# Matching: A Python library for solving matching games with an application to the student-project allocation problem

October 2019

## Summary

Matching is a library for creating and solving instances of matching games. A matching game is typically defined by two sets of players that each have preferences over at least some of the elements of the other. The objective of the game is to find a mapping between the sets of players that maximises the satisfaction of at least one of the sets and the stability of its pairings.

One of the simplest types of matching game models the Stable Marriage Problem (SM). In SM, we have two sets of size $N$: a set of suitors $S$ and a set of reviewers $R$. Each suitor must strictly rank all of the reviewers, and vice versa. We call this arrangement of suitors, reviewers, and their preferences a game of size $N$ (Gale and Shapley 1962).

A matching is any bijection $M$ between $S$ and $R$, and it is considered to be stable if it contains no blocking pairs. In SM, a blocking pair is any pair $(s, r) \in S \times R$ that would rather be matched to one another than their current match. This definition is different for other matching games but the spirit is the same in that a pair blocks a matching if their envy is rational. Irrational envy would be where one player wishes to be matched to another over their current match but the other player does not (or cannot) reciprocate.

Consider the unsolved game of size three shown in Figure 1 as an edgeless graph with suitors on the left and reviewers on the right. Beside each vertex is the name of the player and their associated ranking of the complementary set's elements.

Gale and Shapley (1962) presented an algorithm for finding a unique, stable and suitor-optimal matching to any instance of SM. The matching this algorithm produces is shown in Figure 2.

Using Matching, this game can be modelled as follows:

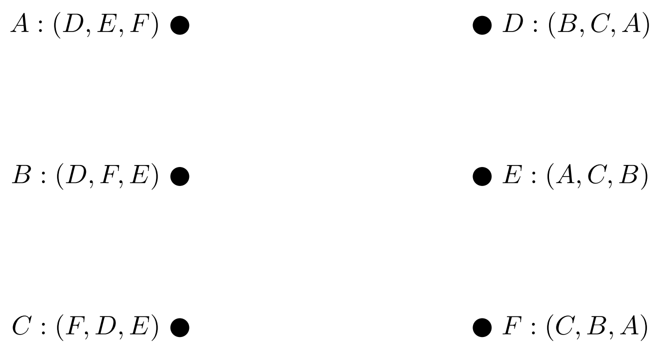

Figure 1: A game of size three.

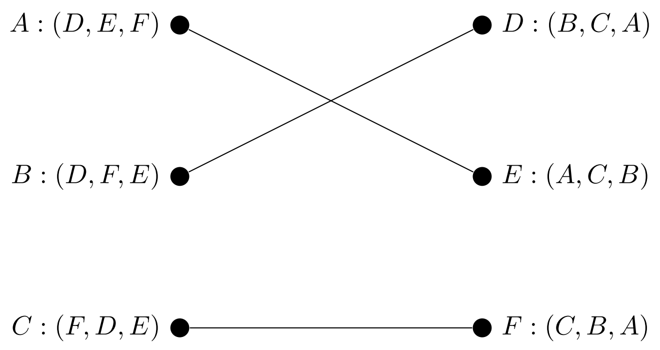

Figure 2: A stable, suitor-optimal solution.

```
>>> from matching import Player
>>> from matching.games import StableMarriage

>>> suitors = [Player(name="A"), Player(name="B"), Player(name="C")]
>>> reviewers = [Player(name="D"), Player(name="E"), Player(name="F")]
>>> (A, B, C), (D, E, F) = suitors, reviewers

>>> A.set_prefs([D, E, F])
>>> B.set_prefs([D, F, E])
>>> C.set_prefs([F, D, E])
>>> D.set_prefs([B, C, A])
>>> E.set_prefs([A, C, B])
>>> F.set_prefs([C, B, A])

>>> game = StableMarriage(suitors, reviewers)
>>> game.solve()
{A: E, B: D, C: F}
```

While it is relatively easy to find solutions to games like this with pen and paper, instances of other matching games tend to have more players than this and require the use of software to be solved in reasonable time.

## Statement of Need

- Although applications to other fields (social care), universities encounter this problem
- Very time-consuming when done by hand; able to compute reasonably large instances in very small amounts of time
- Commercial solvers exist; this is open source and allows for reproducible results
- Matching can be used as an educational tool for learning about Gale-Shapley algorithms

## References

Gale, D., and L.S. Shapley. 1962. "College Admissions and the Stability of Marriage." *The American Mathematical Monthly* 69 (1). Mathematical Association of America: 9–15. https://doi.org/10.2307/2312726.