Vrije
Universiteit
Brussel

Declarative Programming Project: Stable Marriage Problem

Mathijs Saey, 94451, mathsaey@vub.ac.be,
1st Master of Science in Applied Sciences and Engineering:
Computer Science

January 6, 2013

1 Introduction

This is the report about the project for the Declarative Programming course. The goal of this project is to write a program that can find stable matchings for a given *stable marriage problem*. The full project assignment can be found at `https://ai.vub.ac.be/node/1040`. The source code of the project can be found at `https://github.com/mathsaey/Declarative-Programming-Project`, along with a copy of this report.

2 Status

All of the functional and non-functional requirements stated in the project assignment have been implemented. Unfortunately, none of the possible extensions were introduced, this is caused by time constrains and other projects. These time constraints also led to basic lists being preferred over other, more powerful data structures. Nonetheless, all of the basic requirements are up and running, regardless of this minor setbacks.

3 Implementation details

3.1 Data Structures

As previously mentioned, lists were the data structures of choice for this project. In a lot of cases, tuples were used within lists to add extra information to certain elements. For instance, when we keep track of the marriage proposals that somebody received during the Gale-Shapley algorithm a list such as the following is used:

```
[(w, [m1,m2,m3]),|...]
```

Besides these cases, standard lists have been used.

3.2 Algorithms

3.2.1 Gale-Shapley

The Gale-Shapley algorithm is used to generate male-optimal and female-optimal stable matchings from a given set of men, women and preference lists. The implementation presented in this project will be discussed below.

3 types of lists are used in this algorithm, their exact purpose is detailed below.

- The first list contains all the currently "married" couples, married means that this couple got together during one of the previous rounds of the algorithm, it does not imply that their relation is stable, or indeed, permanent.
- The second list is the offer list. It contains anybody that can accept engagement offers, along with the names of the offers that person received. If person X and person Y have proposed to person A, and if B received no proposals, than the list would look like this: [(A,[X,Y]),(B,[])].
- Thee third list is the propose list. It contains anybody that can hand out engagement offers, along with the names of the persons that this person can propose to. These names are sorted based on their priority in the preference list. So if person X prefers person A over B and if person Y has already proposed to everybody on his list then the propose list would look like this: [(X,[A,B]),(Y,[])].

The Gale-Shapley algorithm is relatively easy to understand, it's implementation will be presented here. The male-optimal and female-optimal version of the algorithm share the same implementation. The only difference is the roles that the males and females assume, in the following explanation, we'll be discuss the male-optimal version. The female-optimal version is quite easy to deduce from there on out.

Before we start with the algorithm, we prepare a proposal list for the men, as explained above, this list contains every man, along with the females he is willing to propose to, sorted by rating. We pass the proposal list along and start with the actual loop.

Every iteration of the loop happens in 2 phases. In the proposal phase, every unmarried man proposes to the lady that he likes most, on the condition that that he has not yet proposed to her. We can find this lady by simply looking at his proposal list, remember that we remove a lady from his proposal list once he proposed.

In the marriage phase, every lady looks at all the offers that she received and she accepts the engagement of the man that she favours most (which is the man with the lowest rating). A lady also considers her current husband (if she has one) when considering her partner.

After the marriage phase, we can move on to the next iteration of our loop, we simply keep on looping until no more offers are made, this means that every man is engaged.

3.2.2 Brute-force

The brute-force algorithm was the algorithm of choice to tackle the stable-marriage problem. As the name suggests, this algorithm does not have quite the finesse that Gale-Shapley possesses. The brute-force approach is quite simple to explain. We generate a list of all men and a list of all women. Afterwards we permutate the list of women. We match the women with the men and we see if this combination is stable. If it is stable then we have found a stable matching, otherwise we continue with the next combination.

It should be noted that the brute-force algorithm has no male -or female-optimal version, it simply tries every possible combination, and it does not care about the preference that the selected partners have for each other. Furthermore, it should be obvious that Gale-Shapley is far more optimal than the brute-force approach. A brute-force approach is generally an effective yet slow solution, which is proven again here.

3.3 Files

The source code of this project is divided over multiple files, a brief overview of the role that the code in each file plays is presented here.

- *readme.txt:* Contains further guidelines about using the code as well as an example run of the program.
- *project.pl:* Imports all the other files and contains some general code that invokes other functions. It also contains the output.
- *parser.pl:* Contains the parser that adds the data from the preference files to the database.
- *coupling.pl:* Contains the marriage list management code and the brute-force algorithm.
- *galeshapley.pl:* Contains the implementation of the Gale-Shapley algorithm and all the related code.
- *regret.pl:* Contains the code that calculates the regret score of a matching. It also contains the code that sorts matchings by regret.
- *stabilitychecks.pl:* Contains the code that checks if a given matching is stable.