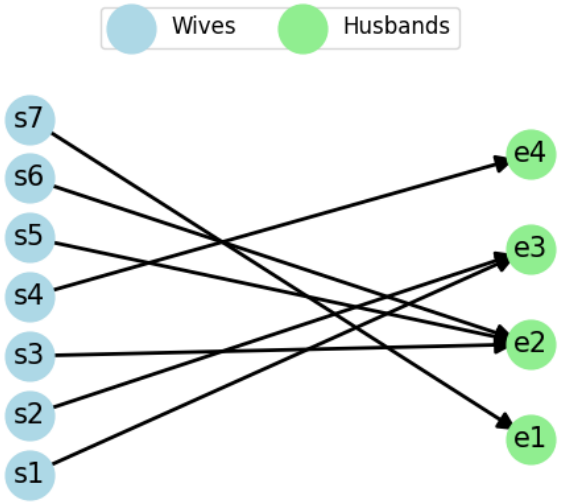
Stable Marriage Problem Implementation Report

Margot Bonafos and Titouan Pastor



Supervised by
Jakllari Gentian

June 21, 2024

Contents

1 Introduction

1.1 Problem Statement

The stable marriage problem is a classic problem in graph theory and game theory. It involves finding a stable matching between two equally sized sets of elements, where stability means that there are no two elements that would prefer each other over their current partners. This project models this problem to assign students to schools, inspired by the French Parcoursup system, where schools can accept multiple students.

1.2 Team

This project was implemented by Titouan Pastor and Margot Bonafos under the guidance of Jakllari Gentian for a course in graph theory. Our goal was to explore and solve a real-world problem using theoretical concepts learned during the course.

2 Algorithm Description

We used the exact same algorithm viewed during course, which will represent wives bidding for husbands. A wife can be the school or the student. The algorithm is as follows:

Algorithm 1 Refinement of the Algorithm

```
Moreproposals ← 1
while Moreproposals do
    countProposals ← 0
    for each husband in husbandList do
        for i = 0 to husband.capacity do
            if i < |husband.preferenceList| then
                wife ← husband.preferenceList[i]
                if result[day][wifeList.index(wife)] is None then
                    result[day][wifeList.index(wife)] ← []
                end if
                result[day][wifeList.index(wife)].append(husband)
                husband.capacity ← husband.capacity - 1
                countProposals ← countProposals + 1
            end if
        end for
    end for
    if countProposals == 0 then
        Moreproposals ← 0
        displayResult(day, result, wifeList)
        displayGraph(day, debugging, husbandList, result, wifeList)
    else
        for each wife in wifeList do
            if result[day][wifeList.index(wife)] is not None then
                while |result[day][wifeList.index(wife)]| > wife.capacity do
                    remove worstHusband from result[day][wifeList.index(wife)]
                    update worstHusband's preferenceList and capacity
                end while
                result[day+1][wifeList.index(wife)] ← result[day][wifeList.index(wife)]
                for each husband in result[day][wifeList.index(wife)] do
                    remove wife from husband's preferenceList
                end for
            end if
        end for
    end if
    day ← day + 1
end while
```

In our adaptation, we modified the algorithm to handle the multiple capacities of schools. This means schools can tentatively accept up to n students where n is the school's capacity.

We also ommitted the parsing algorithm with the excel file as this is not taking part of the stable mariage one. Before running our main algorithm, we parse the excel page(parsing.py), we create a list of students and a list of schools. Each student has a list of preferences for schools and each school has a list of preferences for students. We also have the capacities of each school. We then run the algorithm on this data after checking its integrity(see section 7).

3 Implementation

The implementation was done in Python, utilizing the following key components:

- **Data Input:** Preferences for students and schools are read from an Excel (.xlsx) file, with each sheet representing a different scenario.
- **Data Structures:** Lists and dictionaries store preferences and current matchings.
- **Algorithm:** The modified stable mariage algorithm is implemented to handle multiple capacities for schools.
- **Integrity Check:** A script checks the input data for consistency and correctness before running the algorithm.
- **Visualization:** Results are visualized using a graph where nodes represent students and schools, and edges represent matchings.

4 User Manual

To run the algorithm, follow these steps:

1. Ensure you have Python 3 and the necessary libraries installed. Use the 'requirements.txt' file to install dependencies:

   ```
   pip install -r requirements.txt
   ```

2. Prepare the input Excel file with the correct format: each sheet should represent a different scenario with students' and schools' preferences. It is already filled with some scenarios.
3. Run the script by executing:

   ```
   python parcoursup.py
   ```

4. The script will check the integrity of the Excel file. If errors are found, you will be prompted to correct them.
5. After successful validation, the algorithm will run and produce a stable matching. Results will be displayed graphically and saved to a file if specified.

5 Algorithm Results

We tested the algorithm on several scenarios, including:

- **Handmade Cases:** Manually created scenarios to test specific conditions and edge cases.
- **Random Cases:** Automatically generated scenarios to test the algorithm's robustness and performance.

The results demonstrated that the algorithm quickly finds stable solutions. For example, in scenarios with varying school capacities and student preferences, the algorithm consistently produced matchings where no student and school pair would prefer each other over their current assignments.

5.1 Case Study: Handmade Scenario

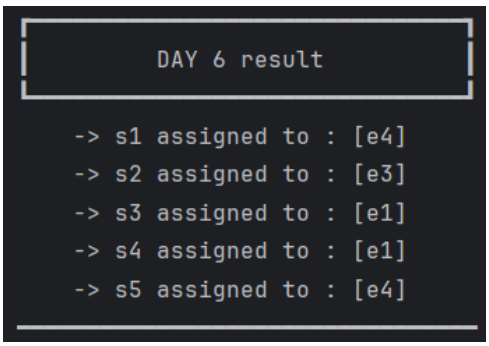
5.1.1 Scenario SC-3

- Input tab

Capacity	**Schools**	**s1**	**s2**	**s3**	**s4**	**s5**
12	e1	4 - 3	2 - 3	1 - 2	5 - 1	3 - 4
1	e2	3 - 4	1 - 2	4 - 4	5 - 2	2 - 3
1	e3	2 - 1	1 - 1	3 - 1	5 - 4	4 - 2
3	e4	3 - 2	1 - 4	4 - 3	2 - 3	5 - 1

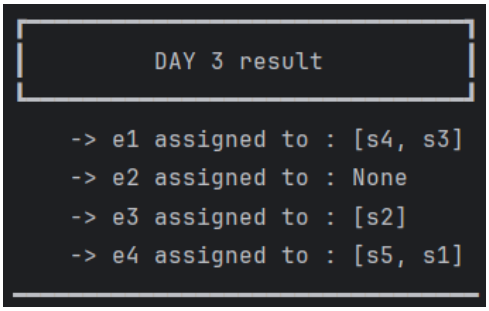
- Results when Students are wifes

	S1	S2	S3	S4	S5
Day 1	e1 - e4	e1 - e2 - e3 - e4	e1	e1 - e4	e1
Day 2	e4	e3	e1 - e4	e1	e1 - e2 - e4
Day 3	e4 - e2	e3	e1	e1	e4
Day 4	e4	e3	e1	e1	e4



- Results when Schools are wifes

	E1	E2	E3	E4
Day 1	S4		S1 - S2 - S3	S5
Day 2	S4 - S3		S2	S5 - S1



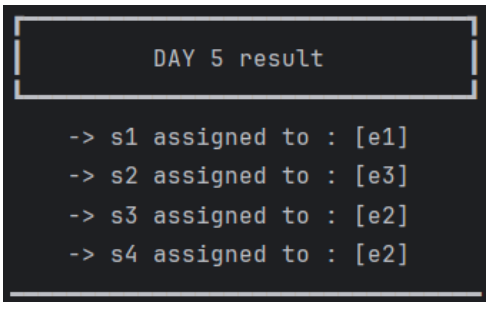
5.1.2 Scenario SC5

- Input tab

Capacity	**Schools**	**s1**	**s2**	**s3**	**s4**
1	e1	1 - 1	4 -1	3 - 1	2 - 2
2	e2	4 - 2	3 -3	2 - 3	1 - 1
1	e3	2 - 3	1 - 2	3 - 2	4 - 3
1	e4	3 - 4	2- 4	4 - 4	1 - 4

- Results when Students are wifes

	S1	S2	S3	S4
Day 1	e1	e3	e2	e2 - e4
Day 2	e1	e3 - e4	e2	e2
Day 3	e1 - e4	e3	e2	e2
Day 4	e1	e3 - e4	e2	e2
Day 5	e1	e3	e2	e2



- Results when Schools are wifes

	E1	E2	E3	E4
Day 1	S1 - s2 - s3	S4		
Day 2	S1	S4	s2 - S3	
Day 3	S1	S4 - s3	S2	

```
        DAY 4 result

-> e1 assigned to : [s1]
-> e2 assigned to : [s4, s3]
-> e3 assigned to : [s2]
-> e4 assigned to : None
```

To conclude, we find the same results when we do it by hand and via the algorithm.

6 Graphs

We included graphical visualizations of the results with the package networkx. In these graphs:

- Students and schools are represented as nodes.
- Edges between nodes represent matchings.
- The graph allows for a clear visual verification of stability and completeness.

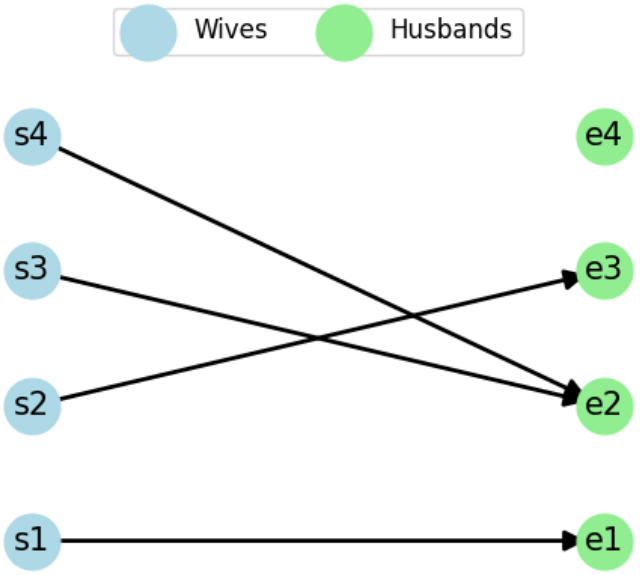


Figure 1: Example of stable matching graph (sc-4)

As you can see here, the graph shows the stable matching between students and schools. Students are on the left representing the wives and schools are on the right representing the husbands. The edges represent the matchings between students and schools. The graph is a clear visual representation of the stable matching found by the algorithm.

7 File Integrity

We developed an integrity check script to ensure the input Excel file is correctly formatted. The script checks:

- Non-zero capacities for schools.
- No duplicate preferences.
- Consistency in the number of students and schools.

If an error is detected, the user is informed to correct the data. This ensures that the algorithm runs smoothly and produces valid results. Exemple

7.1 Exemple with scenario SC-9 : duplicate of preferences

- Input tab

Capacity	Schools	s1	s2	s3	
1	e1	1 - 1	2 - 1	1 - 2	3 - 1
1	e2	1 - 3	3 - 3	4 - 4	2 - 3
1	e3	2 - 3	1 - 2	3 - 1	3 - 2
1	e4	3 - 4	2 - 4	4 - 3	1 - 4

```
Checking the integrity of the scenario...
● An error occurred while checking the integrity of the scenario:
    ╳ The student s1 assigns priority 3 several times for different schools
```

8 Conclusion

In conclusion, developing this algorithm to solve the stable marriage problem with the complexities of the French Parcoursup system was both challenging and rewarding. We successfully implemented a robust solution that ensures stable matchings, even with multiple capacities for schools. The project allowed us to apply theoretical concepts to a real-world problem, enhancing our understanding and skills in graph theory.

Overall, it was an interesting and cool experience to bring this algorithm to life, and we are pleased with the results and insights gained.