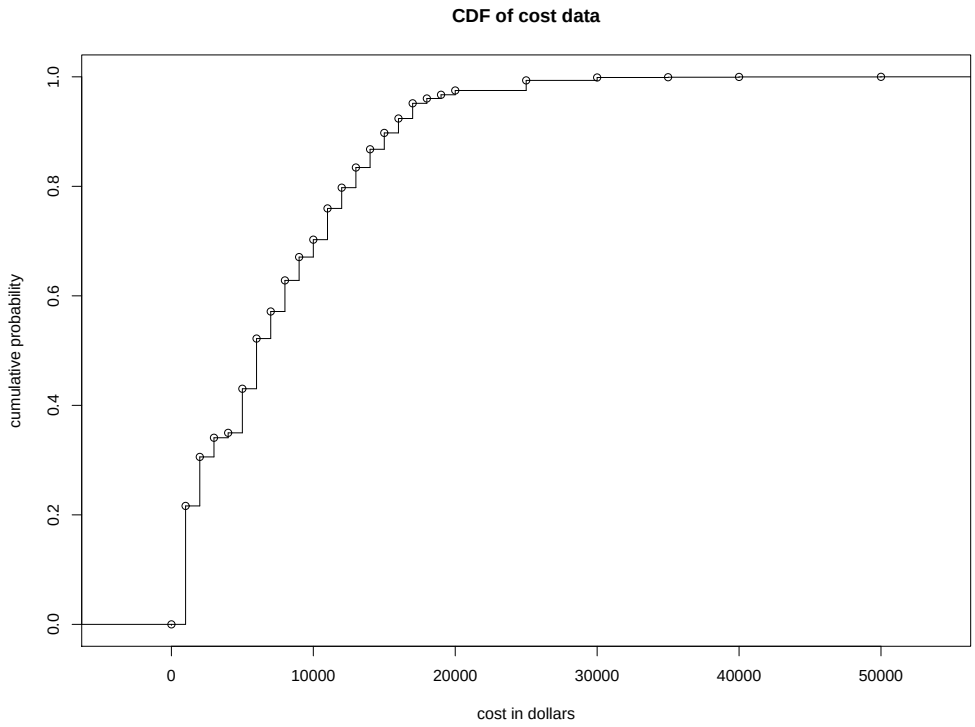

Figure A-1: cost variable CDF

Table A-1: Single variable optimization results

training value	cost	applicant ranking	employer ranking
171	1602440	384	19762
173	1457100	20862	361
172	199	20624	19669
203	1325842	18698	19724

Table A-2: Stable matching GA results

training value	cost	sailor ranking	employer ranking
169.3	1234875	7486	1090
170.0	1315137	3529	2234
169.9	1312559	3487	2286
173.7	1323901	2033	4127
173.8	1312141	2059	4106
173.8	1321340	2078	4092
172.9	1370741	1962	4362
169.1	1213448	7496	1079
169.1	1232020	7286	1120
169.9	1301991	4096	2017
172.2	1402837	1311	6253
171.2	1316561	6010	1433
171.0	1293581	6353	1357
170.1	1342196	4071	1883
169.1	1214059	7551	1073
170.4	1255709	6305	1341
173.3	1406388	1726	4777
170.3	1287351	6589	1229
168.9	1211204	7351	1103
170.3	1286740	6534	1235
169.6	1323159	3671	2173
169.3	1234264	7431	1096
170.2	1336309	3073	2646
172.3	1376202	1543	5595
170.2	1258080	6775	1175
170.0	1323587	4080	1879
170.2	1273841	6745	1222
170.5	1289595	6734	1205
173.4	1394628	1752	4756
169.2	1232631	7341	1114
170.6	1310866	3334	2429
172.1	1336209	2677	3182
169.9	1299974	4360	1798
170.3	1245628	6437	1278
170.6	1349021	5968	1393
173.5	1360748	1845	4519
170.2	1309935	6664	1233
170.2	1331548	5388	1605
170.5	1348410	5913	1399
170.6	1358491	5781	1462

Acknowledgments

Thanks to Janet Spoonamore, Dipankar Dasgupta, Deon Garrett, James Simien, Bill Pullyblank, Darryl Ahner and Richard Deckro for their suggestions and support. This project was funded in part by the Army Research Office.

References

Abdulkadiroglu, A., P. Parag, A. Roth. 2005. The New York City High School Match, *American Economic Review, Papers and Proceedings* **95** 364–367.

Benjamin A., C. Converse, H. Krieger. 1995 How do I marry thee? Let me count the ways. *Discrete Applied Mathematics*, **59** 285–292.

Cioppa T., T. Lucas. 2007 Efficient Nearly Orthogonal and Space-Filling Latin Hypercubes. *Technometrics*, **40** 45–55.

Dasgupta D., G. Hernandez, D. Garrett, P. Vejandla, A. Kaushal, R. Yerneni, J. Simien. 2008. A comparison of multiobjective evolutionary algorithms with informed initialization and kuhn-munkres algorithm for the sailor assignment problem. *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation* 2129–2134.

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

Garrett D. 2009 personal communication.

Gusfield D., R. Irving. 1989. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA.

Hansen B., M. Fredrickson. 2011. http://cran.r-project.org/web/packages/optmatch/index.html

Irving R., P. Leather. 1986. The Complexity of Counting Stable Marriages. *SIAM J. Computing* **15** 655–667.

Irving R., P. Leather, D. Gusfield. 1987. An efficient algorithm for the "optimal" stable marriage *Journal of the Association for Computing Machinery* **34** 532–543.

Kalyanmoy D. 2001. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, New York, NY.

Knuth D. 1976. Stable marriage and its relation to other combinatorial problems: an introduction to the mathematical analysis of algorithms. American Mathematical Society, Providence R.I.

Knuth D., R. Motwani, B. Pittel. 1990. Stable Husbands. *Random Structures and Algorithms* **1** 1–14.

Lennon C.,B. Pittel. 2009. On the Likely Number of Solutions for the Stable Marriage Problem, *Combinatorics, Probability, and Computing*, **18** 371–421.

Pittel B. 1989. The average number of stable matchings. *SIAM J. Disc. Math* **2** 520–549.

Pittel B. 1992. On likely solutions of a stable marriage problem. *Annals of Applied Probability* **2** 358–401.

Pittel B., L. Shepp, and E. Veklerov. 2007. On the number of fixed pairs in a random instance of the stable marriage problem *SIAM Journal of Discrete Math* **21** 947–958.