EVOLVING TINDER

30592 - CODING PROJECT

Giacomo Cirò
Bocconi University

June 5, 2024

ABSTRACT

The Deferred Acceptance Algorithm (DAA) is an iterative method to find a stable matching between two sets of individuals with subjective preferences, referred to as a 1-to-1 matching scenario. However, the DAA matching turns out to be the worst possible matching for one set of agents, and the best possible for the other. In this project, I mitigate the extreme polarity of the DAA using an evolutionary approach to find a stable matching, given two sets of individuals and their preferences.

Keywords 1-to-1 Matching · Evolutionary Algorithms

1 Introduction

Economics studies how individuals and societies allocate scarce resources to satisfy their needs. Usually, this involves monetary transactions and markets. However, there exist some interesting environments where money does not play a role. For example, take two sets of agents where each individual rank the others according to some preference relation. It can be of interest to study possible ways of pairing individuals from the two sets together, such that some axiomatic properties hold. This scenario is typically referred to as 1-to-1 matching problem. Traditional Economics, and in particular the branch of Mechanism Design, proposes the Deferred Acceptance Algorithm (DAA) as a useful iterative algorithm to find a matching that satisfies equilibrium conditions, such as stability and rationality. However, it also satisfies another property. Namely, the yielded matching is the worst for individuals in one of the two sets, and the best for the others, among all possible stable matchings. The DAA is therefore an extremely polarized method for finding stable matchings, which proposes drastic results which either maximize or minize the agents' welfare, without exploring in-between solutions.

Evolutionary Algorithms (EAs) are a group of optimization techniques inspired by the principles of natural evolution, such as selection, mutation, recombination, and inheritance. These algorithms are used to solve optimization and search problems, and are especially useful for non-convex optimization or multi-objective optimization (MOO). They work by iteratively improving a population of candidate solutions with respect to a given target measure, called fitness.

In this report, I explore how evolution can be applied to the 1-to-1 matching problem to tackle the shortcomings of the more traditional Deferred Acceptance Algorithm.

2 Background

2.1 1-to-1 Matching Problem

Marriage Market Consider the sets W, M of n agents and, for simplicity, refer to them as men and women:

$$W = \{w_1, ..., w_n\}$$
$$M = \{m_1, ..., m_n\}$$

For each $w_i \in W$, a preference relation $\succ_i$ is given. That is, $\succ_i$: $M \to M$ is a complete and transitive binary relation such that:

$$m_1 \succ_i m_2 \iff w_i \text{ prefers } m_1 \text{ to } m_2, \forall m_1, m_2 \in M$$

Analogous definition for each $m_j \in M$.

Matching A function $\mu : W \cup M \to W \cup M$ such that:

- $\mu(w) \in M, \forall w \in W$
- $\mu(m) \in W, \forall m \in M$
- $\mu(m) = w \iff \mu(w) = m$

is called a matching. Loosely speaking, men and women are matched with each other in a unique fashion. There cannot be a man matched with two different women, or vice versa. Hence, a matching can be represented by a list of pairs (m, w). Actually, matchings and preferences also include the possibility for an agent to be unmatched. However, for simplicity, I decide to rule out this possibility.

Stable Matching A matching is said to be stable if there are no blocking pairs. A pair (m, w) is said to be blocking if both m, w prefer to be matched with each other compared to whom they're currently matched to. Hence, a matching is stable if the following holds:

$$\nexists m_i \in M, \nexists w_j \in W : w_j \succ_i \mu(m_i), m_i \succ_j \mu(w_j)$$

Social Value Given a preference relation, the score man i assigns to woman j is how high woman j is in man i's preferences, namely 1 for the least preferred woman and n for the most preferred. A quality of a single pair in a matching is the sum of the scores each individual in the pair assigns to the other. The social value of a matching is the sum of the qualities of each pair in the matching. Social value can be restricted to men (women) by only considering the scores the men (women) assign to the women (men) they are matched with. A matching is said to be best for women (men) if the women's (men's) social value is the highest it can be.

Deferred Acceptance Algorithm The Deferred Acceptance Algorithm (DAA) proposed by Gale & Shapley in 1962 is an iterative algorithm to find a stable matching. It works by either men-proposing or women-proposing. Consider the former. At each iteration, every man proposes to his most preferred woman who did not reject him yet. Each woman rejects all the offers she received, but the most preferred one. The algorithm stops when no offer is rejected. The yielded pairs form a matching and the following can be proved:

- the matching obtained via either form of the DAA is stable;
- the matching obtained via men-proposing DAA is the best possible stable matching for men and the worst possible stable matching for women;
- the matching obtained via women-proposing DAA is the best possible stable matching for women and the worst possible stable matching for men.

2.2 Research Question

The shortcoming of this method and what I try to address using EAs can be clearly understood from the previously stated results: the DAA does yield a stable matching, but it does so too extremely, making one group as well off as possible and the other as worse off as possible. Can stable matchings that lie in between be found to accommodate the preferences of both men and women?

3 Methodology

In order to carry out evolutionary optimization, I first define an efficient way of representing a candidate solution and define the multi-objective fitness function. Then, I analyze different flavours of selection, mutation and recombination techniques to find the best ones in the Algorithm Design (AD) stage. Then, I optimize the parameters in the Parameters Optimization (PO) stage. Finally, I compare the evolutionary solution with the DAA stable matching.

During the algorithm design and parameter optimization stage, I stick with low dimensionality ($n = 10$), population size and runtime to speed up computations.

All details regarding algorithm structure can be found in table 1.

Representation	Permutations
Population Management	AD
Recombination	AD
Recombination probability	PO
Mutation	Swap
Mutation probability	PO
Parent selection	AD
Survival selection	AD
Population size (μ)	100
Number of Offspring (λ)	PO
Initialisation	Random
Termination condition	100 generations

Table 1: algorithm details and the stages at which different components are optimized. The pre-defined values are either kept as default or increased later to aid dealing with bigger problem size.

3.1 Permutation Representation

Given two sets of n individuals, a candidate solution, i.e., a matching, is represented as a permutation of a n-dim vector $x \in \mathbb{N}^n$ of indexes, such that each x_i encodes one pair $(i, x_i) = (m, w)$ in the matching. The dimension indicates a man, the entry the woman he is matched with. For example, given:

$$M = \{m_0, m_1, m_2\}$$
$$W = \{w_0, w_1, w_2\}$$

the vector:

$$x = [0, 2, 1]$$

encodes the matching:

$$\mu = \{(m_0, w_0), (m_2, w_1), (m_1, w_2)\}$$

This way of representing a solution ensures all men are matched to at most one woman (no duplicate indexes), and the matching properties hold.

3.2 Multi-Objective Fitness Function

My goal is to address the extreme polarity of the DAA solution. Consequently, the optimization process should not just minimize the number of blocking pairs to find a stable matching, but also account for each group's social value trying to maximize it. I employ a multi-objective fitness function $f : \Omega_n \to \mathbb{R}^3$. Where Ω_n is the set of all possible permutations of a n-dim vector of indexes and $f(x) = [SV_m, SV_w, BP]$ such that:

- SV_m, measures the men social value, to be maximized;
- SV_w, measures the women social value, to be maximized;
- BP, measures the number of blocking pairs, to be minimized.

I decide not to include a zero-blocking pairs constraint, and instead add the number of blocking pairs as an objective to be minimized to allow the algorithm broader exploration. Indeed, stability is a very strict requirement, and it might be that only few stable matchings are possible. Ruling out all non-stable matchings makes exploration almost random, without relying on exploitation at all, and might also cause the model to get stuck as soon as it finds a possible solution. Instead, optimizing the BP objective together with the SV's, ensures more efficient convergence.

Notice that all three objectives are bounded above by n^2. In fact, the number of possible pairs (an upper bound for the number of blocking ones) given two sets of size n is $n \cdot (n-1)$, not allowing for self-matchings. Moreover, in the best case scenario all men and women are matched with their top choice, assuming preferences accordingly. Assigning a score of n to the top choice yields $maxSV_w = maxSV_m = n^2$.

Pareto Dominance Due to multi-dimensional fitness, simple arithmetic comparisons do not make sense. Instead, a useful notion is that of Pareto Dominance. A solution x Pareto dominates another solution y if the corresponding fitness vectors $f(x), f(y)$ satisfy:

- $f(x)_i \geq f(y)_i, \forall i \in \{1, ..., k\}$
- $f(x)_j > f(y)_j, \exists j \in \{1, ..., k\}$

That is, x is at least as good as y in all k objectives and strictly better in at least one. $f(x)_i, f(y)_i$ are multiplied by -1 when comparing objectives to be minimized.

A set of solutions can be divided into Pareto Fronts P_k such that each $y \in P_k$ is dominated only by $x \in P_{k-1}$. The Pareto front P_1, or simply Pareto front, is thus the set of Pareto undominated solutions, i.e., Pareto optimal.

3.3 Algorithm Design

In order to design an evolutionary algorithm, one has to specify the evolution techniques to be used. During this stage, I set mutation to index swap, mutation probability to 0.9 and recombination probability to 0.1. The former is higher to facilitate the minimization of blocking pairs. If there is a blocking pairs, randomly swapping indexes might help to targeted removal, compared to more convoluted crossover strategies which modify the solution more broadly. For the same reason, I decide not to explore other mutation strategies and focus on recombination and selection. I also fix $\lambda = 3 \times pop\_size$ during the AD stage.

I test the following techniques in all possible combinations:

- *Population Management*
  - **S** $\rightarrow$ Simple EA, each individual is combined into a pair of parents. Offspring are generated to replace all parents.;
  - **MCL** $\rightarrow(\mu, \lambda)$-EA. At each generation, the μ fittest are chosen among λ offspring produced by random parents;
  - **MPL** $\rightarrow(\mu + \lambda)$-EA. At each generation, the μ fittest are chosen among a pool of λ offspring generated by random parents plus the old individuals.
- *Crossover*
  - **OX** $\rightarrow$ Ordered Crossover. Randomly select a subsequence from one parent and copy into the offspring. The remaining positions are filled by copying the elements from the second parent;
  - **PMX** $\rightarrow$ Partially Mapped Crossover. Randomly select a subsequence from one parent and copy into the offspring. The remaining positions are filled a map based on how the indexes in the first parent where copied to the offspring, and the original position of this indexes in the second parent.
- *Survival Selection*
  - **NSGA2** $\rightarrow$ Non-dominated Sorting Genetic Algorithm II. It performs selection using Pareto fronts and crowding distance. Lower-ranked fronts and with higher crowding distances are selected to optimize objective while preserving diversity;
  - **SPEA2** $\rightarrow$ Strength Pareto Evolutionary Algorithm II. Each individual is assigned a score based on the number of solutions it dominates and the solutions by which it is dominated. A k-nearest neighbor approach is used to estimate density. Highest score and lowest density solution are selected to optimize objective while preserving diversity.
  - **LEX** $\rightarrow$ Lexicase Selection. It selects individuals based on individual objectives, one at a time in a random order.

For *Parent Selection*, I stick with what the DEAP implementation of the different population management methods used, indicated above.

3.4 Parameters Optimization

After choosing the optimal design and evolution methods, I conduct parameter optimization. The following values are tested:

- *Recombination Probability* $\in [0.1, 0.5, 0.9]$
- *Number of Offspring* (λ) $\in [1.5, 3, 5]$, as ratio to population size μ

The *Mutation Probability* is required to be $1-$*Recombination Probability*.

3.5 Problem Size

One can argue the performance on small n is due to the algorithm simply testing all possible solutions. However, all possible permutations of a n-dim vector are $n!$. Assuming all generated offspring are different from each other, which is an upper bound, it would still take $10!/300 = 3,628,800/300 = 12,096$ generations to test all possible permutations of a 10-dim solution using a population of 300 individuals.

I expect the performance of my model to be generalizable to arbitrary dimensions, even though broader search might be required, for example extending to bigger population size and higher number of generations. Indeed, I test the evolutionary approach using $n = 100$, and the original population of 100 individuals, but also a population 5 times bigger. I let evolution carry on for 500 generations.

4 Results

Throughout this section, I refer to the evolution process carried out until termination condition using one set of specified parameters as one "run". I refer to the plot of the maximum (minimum) value across all individuals in the population of the objective to be maximized (minimized) as simply the plot of that objective.

In order to achieve better generalization, the preferences used for computing the objectives are randomly initialized at each run.

4.1 Algorithm Design

As one could expect, the Simple EA is the worst performing algorithm by orders of magnitude, as shown in the top graph in figure 1. Thus, I remove it from the pool and conduct comparison of MCL and MPL in more detail, as shown in the other graphs of figure 1. The $(\mu + \lambda)$ population management technique performs better across all three objectives.

When considering the SV_m and SV_w objectives, the two crossover methods investigated are quite comparable. However, PMX is faster on the BP objective, performing consistently better across all generations.

Lastly, I compare selection strategies. Surprisingly, the more stochastic LEX selection outperforms the more sophisticated SPEA2 and NSGA2 on the SV_w and BP objectives, achieving both faster convergence and higher fitness. However, in the SV_m objective, it converges too fast and plateau without reaching the all-time high the NSGA2 is able to achieve.

In conclusion, MPL-PMX-LEX is my best-of-the-breed architecture of choice. MPL's performance is significantly better across all three objectives. PMX is strictly better only on the BP task, and at least as good as OX on the other objectives. LEX, even though worse on one objective, is still the best choice due to its performance on the remaining two objectives and its relative simplicity.

Refer to Appendix 7 figure 6 for the plot of the Pareto fronts obtained from all runs in the AD stage.

4.2 Parameters Optimization

The performance of the different combinations of parameters tested during the PO stage is reported in figure 2. Each run is characterized by a different pair $(\lambda, cxpb)$, and recall $mutpb = 1 - cxpb$. When looking at each run separately, there is not a significant difference between the combinations, except for $(1.5, 0.9)$, which is the worse among all.

When grouped by $cxpb$, 0.9 turns out to be the worst possible choice, confirming my initial theory that mutation is what helps the most in minimizing the BP objective, and the drop in performance extends to the SV's as well. As further confirmation of my theory, $cxpb = 0.1$, which implies $mutpb = 0.9$, is the fastest in minimizing the BP objective. However, I decide to opt for $cxpb = mutpb = 0.5$, because even though it's slightly slower to converge on BP, it enables the model to reach highest SV's.

Relative to the number of offspring, $\lambda = 3$ seems to be the best option. Indeed, it significantly outperforms both $\lambda = 1.5$ and $\lambda = 5$ on the SV_m objective. Even though slightly below $\lambda = 1.5$ on the SV_w, it improves on the BP, where the latter fails to find a stable matching.

The final algorithm architecture is reported in table 2. Refer to Appendix 7 figure 5 for the plot of the Pareto fronts obtained from all runs in the PO stage.

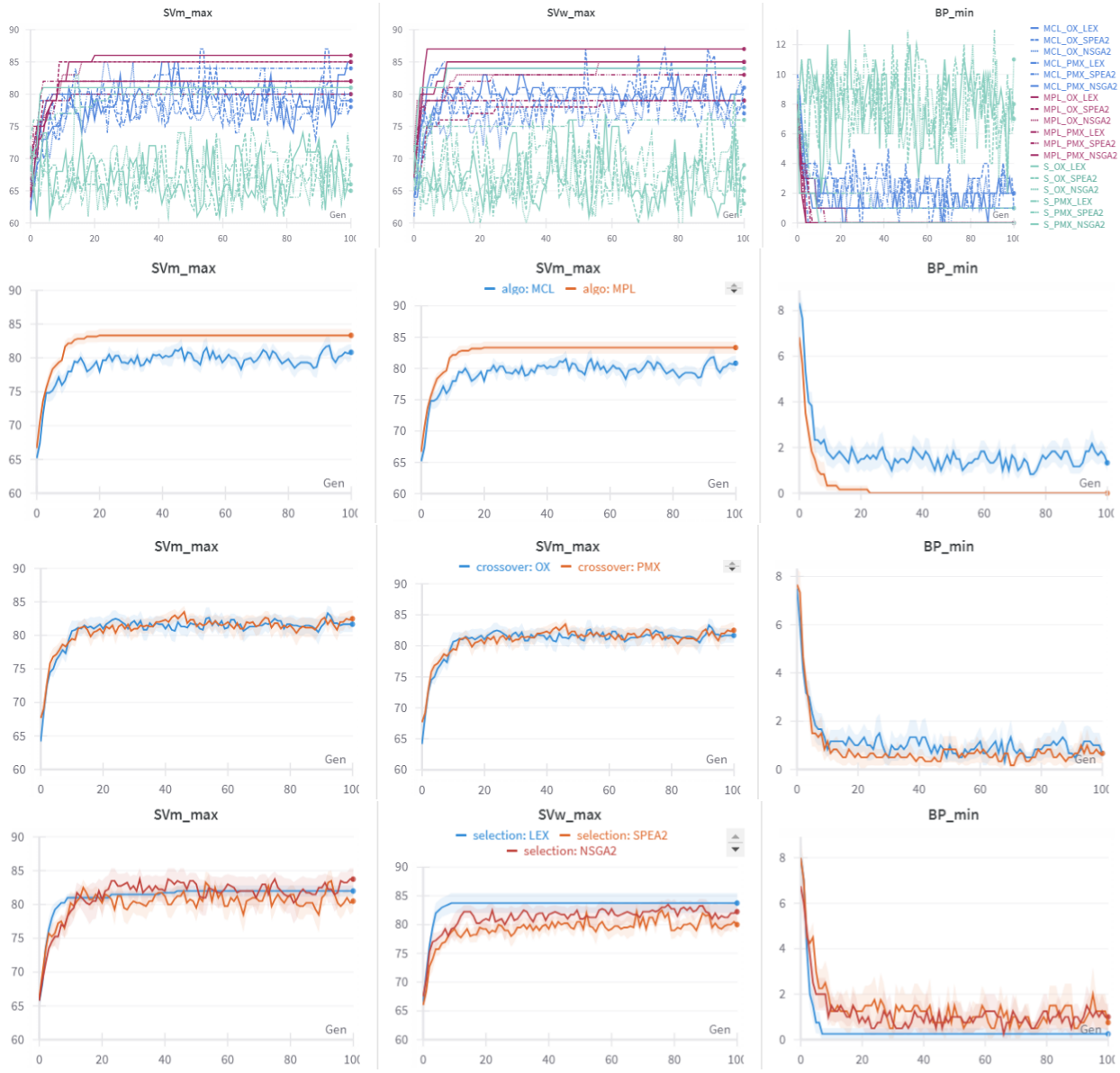

Figure 1: the codes in the legend refer to different techniques tested. Top: plot of the three objectives for all runs. Middle-up: plot of the three objectives mean grouped by population management method. Middle-down: plot of the three objectives mean grouped by crossover method. Bottom: plot of the three objectives mean grouped by selection strategy. Standard errors are shown.

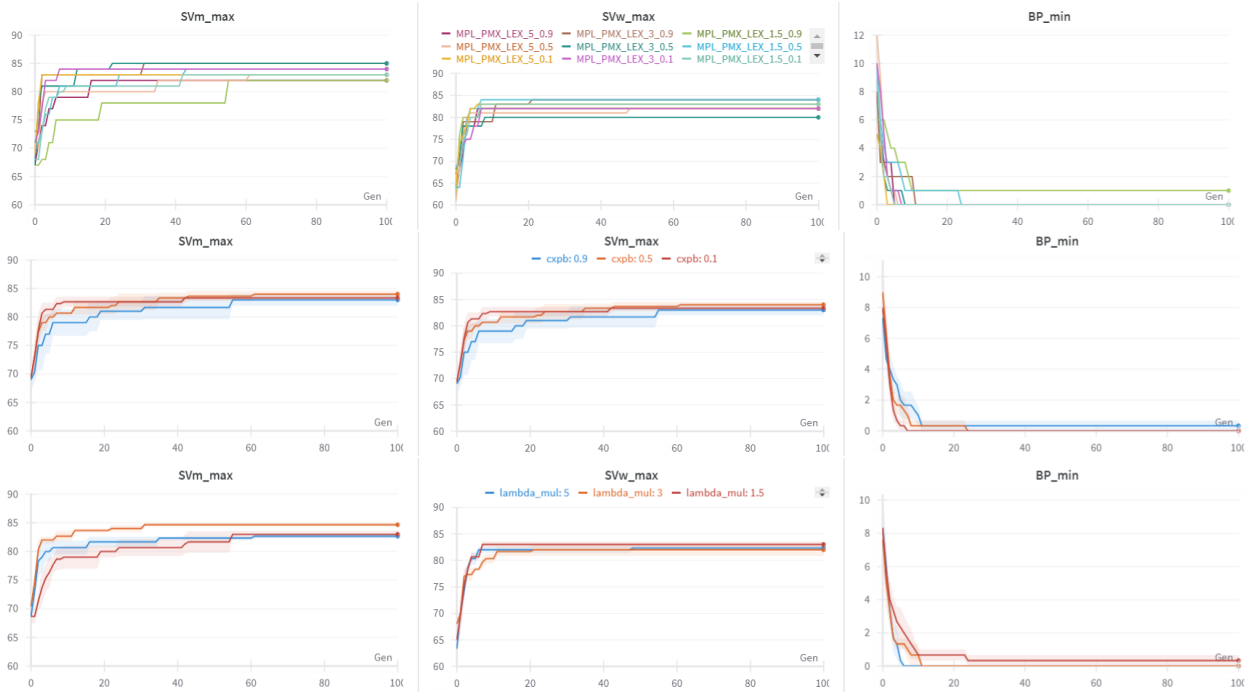

Figure 2: the numbers in the legend refer to lambda and crossover probability tested, respectively. Top: plot of the three objectives for all runs. Middle: plot of the three objectives mean grouped by crossover probabilities. Bottom: plot of the three objectives mean grouped by lambda. Standard errors are shown.

Representation	Permutations
Population Management	MPL
Recombination	PMX
Recombination probability	50%
Mutation	Swap
Mutation probability	50%
Parent selection	Random
Survival selection	LEX
Population size (μ)	100
Number of Offspring (λ)	3×
Initialisation	Random
Termination condition	100 generations

Table 2: final algorithm as a result of the AD and PO stages.

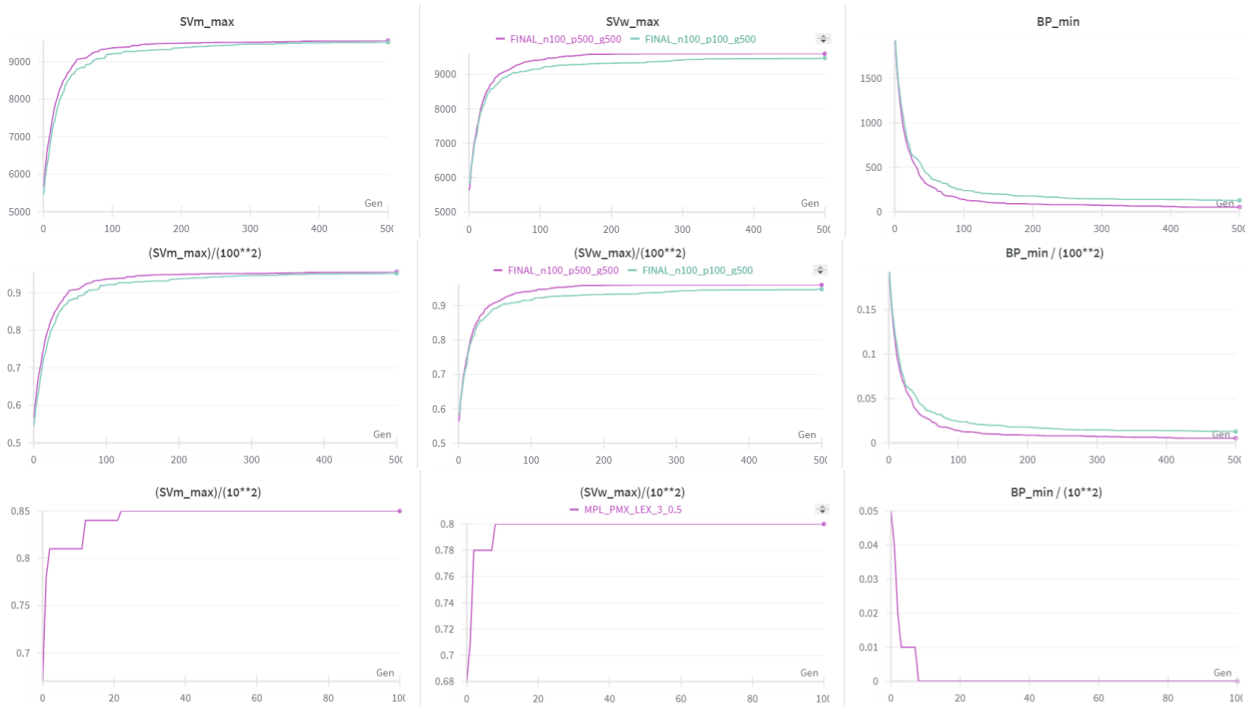

Figure 3: the codes n, p, g in the legend refer to problem size, population size and number of generations respectively. Top: unscaled plot of the three objectives for $n = 100$ problem size. Middle: scaled plot of the three objectives for $n = 100$ problem size. Bottom: scaled plot of the three objectives for $n = 10$ problem size.

4.3 Problem Size

As expected, figure 3 shows that when dealing with greater size problems the algorithm benefits from increased population and number of generations.

For better comparison, I scale all the objectives by their upper bound. On the SV task, both problem sizes are well handled by the algorithm, reaching as high as 90% of the upper bound. However, the evolutionary approach is not able to completely eliminate blocking pairs and find a stable matching for $n = 100$, even though in relative terms it's quite comparable to the performance on lower dimensions: both manage to decrease the number of blocking pairs to less than 1% of the total number of possible pairs.

4.4 Pareto Front

By plotting the Pareto front, it can be seen that the EA is able to find the men-proposing DAA stable matching, together with many other stable matchings, the purple dots in figure 4. Moreover, it yields a continuum of Pareto optimal solutions, with at most 5 blocking pairs, and 2 on average.

Unfortunately, when scaling to $n = 100$ the DAA solution is not found, and wee see the Pareto front distributed around three main clusters, suggesting that perhaps the algorithm got stuck in local minima.

5 Conclusions

The 1-to-1 matching problem is an interesting, yet not trivial optimization problem. Traditionally, Mechanism Design solves it from a stability perspective alone. Conversely, I decide to satisfy other requirements as well, striking a balance between stability and welfare, minimizng blocking pairs while also considering the trade-off with individual benefits.

Multi-objective EAs can be effectively applied to such aim and I conducted a careful design and optimization of a $(\mu + \lambda)$-EA which yielded a satisfactory performance. Indeed, the optimized model was able to find both the DAA solution and other stable matchings at $n = 10$ problem size. Additionally, the resulting Pareto front contains non-stable

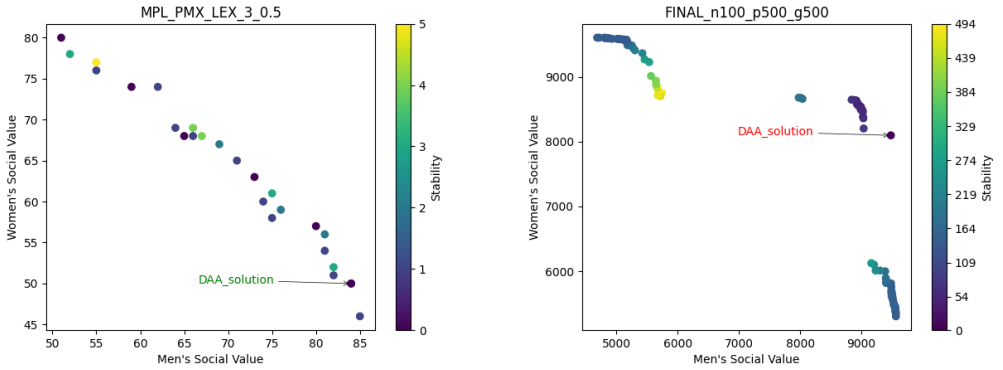

Figure 4: plot of the Pareto front. Number of blocking pairs is color-coded. The men-proposing DAA stable matching is highlighted in green if the evolutionary algorithm was able to find, red otherwise. Left: $n = 10$, $pop\_size = 100$, $n\_gen = 100$; Right: $n = 100$, $pop\_size = 500$, $n\_gen = 500$

yet-undominated solutions, which can be a valid alternative if one decides to relax the no blocking pairs requirement in favor of improving agents' welfare.

Lastly, I increase the problem size and find that my model generalizes quite well. Even though not able to find the DAA solution at $n = 100$, the evolutionary procedure yields a rich Pareto front, with valid alternatives and close-to-optimal solutions.

6 Future Work

It could be interesting to explore a fourth fitness objective, which measures a collective social value, such as $\alpha SV_m + (1-\alpha)SV_w$. This forces the model to look for stable matchings while allowing the user to tweak the relative importance of the two groups of agents.

Moreover, the PO stage suggested that a balance between mutation and recombination is useful in this framework as well. Future research directions might involve exploring more convoluted recombination strategies, which focus on offspring inheriting non-blocking pairs from the parents.

In general, the 1-to-1 matching is an interesting problem to be modelled using multi-objective evolutionary algorithms, and the lack of more traditional solutions stimulates further work in the field.

7 Appendix

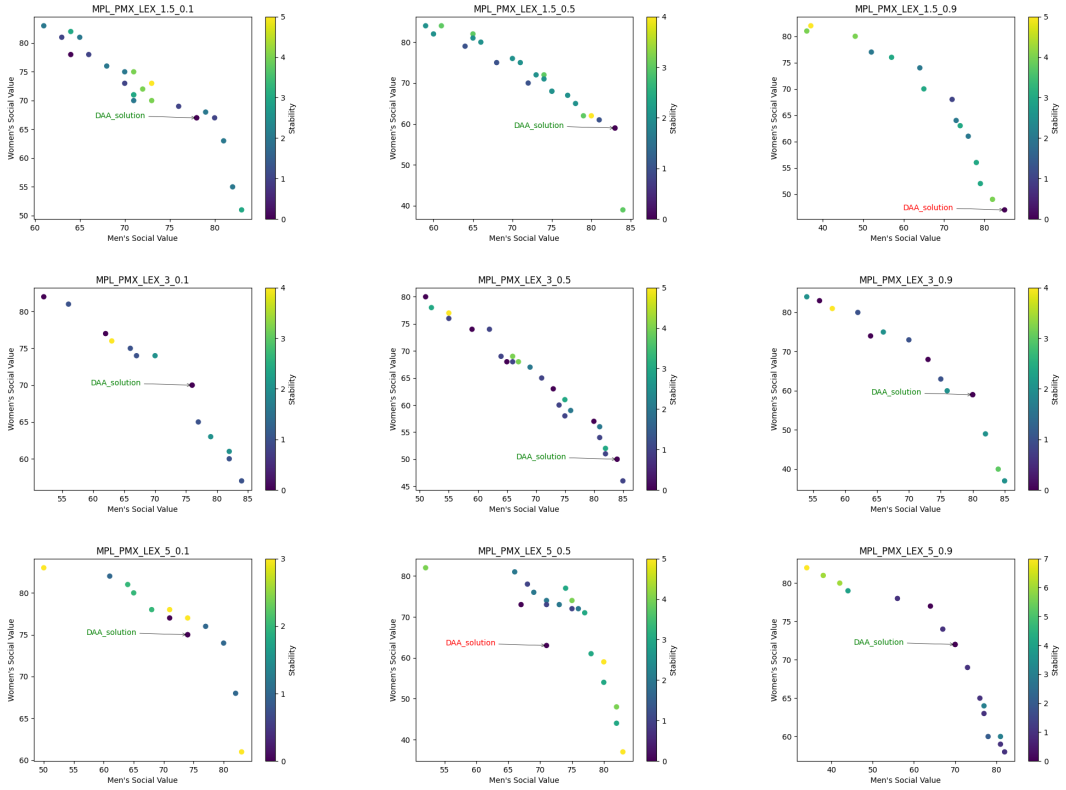

Figure 5: plot of the Pareto fronts obtained from all 9 runs of the PO stage. Number of blocking pairs is color-coded. The men-proposing DAA stable matching is highlighted in green if the evolutionary algorithm was able to find, red otherwise.

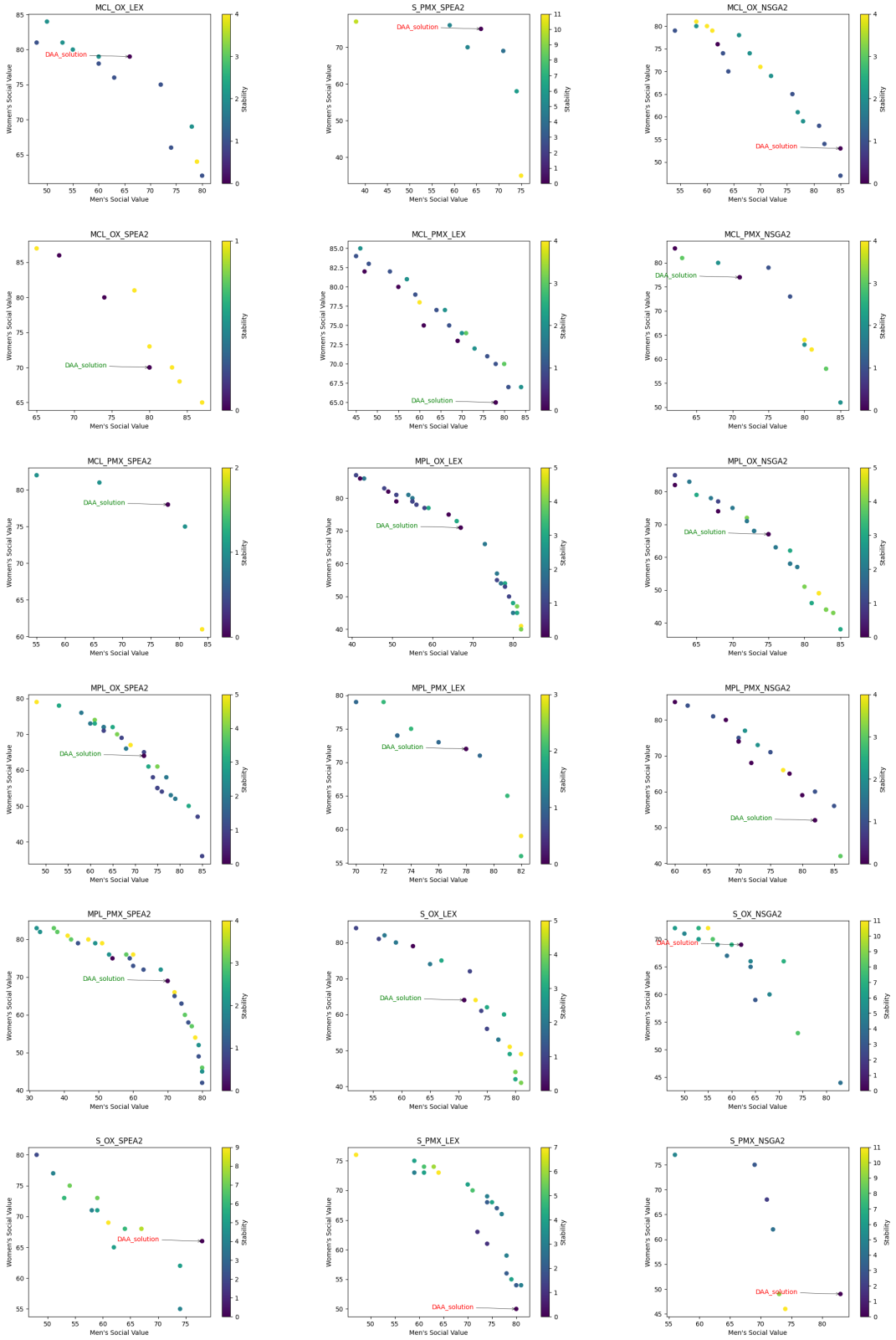

Figure 6: plot of the Pareto fronts obtained from all 18 runs of the AD stage. Number of blocking pairs is color-coded. The men-proposing DAA stable matching is highlighted in green if the evolutionary algorithm was able to find, red otherwise.