COMP4801 Final Year Project

A Web-based Project Allocation System

Interim Report

LEI Wan-hong, 3035202750
Department of Computer Science
The University of Hong Kong

supervised by
Dr. T. W. Chim

January 22, 2017
Abstract

This project presents a stable matching problem between students and projects called the Student-Project Allocation problem (SPA) where students can work collaboratively as a group. A matching between two sets students $S$ and projects $P$ is stable if there are no two pairs $(s, p)$ and $(s', p')$ such that $s$ prefers $p'$ to $p$ and $s'$ prefers $p$ to $p'$. Extensive researches have been carried out in relation to the stable matching problem and yet only few have addressed the SPA problem depicted in this project.

The SPA with group projects is studied in this project. An algorithm called the Spa-Group is designed which aims to generate the stable matching. In addition, a web application called the Project Allocation System (PAS) is developed to simulate the problem in an actual scenario. This enables users to securely submit their preferences online. It can also be used as a report generating tool in regard to matching details.

Currently, a prototype has been developed. This prototype implemented the algorithm to solve the SPA without group projects. The goal is to convert it to solve the SPA with group projects. Besides, a preliminary draft on the Spa-Group algorithm has been done which is based on an existing algorithm.

Next, a preliminary work on the PAS will be followed. This will address the database and user-interface design of the PAS. The next deliverable will be the PAS demo system. This demo system contains basic functions of the system and will serve as the evolutionary prototype of the final PAS.
Acknowledgements

I would like to thank my supervisor Dr. T. W. Chim for his guidance and support. He is always kind and friendly to us. This project would not be possible without his advice. I am also indebted to Ken for his comments and suggestions on the draft of this report. I learnt invaluable report writing and presentation skills from him.

I also want to express my gratitude to the University of Hong Kong, in particular, the Department of Computer Science, for their comfortable study environment in which I can carry out this project.
Contents

List of Figures iii
List of Tables iv
List of Symbols v
Abbreviations vi

1 Introduction 1

2 Background 3
  2.1 Theoretical Aspect ........................................ 3
    2.1.1 The SPA without Group Projects .................. 3
    2.1.2 The SPA with Group Projects ...................... 6
  2.2 Related Work ............................................. 8

3 Objectives 10
  3.1 The SPA-GROUP Algorithm ................................. 10
  3.2 The Project Allocation System ............................ 10
    3.2.1 Scope ............................................. 10
    3.2.2 Testing Data ...................................... 11
  3.3 Goals .................................................. 11

4 Methodology 12
  4.1 Software Development Process ............................ 12
  4.2 Algorithm Design ......................................... 12
  4.3 Technical Aspect of the PAS ............................... 12
    4.3.1 Back-end Technologies ............................. 13
    4.3.2 Front-end Technologies ............................ 14
5 Current Progress

5.1 The SPA-LECTURER Prototype .......................... 15
5.2 The SPA-GROUP Algorithm Draft ........................ 15

6 Difficulties and Limitations 18

6.1 Technical Challenges ................................... 18
  6.1.1 Algorithm Complexity ................................. 18
  6.1.2 Browsers Compatibility ............................... 18

6.2 Limitations ............................................. 19
  6.2.1 Correctness of the SPA-GROUP algorithm ............ 19
  6.2.2 Definition of SPA with group projects ............... 19

7 Schedule 20

8 Conclusion 22

Bibliography 24

Appendix A Pseudocode of the SPA-LECTURER I
List of Figures

2.1 An example to illustrate (a) Unstable matching and (b) Stable matching, with their assigned projects or students boxed. . . . . . . . . . . . . . . . 5
2.2 Example to reduce the SPA with Group Projects into the SPA . . . . 8
4.1 A high-level system architecture diagram. $L_i$ represents the $i$-th layer. . 13
List of Tables

2.1 Matching status for each project $p_j$ in $P$ and each lecturer $l_k$ in $L$. . . . 4

4.1 3 core technologies used in the front-end in the PAS. . . . . . . . . . . 14

7.1 Tentative schedule. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
List of Symbols

\(G_i\) a set of students in a group.

\(L\) a set of \(m\) lecturers.

\(M\) a matching or an assignment.

\(O\) a set of \(r\) group.

\(P\) a set of \(q\) projects.

\(P_{li}\) a subset of projects \(P\) offered by lecturer \(l_i\).

\(R\) a set of Representative-Members tuples among \(r\) group.

\(S\) a set of \(n\) students.

\(\alpha_i\) preference list of student \(s_i\) in \(S\).

\(\beta_{ij}^l\) preference list of lecturer \(l_i\) in \(L\) on project \(p_j\) in \(P_{li}\).

\(c_i\) project’s capacity constraint.

\(d_i\) lecturer’s capacity constraint.
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPA</td>
<td>Student-Project Allocation problem.</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Delivery Network.</td>
</tr>
<tr>
<td>PAS</td>
<td>Project Allocation System.</td>
</tr>
<tr>
<td>RM-tuple</td>
<td>Representative-Members tuple.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In many universities, students are generally required to undertake a project before they graduate. Lecturers first propose one or more projects for students to choose from. Students can then take any one of these projects and are supervised by the corresponding lecturers. In addition, different students can work together as a group. Usually, these projects are not expected to be fully taken up by students whilst the number of these are generally sufficient for all students.

To reduce the chance of clashing between different students or lecturers, they are required to provide a finite number of projects or students that they are interested in. This list of ordering with priority is known as the preference list. The optimal matching between students and projects is then left to be constructed using a centralised matching algorithm based on their preferences. It is important to realise that a centralised matching scheme is preferred than that of decentralised because of the property of optimality [5]. A decentralised approach cannot guarantee the such property in general.

The problem depicted here is similar to that of the Student-Project Allocation problem (SPA) which can be further generalised by the Two-sided Matching problem [1]. Researchers have been studying similar matching problems for decades. A study suggested two linear-time algorithms to solve the SPA without group projects [1]. However, only few studies have been done on the SPA with group projects [2, 3].

This project aims to make two contributions. First, it introduces an algorithm called the SPA-GROUP to deal with the SPA with group projects. Second, it implements a web application called the Project Allocation System (PAS) to stimulate the match-

\footnote{Definition will be discussed in Section 2.1.}
ing problem. Students and lecturers are able to register and submit their preferences online. This goal of this system is to facilitate secretarial staffs to construct the optimal matching.

Currently, a prototype written in Python has been developed. This prototype used the algorithm which solves the SPA without group projects. Besides, a preliminary draft on the SPA-GROUP algorithm is introduced. Along with this, database and basic user-interfaces design will be followed. Preliminary implementation on the PAS with basic functions is also expected to be completed within next month.

The remainder of this report proceeds as follows. First, a precise definition of the SPA with group projects are presented. Next, the PAS is introduced with clear objectives and goals. Then the implementation details of the PAS and the methodology are followed. Current progress of the project is followed. Finally, difficulties and limitations are discussed and a tentative schedule is included.
Chapter 2

Background

Constructing the optimal matching between students and projects can be done manually when both the size of students and projects are fairly small, possibly within 10. A straightforward approach to construct a matching is to individually match each student with an unallocated project that has a highest priority on the student’s preference list. This decentralised approach is known to be deficient and cannot guarantee the stability property \cite{1}. The result generated by a decentralised matching scheme is thereby not optimal.

In addition, when substantial number of students and projects are required to be matched, constructing such a matching manually can take up to few months or even impossible. A software system that can generate the optimal matching is therefore desperately needed.

2.1 Theoretical Aspect

2.1.1 The spa without Group Projects

An instance of the spa without group projects, or simply the original spa, can be defined as follows. Let

\[ S = \{s_1, s_2, \ldots, s_n\} \] be the set of \( n \) students,
\[ L = \{l_1, l_2, \ldots, l_m\} \] be the set of \( m \) lecturers and
\[ P = \{p_1, p_2, \ldots, p_q\} \] be the set of \( q \) projects.
Each lecturer $l_i$ offers a subset of project $P_l$ which is a partition of $P$. Assume each project is uniquely offered by one lecturer, the disjoint union of $P_{l_1}, \ldots, P_{l_m}$ is then $P$. Each student $s_i$ has a preference list $\alpha_i$ which contains a sequence of subset $P$. For example, if student $s_i$ prefers $p_2$ to $p_1$, then $s_i$’s preference list is $\alpha_i = (p_2, p_1)$. Similarly, each lecturer $l_i$ has a preference list $\beta_i^j$ for each project $p_j$ in $P_{l_i}$ (i.e. offered project) in a sequence. Also, each project $p_i$ has a capacity constraint $c_i$ which limits the maximum number of students in the project. Likewise, each lecturer $l_i$ has a capacity constraint $d_i$ which limits the maximum number of students that the lecturer can supervise. It follows that a matching or an assignment $M$ is a subset of $S \times P$ such that [1]:

1. If $(s_i, p_j)$ belongs to $M$, then $p_j$ belongs to $\alpha_i$, that is, $s_i$’s preference list contains project $p_j$;

2. For each student $s_i$, at most one project $p_j$ is matched to $s_i$ in $M$;

3. For each project $p_i$ in $P$,

\[
\sum_{\forall (s_k, p_i) \in M} s_k \leq c_i,
\]

that is, the number of students matched must not be greater than $c_i$; and

4. For each lecturer $l_i$ in $L$,

\[
\sum_{\forall (s_k, p_j) \in M} s_k \leq d_i,
\]

that is, the number of students matched must not be greater than $d_i$.

Then $(s_i, p_j)$ means that $s_i$ is assigned to $p_j$ and is denoted by $M(p_j) = \{s_i\}$. If $p_j$ is offered by $l_k$ then it also means $l_k$ is assigned to $s_i$ and is denoted by $M(l_k) = \{s_i\}$. Project $p_i$ is under-subscribed, full or over-subscribed if the total number of students assigned to $p_i$ is less than, equal to or greater than the project capacity $c_i$ [1]. This applies to lecturer $l_k$ with capacity $d_k$ who has assigned to $M(l_k)$. [5]. Table 2.1 outlines

<table>
<thead>
<tr>
<th>Matching Status</th>
<th>Under-subscribed</th>
<th>Full</th>
<th>Over-subscribed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project $p_i$</td>
<td>$</td>
<td>M(p_i)</td>
<td>&lt; c_i$</td>
</tr>
<tr>
<td>Lecturer $l_k$</td>
<td>$</td>
<td>M(l_k)</td>
<td>&lt; d_k$</td>
</tr>
</tbody>
</table>

Table 2.1: Matching status for each project $p_j$ in $P$ and each lecturer $l_k$ in $L$. 

4
all possible matching status for each project $p_j$ in $P$ and each lecturer $l_k$ in $L$. The criteria for different matching status are tabulated with the corresponding status in the column title.

Furthermore, the stability of a matching may be defined as follows [4]:

**Definition 1** A matching $M$ is called stable if it does not exist two students $s_1$ with project $p_1$ and $s_2$ with projects $p_2$ that $s_1$ prefers $p_2$ to $p_1$ and $s_2$ prefers $p_1$ to $p_2$.

Figure 2.1 is an example to illustrate the difference between stable and unstable matching. Suppose lecturer $l_1$ offers project $p_1$ and that of $l_2$ offers project $p_2$. Student $s_1$ has a preference list of $\alpha_1 = \{p_2, p_1\}$ such that $p_2$ is more preferable than $p_1$. This priority, as defined, applies to preference list of students or lecturers.

![Diagram](image)

Figure 2.1: An example to illustrate (a) Unstable matching and (b) Stable matching, with their assigned projects or students boxed.

An unstable matching $N$ happens when $N = \{(s_1, p_1), (s_2, p_2)\}$ (see Figure 2.1(a)). When such a pair exists, a stable matching $M = \{(s_1, p_2), (s_2, p_1)\}$ (see Figure 2.1(b)) can always obtain by swapping their projects. This stable matching is better than that of the unstable because it prevents “private” re-assignment, that is, $s_1$ and $s_2$ agree to swap their projects given an unstable matching $N$.

There are potentially many stable matchings existed in a SPA instance. However the optimal one is preferred among all these matchings [4].

**Definition 2** A stable matching is called optimal if every student is at least as well off in this matching as in any other stable matchings.

This definition says that a stable matching is optimal if and only if it matches each student with a project that has a highest priority on the its preference list than that of the other projects among other stable matchings. This optimal matching is unique if it is existed [4].
2.1.2 The SPA with Group Projects

Given the definition in the original SPA, an instance of SPA with group projects may be defined as follows. Let

\[ O = \{ G_1, G_2, \ldots, G_r \} \]

be the set of \( r \) group.

Each group \( G_i \) in \( O \) is a subset of students \( S \) such that each student \( s_k \) in \( S \) is uniquely existed in \( O \). Specifically, if \( s \) in \( G \) then \( s \) will not in \( O \setminus G \). This means a student cannot belong to 2 different group. A single student \( s \) can also in a group \( G_i = \{ s \} \). The cardinality of each group \( |G_i| \), or the group capacity, is the total number of members in the group \( G_i \). It follows that a set of \( r \) group \( O \) can be expressed as

\[
O = \left\{ \left\{ s_{1,1}, \ldots, s_{1,|G_1|} \right\}, \left\{ s_{2,1}, \ldots, s_{2,|G_2|} \right\}, \ldots, \left\{ s_{r,1}, \ldots, s_{r,|G_r|} \right\} \right\},
\]

where each \( s_{i,j} \) is the \( j \)-th student in a group \( G_i \). A representative of each group \( G_i \) is elected among themselves who delegates the group \( G_i \). Without loss of generality, the first student \( s_{i,1} \) in each group \( G_i \) is assumed to be the representative of the group \( G_i \).

A Representative-Members tuple (RM-tuple) \( r_i \) for each group \( G_i \) is defined as

\[ r_i = (\text{representative of } G_i, \text{ number of members in group } G_i). \]

A set of RM-tuples \( R \) can then be constructed as follows.

\[ R = \{ (s_{1,1}, |G_1|), (s_{2,1}, |G_2|), \ldots, (s_{r,1}, |G_r|) \} = \{r_1, r_2, \ldots, r_r\}. \]

This representation \( R \) over a set of \( r \) group \( O \) reduces the problem into the original SPA with additional group capacity constraint \( |G| \). Each group \( G_i \) in \( O \) has a total number of members \( |G_i| \) who is delegated by a single student \( s_{i,1} \). All students \( s_{i,1}, \ldots, s_{i,|G_i|} \) in the same group \( G_i \) has the same preference which is delegated by \( s_{i,1} \). Each lecturer \( l_k \) only need to rank the RM-tuple of group.

Similar to the original SPA, a matching \( M' \) is a subset of \( R \times P \) such that:

1. If \((r_i, p_j)\) belongs to \( M' \), then \( p_j \) is in \( s_{i,1} \) preference list;
2. For each group \( G_i \), at most one project \( p_j \) is matched to \( G_i \) in \( M' \);
3. For each project \( p_i \) in \( P \),
\[
\sum_{\forall (r_k, p_i) \in M'} |G_k| \leq c_i,
\]
that is, the total number of students in all group matched must not be greater than \( c_i \); and

4. For each lecturer \( l_i \) in \( L \), for all \( p_j \) in \( P_{l_i} \)
\[
\sum_{\forall (r_k, p_j) \in M'} |G_k| \leq d_i,
\]
that is, the total number of students in all group matched must not be greater than \( d_i \).

Then \((r_i, p_j)\) means that all students in the group \( G_i \) are assigned to \( p_j \). If \( p_j \) is offered by \( l_k \) then \( l_k \) supervises all students in the group \( G_i \). The matching status of \( M'(p_j) \) and \( M'(l_k) \) are, however, required to define rigorously to avoid confusion. The matching status of \( p_j \) in \( M' \) is defined as:

**Definition 3** If \( M' \in R \times P \),
\[
M'(p_j) = \{ G_i \in S \mid (r_i, p_j) \in M' \}
\]
is called the **matching status** of \( p_j \) in \( M' \) and is denoted by \( M'(p_j) \).

Likewise, the matching status of \( l_k \) in \( M' \) is defined as:

**Definition 4** If \( M' \in R \times P \),
\[
M'(l_k) = \{ G_i \in S \mid (r_i, p_j) \in M' \text{ and } p_j \in P_{l_k} \}
\]
is called the **matching status** of \( l_k \) in \( M' \) and is denoted by \( M'(l_k) \).

It follows that the matching status of projects or lecturers is the same as in Table 2.1. Essentially, the status is summing all students from all group matched instead of all students matched.

Besides, the stability property will be followed provided that both projects have sufficient capacities to hold the group members (i.e. group capacity \(|G|\)) after swapping. The optimality is then defined as in the original SPA.
Students $S$ Group $G$ Set $R$

$$s_1$$
$$s_2$$
$$s_3$$
$$s_4$$
$$s_5$$
$$s_6$$

$G_1$

$$s_1, 3$$

$G_2$

$$s_4, 2$$

$G_3$

$$s_6, 1$$

SPA Group

Lecturers’ Preferences

Stable Matching $M' \in R \times P$

SPA with additional group capacity constraints

Figure 2.2: Example to reduce the SPA with Group Projects into the SPA

Figure 2.2 illustrates an example of reduction from the original SPA to the new SPA with group projects. There are 6 students $S = \{s_1 \cdots s_6\}$ with 3 groups. Group $G_1$ has students $s_1, s_2, s_3$, group $G_2$ has students $s_4, s_5$ and group $G_3$ has student $s_6$ only. The RM-tuple of $G_1$ is therefore $r_1 = (s_1, 3)$, and $G_2$ is $r_2 = (s_4, 2)$, and $G_3$ is $r_3 = (s_6, 1)$. The SPA-Group will then generate a stable matching $M'$ from the preferences and the RM-tuples.

2.2 Related Work

There are two linear-time algorithms, SPA-STUDENT and SPA-LECTURER, suggested to solve the original SPA using appropriate data structures [1]. Conceivably, either students or lecturers are allowed to reject their temporary assignments in order to subsequently allocate better assignments. Therefore the SPA-STUDENT algorithm generates a student-oriented matching but the SPA-LECTURER algorithm generates a lecturer-oriented matching. However, this paper defined lecturers’ preference lists differently. Each lecturer $l_k$ has a single preference list which contains all students who are enrolled in $l_k$’s projects. All students in all projects are then rank accordingly. This allows lecturers to rank their students once only.
Another research studied a SPA problem with grouping [3]. But the preference list of lecturers is absent. Instead, the relationship between students and projects is determined by a weighting factor. They proved that such problem is NP-COMPLETE unless each project has a capacity smaller than or equal to 2, that is, \( c_i \leq 2 \) for all \( p_i \) in \( P \). An approximation algorithm is suggested to tackle this problem in general. A linear programming technique is also suggested by a study [2]. Because this technique is not consider in this project, the result of the study may not be applicable.
Chapter 3

Objectives

This project has two main objectives. First, it aims to design an enhanced algorithm called the Spa-GROUP to solve the Student-Project Allocation problem (SPA) with group projects. This algorithm generates an optimal matching based the constraints stipulated in Section 2.1. Second, it aims to implement a web application called the Project Allocation System (PAS). This web-based system utilises the enhanced algorithm to generate the optimal matching.

3.1 The Spa-GROUP Algorithm

The Spa-GROUP algorithm should contain following properties:

1. Optimal matching. The matching generated should be optimal.

2. Lecturer-oriented matching. Only lecturers are allowed to reject their temporary assignments.

3.2 The Project Allocation System

3.2.1 Scope

The PAS should contain following major functions:

1. Student registration. Each student can register as a user in the system.

2. Proposing projects. Each lecturer can propose various projects.
3. *Ranking projects/students.* Users are allowed to rank a finite number of projects or students.

4. *Generate the optimal matching.* After the system collected preference lists from all users, the optimal matching is generated.

5. *Checking Stability.* A function to check if the generated matching is stable or not.

6. *Administrator panel.* For administrator to manage the system. One major function is to approve generated matching.

7. *Generating Reports.* The system can generate various metrics based on the matching result. These metrics can be the total number of matchings, loading factor of each lecturer $l_k$ (i.e. ratio of supervised projects $M(l_k)$ to capacity $d_k$).

### 3.2.2 Testing Data

The system will first take a simple $5 \times 5$ matching which comprises 5 students, lecturers and projects for initial testing. The optimal matching will be done manually to check against the generated matching. The number 5 is chosen because the optimal matching can be readily done by hand and checking against it is effortless.

A random test case generator will then be programmed after the system passes the initial testing. A set of $50 \times 50$, $100 \times 100$, $500 \times 500$ and $1000 \times 1000$ matchings will be generated for testing purpose. However, the stability of the results can only be checked using the stability checking function because of the large size of data set.

### 3.3 Goals

The ultimate goal of this project is to improve the efficiency in constructing the optimal matching. It is a tedious and painstaking task to construct this matching manually. The web application PAS is built to achieve this goal. By utilising the web nature of the system, the time and cost in collecting user’s preferences can be greatly reduced. The optimal matching between students and projects can then be easily generated. In addition, the SPA-GROUP algorithm allows different students work together as a group while maintaining the optimality of the matching.
Chapter 4

Methodology

4.1 Software Development Process

An unified software development process is used in this project. The advantage of this process is that error can be readily rectified even at the later phase of the project. The entire development process is divided into three distinct phases where each phase contains iterative and incremental cycles. Changes are constantly made in each cycle and followed by a testing. This can accelerate the development process. The three phases are Inception, Elaboration and Construction phases. The details of each phase are given in Table 7.1.

4.2 Algorithm Design

The SPA-GROUP algorithm may be designed based on the SPA-LECTUER algorithm which solves the SPA without group projects [1]. The pseudocode of SPA-LECTUER is attached to the Appendix A. It has been proved that the generated is optimal by a rigorous proof on its correctness. The SPA-GROUP algorithm is expected to be based on the SPA-LECTURER algorithm upon modification.

4.3 Technical Aspect of the PAS

Since the Project Allocation System (PAS) is a web application, both web server and database management system are necessary. A high-level system architecture diagram is illustrated in Figure 4.1. The system consist of front-end and back-end (see the left
hand side of Figure 4.1) and is categorised into 5 layers (see the right hand side of Figure 4.1). All software used in this project are free and open source.

<table>
<thead>
<tr>
<th>Front-end</th>
<th>Back-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Allocation System (PAS)</td>
<td>Hardware</td>
</tr>
<tr>
<td>NGINX</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>Django</td>
<td>GNU/Linux</td>
</tr>
</tbody>
</table>

proxy to Django using uwsgi module

Figure 4.1: A high-level system architecture diagram. $L_i$ represents the $i$-th layer.

### 4.3.1 Back-end Technologies

The details of each layer are discussed below. Each layer can be referred to the right hand side of Figure 4.1.

$L_0$: **Hardware Layer**

This layer contains necessary hardware for the PAS to deploy.

$L_1$: **Operating System**

The PAS will deploy under the GNU/Linux operating system which operated above the $L_0$: Hardware Layer. This offers a robust and security platform for software to run.

$L_2$: **Database System**

PostgreSQL is chosen as the database system. It is an object-relational database system (ORDS) that is renowned for its reliability and data integrity.

$L_3$: **Web Server**

The PAS uses the Django web framework as its back-end system. The scripting language for the Django is Python. With comprehensive built-in functions in Django, it provides a rapid development environment.
To handle users’ requests, NGINX is chosen for the web server. It is a lightweight and yet a highly scalable server compared with Apache. By adding the uWSGI module, it serves as a proxy server to redirect requests to the Django web framework so as to provide dynamic web contents. Whenever there is a request sent from the user, NGINX will redirect the request to Django through the uWSGI module.

Together with the PostgreSQL database system, the performance of the PAS will outperform the traditional LAMP stack which comprises GNU/Linux, Apache web server, MySQL database system and PHP.

### 4.3.2 Front-end Technologies

The front-end comprises of the web application PAS which operated in the $L_4$: Application Layer (see Figure 4.1). There are 3 core web technologies used in this layer. These technologies are HTML, CSS and JavaScript. Each following technology has following function on the PAS:

<table>
<thead>
<tr>
<th>Technology</th>
<th>Full name</th>
<th>Major functions on a web page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
<td>Outline the structure of a web page, it serves as the building blocks</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
<td>Set the visual style to provide a better user interface</td>
</tr>
<tr>
<td>JavaScript</td>
<td>N/A</td>
<td>Provide interactive elements</td>
</tr>
</tbody>
</table>

Table 4.1: 3 core technologies used in the front-end in the PAS.
Chapter 5

Current Progress

At present, a SPA-LECTURER prototype has been implemented to solve the Student-Project Allocation problem (SPA) without group projects. The aim of the prototype is to provide a pragmatic solution for the SPA without group projects. In addition, a preliminary draft on the SPA-GROUP algorithm has been done. It is based on the existing SPA-LECTURER algorithm.

5.1 The SPA-LECTURER Prototype

This prototype is a simple command-line application written in Python such that it can be readily integrated into Django subsequently. It is built upon the pesudocode of SPA-LECTURER (see Appendix A). The purpose of this prototype is to illustrate the underlying principle of the matching procedure of SPA without group projects. It is hoped that this prototype can convert to support SPA-GROUP algorithm.

A generator is also developed to generate random data set for testing. Currently, matching size up to ten thousand is tested and the result is stable.

5.2 The SPA-GROUP Algorithm Draft

The pseduoicode of the current SPA-GROUP algorithm is listed in Algorithm 1. The algorithm takes an instance $I$ of SPA with group projects as an input. Students and lecturers are initialised as free which mean no project has been matched to them. The matching $M' \in R \times P$ is also initialised as an empty set. The algorithm then continue search for a lecturer $l_k$ with an under-subscribed project $p_j$ in $P_{l_k}$ (i.e. offered project $p_j$
is under-subscribed). If such a project $p_j$ exists for lecturer $l_k$, a group $G_i$ has following conditions in order to match with $p_j$:

1. on the $l_k$’s preference list on project $p_j$ (i.e. $\beta^j_k$), the first RM-tuple $r_i = (s_{i,1}, |G_i|)$ of the group $G_i$ who is delegated by $s_{i,1}$, must be preferred $p_j$ to current temporary assignment, if any; and

2. on the $s_{i,1}$ preference list $\alpha_{i,1}$, $p_j$ must be the first such under-subscribed project; and

3. the group capacity $|G_i|$ must not be larger than the current current $l_k$ availability, that is

$$|G_i| < (d_k - |M'(l_k)|),$$

such that $l_k$ can supervise all students from group $G_i$.

**Algorithm 1** The pseudocode of the current SPA-GROUP.

```plaintext
1: procedure SPA-GROUP($I$)
2: initialise each student, project and lecturer to be free
3: $M' = \emptyset$  \textcolor{gray}{$\triangleright$ the matching result}
4: while there is a lecturer $l_k$ with a project $p_j \in P_{l_k}$ under-subscribed do
5: \hspace{1em} $r_i = $ first such RM-tuple on $l_k$’s preference list on project $p_j$
6: \hspace{1em} $p_j = $ first such project on $s_{i,1}$’s preference list
7: \hspace{1em} if $|G_i| > (d_k - |M'(l_k)|)$ then \textcolor{gray}{$\triangleright |G_i|$ larger than $l_k$ capacity}
8: \hspace{2em} delete $r_i$ on $l_k$’s preference list
9: \hspace{2em} continue
10: \hspace{1em} if $(r_i, p_j) \notin M'$ then
11: \hspace{2em} if $s_{i,1}$ is assigned to a project $p$ then
12: \hspace{3em} delete $(r_i, p) \in M'$
13: \hspace{3em} assign $(r_i, p_j) \in M'$ \textcolor{gray}{$\triangleright l_k$ offers $p_j$ to $G_i$}
14: \hspace{2em} for all successor $p_l$ of $p_j$ on $\alpha_{i,1}$ do
15: \hspace{3em} delete $p_l$ on $\alpha_{i,1}$ and $\beta^d_l$ \textcolor{gray}{$\triangleright$ if $p_l$ is offered by $l_d$}
16: \hspace{2em} else
17: \hspace{2em} continue \textcolor{gray}{$\triangleright$ next lecturer or project}
18: return $M' \in R \times P$
```

If the condition 3 fails, then the algorithm will remove the RM-tuple $(s_{i,1}, G_i)$ from $l_k$’s preference list $\beta^j_k$ and continue search for next group $(s_{i+1,1}, G_{i+1})$ on the list. These criteria guarantee the optimality of a matching. When such a group $G_i$ is found, the algorithm will break the current assignment of $G_i$, if any, such that $p_j$ offered by $l_k$ can
be assigned to $G_i$. It follows that for each projects $p_l$ such that $s_{i,1}$ prefers $p_j$ to $p_l$ is removed from $s_{i,1}$’s preference list $\alpha_{i,1}$. If $p_l$ is offered by $l_d$, then $p_l$ is also removed from $l_d$’s preference list $\beta_{d}^{l}$. It is because both of them will not prefer a project (or a student) that has lower preference than their current assignment. The algorithm terminates when no $l_k$, $p_j$ and $G_i$ can be found.

Nevertheless, the current SPA-GROUP algorithm has not yet been proved or implemented and subsequent modification is expected. It serves as the purpose to explore the possibilities in solving the problem based on the SPA-LECTURER algorithm [1]. The SPA-GROUP algorithm is largely contingent upon this algorithm.
Chapter 6

Difficulties and Limitations

6.1 Technical Challenges

6.1.1 Algorithm Complexity

The Student-Project Allocation problem (SPA) with group projects introduced in this project can be an NP-COMPLETE problem. Since such problems cannot be solved in polynomial-time, the performance of the SPA-GROUP algorithm will be greatly reduced when the input is vast, say millions of students or projects. Generally, techniques like approximation or randomisation are used to cope with NP-COMPLETE problems. In this project, parameterisation technique may be used if it is an NP-COMPLETE problem. Parameter(s) is/are fixed to solve the problem in polynomial-time. For example, capacity of projects or lecturers is fixed to a upper or lower bound.

6.1.2 Browsers Compatibility

Since only web interface will be implemented, there may have discrepancy between different devices. This will lead to difficulties for users to use the system. It is hoped that by using some well-written web front-end frameworks, the user interface can maintain its consistency across different devices.
6.2 Limitations

6.2.1 Correctness of the SPA-GROUP algorithm

Since this project mainly focuses on the Project Allocation System (PAS) implementation, rigorous mathematical proof of the correctness of the SPA-GROUP algorithm is not expected to be included. This leads to a difficulty in analysing the result whether it is the optimal matching or not. When substantial number of students and projects are required to be matched, checking the result manually is impossible. The correctness of the result can only be measured using the Stability Checking function which required exhaustive testings.

6.2.2 Definition of spa with group projects

The definition of spa with group projects in Subsection 2.1.2 has an assumption that each group $G_i$’s representative has delegated all $G_i$ students’ preferences. All members from the same group $G_i$ have assumed to agree representative preference. In general, this assumption is acceptable because otherwise the student $s_d$ will not join the group $G_i$ if $s_d$ has a different preference than that of $s_{i,1}$. However, a more general situation where each student $s_k$ in group $G_i$ has a different preference than that of $s_{i,1}$ is not addressed in this project.
Chapter 7

Schedule

The tentative schedule to design and implement the Project Allocation System (PAS) is given in Table 7.1. There are distinct milestones to be achieved on each phase. There are 3 major deliverables on each phase.

1. **Spa-lecturer Prototype**, *completed, delivered in November 2016.*

   This Spa-lecturer prototype implemented the Spa-LECTURER algorithm. This prototype is programmed in Python and is aimed to solve the SPA without group projects.

2. **The PAS**, to be delivered in April 2017.

   All functions of the PAS, including matching generation and administrator panel, will be completed together with a meticulous final report containing all the details.
<table>
<thead>
<tr>
<th>Date</th>
<th>Tasks</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Oct 2016</td>
<td><strong>Deliverables of Phase 1</strong> <em>(Inception)</em></td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>• Detailed project plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Project website</td>
<td></td>
</tr>
<tr>
<td>Oct 2016</td>
<td>Requirements gathering and analysis</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>• Collect user requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Further study on SPA based on the requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Implement a prototype to solve the SPA</td>
<td></td>
</tr>
<tr>
<td>Nov 2016</td>
<td>Deliver the SPA-LECTURER prototype and the preliminary draft on the SPA-GROUP algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>Dec 2016</td>
<td>Development on the PAS Demo System</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>• Database and user interface design</td>
<td></td>
</tr>
<tr>
<td>22 Jan 2017</td>
<td><strong>Deliverables of Phase 2</strong> <em>(Elaboration)</em></td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>• Detailed interim report</td>
<td></td>
</tr>
<tr>
<td>Feb 2017 to Mar 2017</td>
<td>System Design and Construction</td>
<td>In-progress</td>
</tr>
<tr>
<td></td>
<td>• Refine the PAS Demo System</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Implement the administrator user interface</td>
<td></td>
</tr>
<tr>
<td>Apr 2017</td>
<td>Deployment and Testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Final testing and optimisation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Deploy the PAS</td>
<td></td>
</tr>
<tr>
<td>17 Apr 2017</td>
<td><strong>Deliverables of Phase 3</strong> <em>(Construction)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Finalised PAS implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Final report</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Tentative schedule.
Chapter 8

Conclusion

This project has introduced the web-based Project Allocation System (PAS) that aimed to model the Student-Project Allocation problem (SPA) with group projects in a real-life scenario. The web-based nature of the system allows great flexibility across devices. This SPA with group projects allows different students work together as a group which is commonly adopted by many universities. Also, the project has presented an algorithm called the SPA-GROUP that targeted to generate the optimal matching between students and projects based on the preferences and the capacities. Unlike previous algorithms, it supports grouping between different students.

This report described the implementation details of the PAS, in particular, the system architecture is presented. Although currently only the SPA-LECTURER prototype using the existing algorithm is implemented, it is hoped that this can be possibly converted or merged into the final PAS. In addition, the preliminary draft on the SPA-GROUP algorithm helped to identify the underlying principle of the existing SPA-LECTURER algorithm. Along with this, a PAS Demo System is expected to be delivered next.

Despite the project only describes a matching problem between students and projects, there are versatile applications of this centralised matching algorithm. It is in fact constantly appears on different contexts. For example, in a Content Delivery Network\(^1\) (CDN), a group of users (group of students) from related regions may send requests to the CDN. It will then list out a number of available servers (projects) that can be used to respond to the requests with a priority that possibly based on the geographical location. A centralised matching scheme is run to generate the optimal matching that

---

\(^1\) a distributed network of servers that provides high performance service in delivering web content
maps each user to the closest server. There are potentially many uses of this algorithm that are left to be discovered.
Bibliography


Appendix A

Pseudocode of the SPA-LECTURER

The pseudocode of SPA-LECTURER algorithm is listed in Algorithm 2. This algorithm is an adaption from a study [1]. Changes have been made to conform the definition stipulated in Section 2.1. The main logic of this algorithm is similar to that of the SPA-GROUP algorithm except that no group capacity is checked.

Algorithm 2 The pseudocode of the SPA-LECTURER.

1: procedure SPA-LECTURER(I) 
2: initialise each student, project and lecturer to be free
3: \( M = \emptyset \) \( \triangleright \) the matching result
4: while there is a lecturer \( l_k \) with a \( p_j \in P_{l_k} \) under-subscribed do
5: \( s_i \) = first student on \( l_k \)’s preference list \( \triangleright s_i \in \beta^j_k \)
6: \( p_j \) = first project on \( s_i \)’s preference list \( \triangleright p_j \in \alpha_i \)
7: if \( (s_i, p_j) \notin M \) then
8: if \( s_i \) is assigned to a project \( p \) then
9: delete \((s_i, p)\) \( \in M \) \( \triangleright \) delete current \( s_i \)’s matching
10: assign \((s_i, p_j)\) \( \in M \) \( \triangleright l_k \) offers \( p_j \) to \( s_i \)
11: for all successor \( p_l \) of \( p_j \) on \( \alpha_i \) do
12: delete \( p_l \) on \( \alpha_i \) and \( \beta^l_i \) \( \triangleright \) if \( p_l \) is offered by \( l_d \)
13: else
14: continue
15: return \( M \in S \times P \)