International Journal of Computer Theory and Engineering, Vol. 3, No. 2, April 2011
ISSN: 1793-8201

A Visual Implementation of Student Project Allocation

Mahmoud I. Moussa and Ahmed H. Abu El-Atta

Abstract— **We developed recently a new and novel student project allocation model (SPA-(s, p)) in which the lecturers have preference lists over pairs (student, project), and the students have preference lists over projects. SPA-(s, p) is turned out to be very useful in combination between the student project allocation models with preference lists over students (or projects) [8, 9]. SPA-(s, p) proposes several ways to construct the lecturer's preference lists which give us higher efficiency and accurate results. This study presents new data structure which reduces the space to present an instance of SPA-(s, p). Furthermore, this study presents a visualization of SPA-(s, p) model. The visualization is implemented in java for the fact that it is a web-oriented language.**

Index Terms— **Matching, Algorithms, Visualization.**

I. Introduction

Visualization programs have recently been used more widely to help students understand some algorithms and also clarify some data structures [7, 8, and 9]. In 1999 Byrne, Catrambone and Stasko [10] presented a study which assesses the performance of animation to help students understand and realize the algorithm's steps more effectively. The aim of their study is to examine whether animations helped students to understand the concepts of procedural algorithms. The results showed that the animations help the students to learn the algorithm and expect its behavior more efficiently. In this type of programs the steps of the algorithm are represent by some images and animations to help students understand how this algorithm work. During the execution of the program, the algorithm transforms from state to another. These states can be animated for the students, one after another. The students may have some kind of control over the process, so they can interact with the system in order to stop, continue or step through the animation. In this paper we present visualization program for student project allocation with preference over pairs which was an application of the stable matching problem.

In many colleges, students have to take upon themselves projects in some fields. To do this, the lecturers offer some project topics; both projects and lecturers have capacity constraints. The students choose from among these projects. Each student gives a preference list over the projects that he finds acceptable, the lecturers give preference lists over the students, and other time they give preference lists over the projects that they offer. The student project allocation problem with preference over students (denoted SPA) was studied by Abraham [1] and Abraham and Irving [3]. In their model, the students supply preference lists over projects that were offered by lecturers and each lecturer supplies a preference list over students who show interest in one or more of his projects. Figure 1 describes an instance I_1 of SPA where two students s_1 and s_2 and two lecturers l_1 and l_2 indicated their preferences for the projects and students respectively. Each project has a capacity of one. Lecturer l_1 can supervise two students whereas lecturer l_2 can supervise only one.

Students' preferences	Lecturers' preferences
$s_1 : p_2\ p_1\ p_3$	$l_1 : s_1\ s_2$
$s_2 : p_2\ p_3$	$l_2 : s_2\ s_1$
l_1 offers p_1	l_2 offers p_3

Figure 1: An instance I_1 of the SPA model.

Manlove and O'Malley [2] presented a model for the student project allocation with preference lists over projects denoted SPA-P. Figure2 shows an instance I_2 of the SPA-P model, the students supply preference lists over projects, while the lecturers indicate their preferences for the projects.

Students' preferences	Lecturers' preferences
$s_1 : p_2\ p_3\ p_1$	$l_1 : p_1\ p_2$
$s_2 : p_3\ p_2\ p_4$	$l_2 : p_3\ p_4$
l_1 offers $p_1\ p_2$	l_2 offers $p_3\ p_4$

Figure 2: An instance I_2 of SPA-P model.

SPA is a two-sided matching problem[1] because the input of SPA is a two disjoint sets A (in this case A is the set of students) and B (in this case B is the set of projects), and we seek to match members of A to members of B subject to various criteria. In 2003 M. Thorn [4] presented an automated system for allocating students to projects at the Department of Computer Science, University of York. Other university departments in particular seek to automate the allocation of students to projects [5]. In 2005, D. F. Manlove and G. O'Malley [2] gave a student project allocation with preference over projects (SPA-P). Manlove and O'Malley prove that; the SPA-P model is NP-Complete problem and he gives an approximate algorithm to solve that problem. In 2006 J. Mestre[6] gave a linear time algorithm to find a matching M with the property that there is no other matching M ´ preferred by a weighted majority of agents. The algorithm is for a version of the problem in which each applicant has an associated weight. The authors in [1, 2] notes

Manuscript received May 9, 2010.
Mahmoud I. Moussa is now with Computer Science Department , Faculty of Computers and Information, Benha University, Egypt (e-mail: moussa_6060@yahoo.com).
Ahmed H. Abu El-Atta is now with Computer Science Department, Faculty of Computers and Information, Benha University, Egypt (e-mail: ahmed_123_hasan@yahoo.com).

that a new model over (student, project) pairs may improve the results of the problem.

This paper presents a new data structure for the student project allocation problem with preference lists over (student, project) pairs that we denote SPA-(s, p) to reduce the space that needed to present an instance of SPA-(s, p). In SPA-(s, p) the students supply preference lists over projects, and the lecturers supply preference lists over (student, project) pairs. This study uses java-applet program to present a visualization of the student project allocation algorithm with Preference over Pairs based on the fact that java is a web-oriented language and object-oriented language.

Figure 3 shows an instance I_3 of SPA-(s, p), in which each lecturer has a capacity of two, the projects p_1, p_2, and p_4 have capacities one, whereas project p_3 has a capacity two.

Students' preferences	Lecturers' preferences
$s_1{:}\, p_1\ p_3$	$l_1{:}\,(s_3,p_1)(s_2,p_2)(s_1,p_1)(s_3,\ p_2)$
$s_2{:}\, p_2\ p_3\ p_4$	$l_2{:}\,(s_2,p_3)(s_3,p_3)(s_2,p_4)(s_1,p_3)$
$s_3{:}\, p_2\ p_1\ p_3$	l_1 offers $p_1\ p_2$ l_2 offers $p_3\ p_4$

Figure 3: An instance I_3 of SPA-(s, p).

SPA-(s, p) provides many of the facilities and possibilities for building a preference lists over pairs (student, project). These possibilities achieve a balance between students, which diminish the number of unmatched students by preventing student from quarantine on another. SPA-(s, p) motivates the algorithm to find the maximum cardinality of stable matching. These improvements do not exist in SPA and SPA-p models. To clarify more, assume that the student s_i is the first student in the preference list L_k of l_k. In SPA, the student s_i has a greater opportunity to choose his favorite project among the offers of l_k, this will reduce the chances of other students. SPA-(s, p) overcomes these shortcomings because the lecturers can be twinned between students and projects, the lecturer l_k may prefer s_i to work in some projects, in the same time he prefers other students to work in other projects. For example, a small SPA instance consists of two students s_1 and s_2 and one lecturer l_1 offers the projects p_1 and p_2. Each project has capacity 1, whilst l_1 has capacity 2. Student s_1 prefers p_1 to p_2, whilst s_2 finds only p_1 acceptable. Lecturer l_1 prefers s_1 to s_2. Clearly then, the matching $M_1 = \{(s_1,\ p_2)\ (s_2,\ p_1)\}$ admits the blocking pair $(s_1,\ p_1)$, whilst $M_2 = \{(s_1,\ p_1)\}$ is the only stable matching. In SPA-(s, p) lecturer l_1 prefers s_1 to s_2 too, the lecturer l_1 twines between students and projects, then the preference list of lecturer l_1 may be as the following $l_1 = \{\ (s_1,\ p_2)\ (s_2,\ p_1)(s_1,\ p_1)\}$ so it is clear that M_1 become the optimal matching of that instance. In SPA model lecturer give his preference over students, so if he prefers a student s_i to another one s_r then he will prefer s_i to s_r in all projects he offered. In this case student s_r may be unmatched at all. If the lecturer supplies preference over pairs, the student s_r has a chance to work in one of the projects offered by lecturer l_1 subject to the same criteria. On the other hand; SPA-p model gives preference over projects with indifference between the students, which may deprive the students to work with their preferred projects. But SPA-(s, p) works indifference (cases 2.c and 3.b), and it works too towards the wishes of students and it avoids unexpected un-assignments (cases 2.a, 2.b and 3.a). For example, a small SPA-p instance consists of two students s_1 and s_2 and one lecturer l_1 offers the projects p_1 p_2 and p_3. Each project has capacity 1, whilst l_1 has capacity 2. The student s_1 prefers the project p_3 to the project p_1 and the student s_2 prefers the project p_3 to the project p_2. This instance has two stable matching $M_1 = \{(s_1, p_3)\ (s_2, p_2)\}$ and $M_2 = \{(s_1, p_1)\ (s_2, p_3)\}$. It is clearly that M_2 is better than M_1 but in that model it is NP problem to find best matching, the authors of SPA-p models [2] note that a new model over pairs may solve that problem. In SPA-(s, p) model; The stability can be defined as the following: a stable matching M guarantees that there is no pair $(s_i,\ p_j) \notin M$ where l_k is the lecturer who offers p_j, such that s_i is unassigned or prefers p_j to remain within assignment in M and also l_k is undersubscribed or prefers $(s_i,\ p_j)$ to the worst pair $(s_i,\ p_k)$ in M. A new definition of the blocking pair has been introduced. The remainder of this paper is organized as follows. In section 2 we give a formal definition of the SPA-(s, p) and show some methods to ranking preference lists of lecturers, we present and discuss a student-oriented algorithm for SPA-(s, p). Section 3 is the conclusion of this research.

II. Definition of The SPA-(s, p) Model

An instance of SPA-(s, p) consists of a set of students $S = \{s_1, s_2, \dots, s_n\}$, a set of projects $P = \{p_1, p_2, \dots, p_m\}$, and a set of lecturers $L = \{l_1, l_2, \dots, l_q\}$. Each lecturer l_k offers a non-empty set of projects P_k, so the project set P has the partition $P_1, P_2, \dots, P_q$. Each student s_i supplies a set of projects $A_i \subseteq P$. Then student s_i ranks A_i in strict order to construct his preference list. For any project p_j on s_i's preference list, we say that s_i finds $p_j$acceptable. For each project $p_j \in P_k$ we define L_k^j as the project preference list of p_j by deleting all pairs that do not contain p_j from L_k then we take students from the remaining pairs in the same order of that pairs. Each lecturer l_k has a capacity d_k. Similarly, each project p_j has a capacity c_j. We assume that $Max\ \{c_j : p_j \in P_k\} \le d_k \le \sum\{c_j : p_j \in P_k\}$.

In the other hand, each lecturer l_k scan the students' preference lists to find the students that are find one or more of his project acceptable and he constructs B_k from students' preference lists as follows $B_k = \{(s_i, p_j) \in S \times P : p_j \in P_k\ and\ p_j \in A_i\}$ (i.e. B_k is the set of (s_i, p_j) pairs such that students s_i finds p_j acceptable where p_j is offered by l_k). Each lecturer l_k supplies a preference list L_k ranking B_k. Where B_k consists of (student, project) pairs, the ranking of B_k is depend on the ranking of student, project, or both. For some cases, lecturer l_k must give an order for students $l_k^s = \{s_{\pi_1}\ s_{\pi_2}\ \dots\ s_{\pi_b}\}$ and projects $l_k^p = \{p_{\pi_1}\ p_{\pi_2}\ \dots\ p_{\pi_c}\}$. In the following we present many ways to rank B_k;

Case I: Lecturers rank B_k respect to both students and projects. In this case each lecturer l_k gives weight to each pair in his B_k and then he orders that pairs respect to their weights not respect to students only or project only. The lecturers rank their preference lists like in instance I_3. Lecturer l_1 mates between student s_3 and project p_1, student s_2 and project p_2, and students s_1 and project p_1. Again he mates between student s_3 and project p_2. Lecturer l_2 mates between student s_2 and project p_3, student s_3 and project p_3, and students s_1 and project p_3. Again he mates between student s_2 and project p_4. In instance I_3 the lecturers order the pairs based on the strong performance of the students in the projects.

Students' preferences

$s_1 : p_2\ p_1$	$l_1^s : s_1\ s_2$
$s_2 : p_1\ p_2$	$l_1^p : p_1\ p_2$

Figure4: Preference lists to create an instance I_4 of SPA-(s, p).

In the next cases during the construction of B_k; the lecturers take into account the preference list of students on projects, or the lecturers' preferences on the projects. Indifferent with the first case where the lecturers give preference list of pairs (student, project) based on their point of view only.

Case II: In this case, each lecturer l_k is working scan of students who accept one of his projects; the lecturer supplies a preference list over these students denoted $l_k^s = \{s_{\pi_1}\ s_{\pi_2}\ \dots\ s_{\pi_b}\}$. The lecturer l_k constructs the preference list L_k as following; he first divide his B_k into b ordered subsets $B_k^{s_{\pi_1}}, B_k^{s_{\pi_2}}, \dots, B_k^{s_{\pi_b}}$ where each $B_k^{s_{\pi_i}}$ is defined as $B_k^{s_{\pi_i}} = \{(s_{\pi_i}, p_j) : (s_{\pi_i}, p_j) \in B_k\}$ (i.e. $B_k^{s_{\pi_i}}$ is the set of all pairs in B_k that contains student s_{π_i}) and $1 \le i \le b$. The preference list L_k is then constructed by concatenating the subsets $B_k^{s_{\pi_i}}$ one after another with respect to the students order in l_k^s. The pairs inside the subsets $B_k^{s_{\pi_1}}, B_k^{s_{\pi_2}}, \dots, B_k^{s_{\pi_b}}$ are ordered lexicographically according to their end point in different ways as the following;

a) There is a symmetrical arrangements between the order of the projects in the student s_{π_i} preference list and the order of the pairs in $B_k^{s_{\pi_i}}$. Figure 4 illustrates this process: Let unordered primary list $B_1 = \{(s_1, p_1)\ (s_2, p_2)(s_1, p_2)(s_2, p_1)\ \}$, B_1 is divided into $B_1^{s_1} = \{(s_1,\ p_2)\ (s_1,\ p_1)\}$ and $B_1^{s_2} = \{(s_2,\ p_1)\ (s_2,\ p_2)\}$ where $B_1^{s_1}$ and $B_1^{s_2}$ are ordered symmetrical the preferences lists of s_1, s_2 respectively. The subset $B_1^{s_1}$ inherits its order from s_1 preference list so lecturer l_k prefers $(s_1,\ p_2)$ to $(s_1,\ p_1)$ because s_1 prefers p_2 to p_1, and the same for $B_1^{s_2}$. Finally place $B_1^{s_2}$ after $B_1^{s_1}$ because lecturer l_k prefers student s_1 to the student s_2 in l_1^s then the lecturer's preferences list is $L_k = \{(s_1,\ p_2)\ (s_1,\ p_1)\ (s_2,\ p_1)\ (s_2,\ p_2)\}$. For any two students $s_i, s_t \in L_k$ the lecturer l_k prefers s_i to s_t iff the lecturer prefers (s_i, p_v) to (s_t, p_u).

b) There is a symmetrical arrangements between the order of the projects in the lecturer l_k preference list and the order of the pairs in $B_k^{s_{\pi_i}}$. Figure 4 gives an example to illustrate these arrangements: Let $B_1 = \{(s_1,\ p_1)\ (s_2,\ p_2)\ (s_1,\ p_2)\ (s_2,\ p_1)\}$, B_1 is divided into $B_1^{s_1} = \{(s_1,\ p_1)\ (s_1,\ p_2)\}$ and $B_1^{s_2} = \{(s_2,\ p_1)\ (s_2,\ p_2)\}$ where $B_1^{s_1}$ and $B_1^{s_2}$ are ordered symmetrical with the lecturers' preferences lists l_1^p over projects, for the subset $B_1^{s_1}$ the lecturer l_1 prefers $(s_1,\ p_1)$ to $(s_1,\ p_2)$ because l_1 prefers p_1 to p_2 in l_1^p, in the subset $B_1^{s_2}$ the lecturer l_1 prefers $(s_2,\ p_1)$ to $(s_2,\ p_2)$ because l_1 prefers p_1 to p_2 in l_1^p. The subset $B_1^{s_2}$ concatenates after $B_1^{s_1}$ because lecturer l_k prefers s_1 to s_2 in l_1^s. Finally the lecturer's preference list is $L_k = \{(s_1,\ p_1)\ (s_1,\ p_2)\ (s_2, p_1)\ (s_2,\ p_2)\}$.

c) For any two pairs contain the same student s_{π_i} the lecturer l_k does not prefer one to other. Returns to instance I_4 in figure 4, let a given unordered list B_1 such that $B_1 = \{(s_1,\ p_1)\ (s_2,\ p_2)\ (s_1,\ p_2)\ (s_2,\ p_1)\}$, then we divide B_1 into $B_1^{s_1} = \{\{(s_1,\ p_1)\ (s_1,\ p_2)\}\}$ and $B_1^{s_2} = \{\{(s_2,\ p_1)\ (s_2,\ p_2)\}\}$ where lecturer l_k indifferent between pairs in the same partition. Finally we we place $B_1^{s_2}$ after $B_1^{s_1}$ because lecturer l_k prefers student s_1 to the student s_2 in l_1^s, then the lecturer's preferences $L_k = \{\{(s_1, p_1)\ (s_1, p_2)\}\ \{(s_2, p_1)\ (s_2, p_2)\}\}$ where $\{(s_1, p_1)\ (s_1, p_2)\}$ means lecturer l_k is indifferent between these two pairs.

Case III: the lecturer supplies a preference list that contains all projects he offers denoted $l_k^p = \{p_{\pi_1}\ p_{\pi_2}\ \dots\ p_{\pi_c}\}$. The lecturer l_k constructs the preference list L_k as following; he constructs $B_k^{p_{\pi_1}}, B_k^{p_{\pi_2}}, \dots, B_k^{p_{\pi_c}}$ where each subset is defined as $B_k^{p_{\pi_j}} = \{(s_i, p_{\pi_j}) : (s_i, p_{\pi_j}) \in B_k\}$(i.e. $B_k^{p_{\pi_j}}$ is the set of all pairs in B_k that contains project p_{π_j}) where $1 \le j \le c$. Then the preference list L_k is constructed by concatenating the partition $B_k^{p_{\pi_j}}$ one after another with respect to the projects' order in the l_k^p list. The pairs inside $B_k^{p_{\pi_j}}$ are ordered lexicographically according to their end point in different ways as the following

a) There is a symmetrical arrangements between the order of the students in the lecturer preference list l_k^s and the order of the pairs in $B_k^{p_{\pi_j}}$. For a given unordered list $B_1 = \{(s_1,\ p_1)\ (s_2,\ p_2)\ (s_1,\ p_2)\ (s_2,\ p_1)\}$, and based on the data in the instance I_4, the lecturer's preference list L_k will be $\{(s_1, p_1)(s_2, p_1)(s_1, p_2)(s_2, p_2)\}$. Any student s_i prefers the project p_u to the project p_r iff the lecturer l_k prefers the pair (s_i, p_u) to the pair (s_i, p_r).

b) For any two pairs contain the same project p_{π_j} the lecturer l_k does not prefer one to other(i.e. for any two pairs $(s_r, p_{\pi_j}), (s_t, p_{\pi_j}) \in B_k^{p_{\pi_j}}$ lecturer l_k does not prefer (s_r, p_{π_j}) to (s_t, p_{π_j}) and vice versa). Returns to instance I_4 in figure 4, let a given unordered list $B_1 = \{(s_1, p_1)(s_2, p_2)(s_1, p_2)(s_2,\ p_1)\}$, then we divide B_1 into $B_1^{p_1} = \{((s_2,\ p_1)\ (s_1,\ p_1))\}$ and $B_1^{p_2} = \{((s_1,\ p_2)\ (s_2,\ p_2))\}$, the lecturer l_1 is indifferent between the pairs that contain the same project. The subset $B_1^{p_2}$ placed after the subset $B_1^{p_1}$ because the lecturer l_k prefers project p_1 to the project p_2 in l_1^p then the lecturer's preference list L_1 is $\{\{(s_2,\ p_1)\ (s_1,\ p_1)\}\ \{(s_1,\ p_2)\ (s_2,\ p_2)\}\}$ where $\{(s_2,\ p_1)\ (s_1,\ p_1)\}$ means that the lecturer is indifferent between these two pairs.

An assignment $M \subset S \times P$ is called match if:

1. $(s_i, p_j) \in M$ Implies that $p_j \in A_i$
2. Each student is assigned to at most one project,
3. Each project $p_j \in P$ is assigned at most c_j students, and
4. Each lecturer $l_k \in L$, supervises at most d_k student.

If M is a match then for any student s we define M(s) to be the project which is applied to s in M and the same for each project p (or lecturer l) we define M(p) (or M(l)) to be the set of students (or pairs) that are assigned to project p (or lecturer l) in M. We say that the project p_j is under-subscribed, full, or over-subscribed if $|M(p_j)|$ is less than, equal to, or greater than c_j, respectively. Similarly, lecturer l_k is under-subscribed, full, or over-subscribed if $|M(l_k)|$ is less than, equal to, or greater than d_k respectively. The pair $(s_i, p_j) \in S \times P \setminus M$ blocks a matching M, where l_k is the lecturer who offers p_j, if:

1. $p_j \in A_i$ (i.e. s_i finds p_j acceptable).
2. Either s_i is unmatched in M, or s_i prefers p_j to $M(s_i)$.
3. Either
   - 3.1 p_j is under subscribed and either
     - (a) $M(s_i) \in P_k$ and l_k prefers (s_i, p_j) to $(s_i, M(s_i))$, or
     - (b) $M(s_i) \notin P_k$ and l_k is under- subscribed, or
     - (c) $M(s_i) \notin P_k$ and l_k is full, and l_k prefers (s_i, p_j) to the worst pair (s_r, p_u) that is being assigned to l_k, or
   - 3.2 p_j is full and l_k prefers (s_i, p_j) to the pair (s_r, p_j), where s_r is the worst student in $M(p_j)$ and either
     - (a) $M(s_i) \notin P_k$, or
     - (b) $M(s_i) \in P_k$ and l_k prefers (s_i, p_j) to $(s_i, M(s_i))$.

We call (s_i, p_j) a blocking pair of M. A matching is stable if it admits no blocking pairs. A student s_i can improve his project to p_j in two cases; first case if s_i is not assigned to l_k and either if there is space in p_j and l_k ,or there is space in p_j but l_k is full and prefers (s_i, p_j) to the worst pair (s_r, p_u) assigned to him. Otherwise, if p_j is full then s_i can be assigned to p_j only when l_k prefers (s_i, p_j) to (s_r, p_j) where s_r is the worst student assigned to p_j. On the other hand, the second case happen when s_i is assigned to project was offered by lecturer l_k. In this case to improve s_i project to p_j, lecturer l_k must prefers (s_i, p_j) to $(s_i, M(s_i))$. That new condition is added to the previous conditions.

III. Student-Oriented Algorithm For SPA-(S, P)

The student-oriented algorithm for SPA-(s, p) is similar to the student-oriented algorithm of SPA [3]. The algorithm is divided into number of passes. Initially, all students are free, and all projects and lecturers are under-subscribed. In each pass, a free student is assigned to the first project in his preference list. This leads to a provisional assignment between students, projects and lecturers, this assignment can be broken later when a project or a lecturer becomes over-subscribed. Also some entries may be deleted from the preference lists of the students, projects, and lecturers when a project or a lecturer become a full. The process DELETE (s_i, p_j) indicates delete p_j from the preference list of s_i, delete s_i from preference list of p_j and delete (s_i, p_j) from the preference list of l_k. For any project p_j offered by lecturer l_k, when project p_j becomes full during the execution of the algorithm it may become under-subscribed again only if l_k becomes over-subscribed and one of his assignments involving p_j is broken. Also, if l_k becomes full during the execution of the algorithm it does not become under-subscribed again. The student-oriented algorithm for SPA-(s, p)-student is an extension of the student-oriented algorithm of SPA [1, 3]. So, this algorithm inherits its correctness, together with the optimality property of the constructed matching from the student-oriented algorithm of SPA with preference list over student.

IV. Data Structures For An Instance Of SPA-(S, P)

The data structure we use is a linked list embedded in an array. We call that array the main array. Each entity in that array consists of a place of data ((student, project) pair), also has six pointers, three next pointers and three previous pointers. These pointers are divided on student, project and lecturer. For each pair (s_i, p_j) in the array, has pointer (next pointer for student) holds the index of the entity that contains the pair(s_i, p_r), where the project p_r is the successor of the project p_j in the student s_i preference list, also there is another pointer (previous pointer for student) holds the index of the pair(s_i, p_v), where the project p_j is the successor of the project p_v in the student s_i preference list. For each student s_i there are another two pointers the first one points to the entity that hold the first project in his list, and the another pointer points to the entity that hold the last project in his list. By these pointers we can travel through the student preference list. In the same way we use the same way to connect all pairs that construct the preference list of any lecturer; also we do the same to present the preference list of each project. In figure 5, an instance of SPA-(s, p) consists of two students and two lecturers and five projects.

Students' preferences Lecturers' preferences

$s_1 : p_1 \ p_5 \ p_2$ $\quad l_1 : (s_1, p_1) \ (s_2, p_3) \ (s_1, p_2) \ (s_2, p_2)$
$s_2 : p_2 \ p_4 \ p_3$ $\quad l_2 : (s_2, p_4) \ (s_1, p_5)$
l_1 offers $p_1 \ p_2 \ p_3$ $\quad l_2$ offers $p_4 \ p_5$

Figure 5: An instance of SPA-(s, p) model.

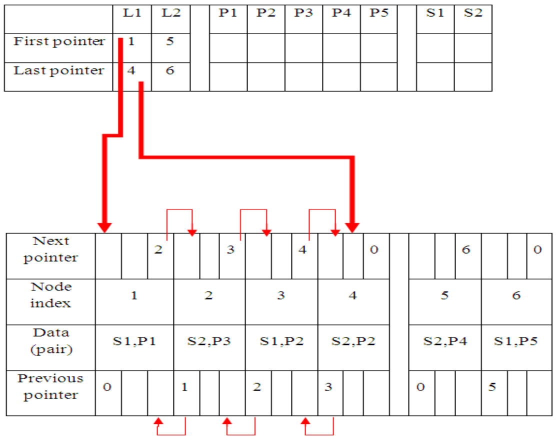

Figure 6: The lecturers' preference lists in the main array and with first and last pointers.

Figure 8 show the data structure that present the given instance in figure 5. First we scan the lecturers' preference lists and put these lists one after another in the main array and we connect these nodes by the next and previous pointer which used for lecturers' lists. When we scan the preference list of each lecturer we connect pairs in his preference lists with pointers (red) as we see in the figure 6, at the end of the scan of each lecturer's preference list we store the indexes of the first pair in his preference list and the last pair in his preference list to be able to travel though his preference list by using these pointers. In figure 6 the first pointer for lecturer l_1 is 1 (first node) the next pair is the second node and it continue to the last node (fourth node) which its index is stored in the last pointer for the lecturer l_1.

While we scan the lecturers' preference lists we can also connect the pairs that contain the same project to construct the projects' preference lists. As we can see in figure 7, the project p_2 has preference list contain two nodes the third node and the fourth node. The first pointer of project p_2 point

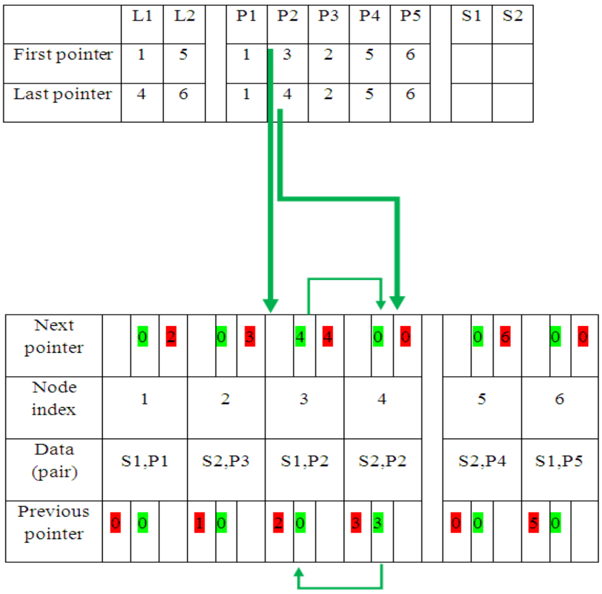

Figure 7: The projects' preference lists in the main array and with first and last pointers.

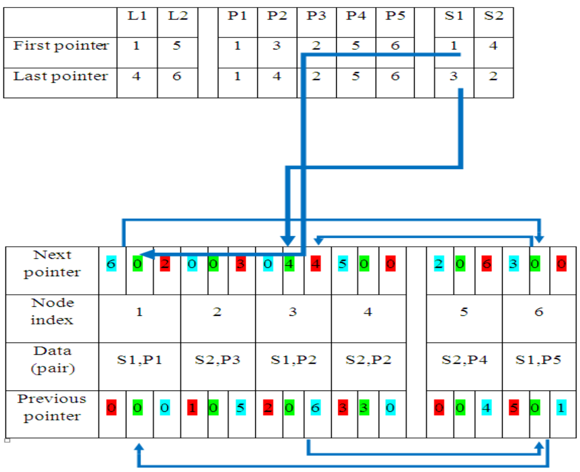

Figure 8: The students' preference lists in the main array and with first and last pointers.

to the third node and the last pointer point to the fourth node and by using these pointers (green) we can walk throw the preference list of project p_2.

After the scan of all lectures' preference lists, we scan preference lists of students to connect the entities in the main array to create the preference lists of the students. We can use a temporary two damnations array to store the indexes of the pairs in the main array when we create the main array during the scan of the lecturers' preference lists. This temporary array will reduce the time of finding the index of any pair in the main array. After we use the temporary array to connect the students' preference lists we delete that temporary array. After we scan preference list of student s_1 we store his first pointer was the index of the node 1 which hold the first project in his preference list and the last pointer for student s_1 holds the index of the node 2 which contains the last project in his preference lists. By using these pointers (blue) we can travel though the preference list of the student s_1 as we see in figure 8. The time of construction is $O(\lambda)$, where the λ is the total length of the preference lists. This data structure reduces the time of deleting or breaking operations, where we only deal with one array and not with three arrays for each preference list. The running time of the algorithm will stay $O(\lambda)$. Also, it reduce the memory space that is needed to represent the preference lists by 1/k where k > 2.

V. SPA-(S, P) STUDENT-ORIENTED VISUALIZATION

This study set out with the aim of assessing the importance of an applet program in the student project allocation problem with preference over pairs. The program starts with a window divided into two parts as shown in figure 9. The first part is a visual panel on which preference lists of students or lecturers are drawn. The second part is utility panel which consists of four buttons and text filed. File problem button is used to display an instance of SPA-(s, p) model sorted in text file. Random problem button is used to create an instance of SPA-(s, p) not sorted in file. Solve button is used to begin solving the instance without stopping. One step button is used to solve one step and stop after that step. In the text area some sentences are written to clarify the current step.

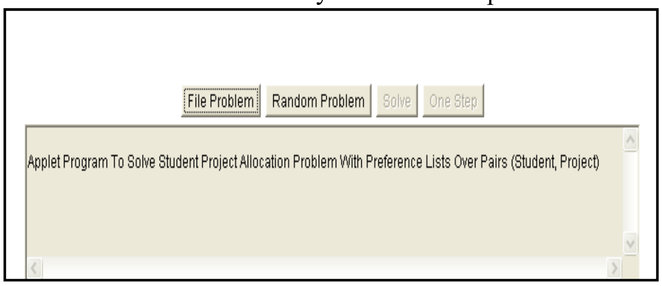

Figure 9: start window in SPA-(s, p) applet program has four buttons

The Creation of the Instances: In the applet program the yellow color means that student is free that is, he has no an assignments with any project that preferred, while the yellow color means that, the lecturers or the projects are under-subscribed. The green color is used to represent a primary assignment between students and projects. When project or lecturer becomes full or over-subscribed they colored orange or red respectively. Clicking the file problem button or the random problem button an instance is drawing on the visual panel, firstly, the students and the lecturers appear on the panel without their preference lists, and projects which are offered by the same lecturer are linked to him by lines. After that each student creates his preference list over projects, and lecturers construct their preference lists over pairs, that lists is displayed on the visual panel.

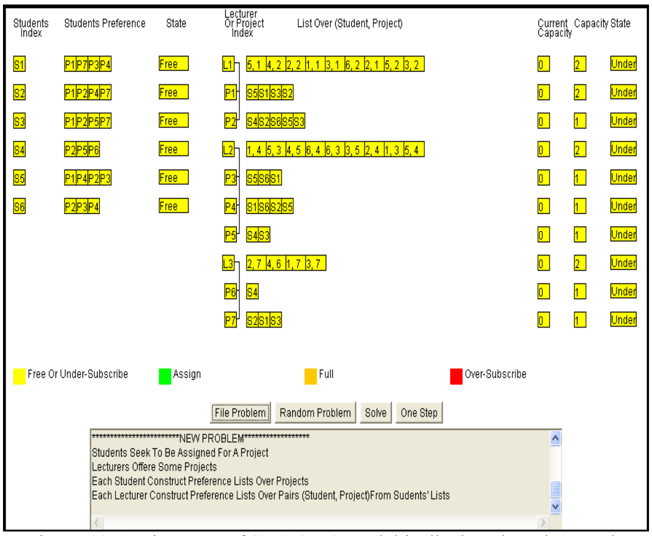

Figure 10: An instance of SPA-(s, p) model is displayed on the panel.

For each project $p_j \in P_k$ a preference list L_k^j is created from the preference list of the lecturer l_k who offers that project. Figure 10 contains a number of columns, the first, named student index which contains a list of students, and to the left there is the second column shows the students' preference lists, followed by the third column on the right refers to the current state of the student in terms of whether it is linked with a project or he is free. The fourth column contains the index of the lecturers and their offered projects, that followed by the fifth column contains the lecturers' preference lists over the student-project pairs. In the far right there are other three columns, one refers to the provisional capacity during the execution of the algorithm and the second refers to the actual capacity of lecturers and projects, the third column, which stands at the extreme right refers to the moment state of the projects and the lecturers in terms of whether they are linked to either one of the students or they are still free.

The execution trace of the Algorithm:

1) Assignment: In this process the student choose the first project in his preference list over projects, this choice results a correlation between student and professor who offered the project, the choice shows through flashing the boxes of the student, the project and the lecturer in the panel and the choice colors these boxes with green color and linking the student to the project by an edge. The flashing refers to the provisional assignment. Finally, states and capacities of the student, the project and the lecturer are updated, see figure 11.

2) Deletion: During the execution of the algorithm, any lecturer or project may become full capacity. In this case, entries are possibly deleted from the students' preference lists, and from the projected preference lists of lecturers. Let (s_v, p_u) is the worst pair assigned to l_k and the successor

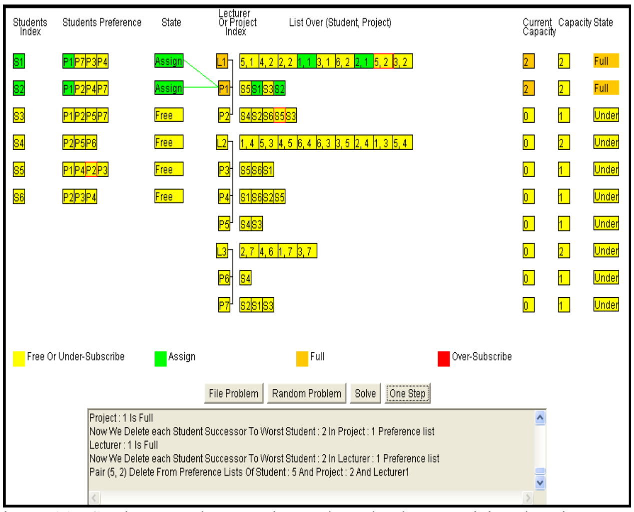

Figure 11: Students apply to projects, these lead to provisional assignments between students and projects.

pair(s_i, p_j) is deleted because l_k becomes full at the same time, the student s_i has to delete from the projected preference list of the lecturer l_k. In this case the applet colors the box that contains the name of this lecturer or the project with orange color and deletes all the unwanted pairs or students from the preference list. In figure 11; the lecturer l_1 and project p_1 become full so their corresponding names are colored orange, the pairs (s_5, p_2) and (s_3, p_2) are the successors to worst pair (s_2, p_1) assigned to lecturer l_1 so this two pairs are deleted from the preference list of lecturer l_1 and the students s_3 and s_5 are deleted from preference list of project p_2 and at the same time project p_2 is deleted from preference lists of those students. Figure 12 shows preference lists after deleting the two pairs (s_5, p_2) and (s_3, p_2) from the lecturer's preference list. Also the students s_3 and s_5 are deleted from the preference list of the project p_2 and the project p_2 is deleted from preference lists of those students.

3) Break: A free student is assigned to the first project in his preference list. This leads to a provisional assignment between students, projects and lecturers, this assignment can

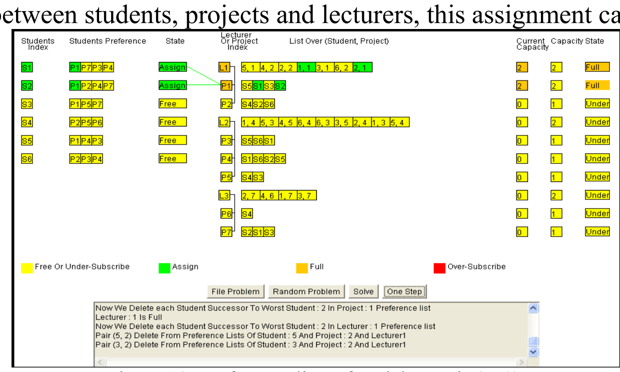

Figure 12: preference lists after delete pair (5, 2)

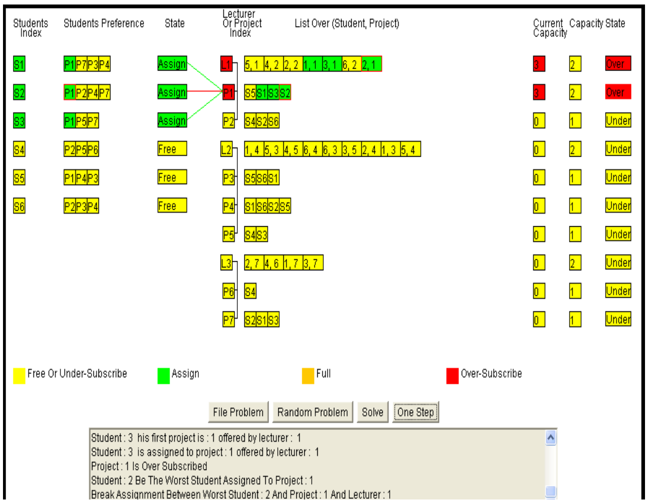

Figure 13: lecturer l_1 and project p_1 are over-subscribed, the worst student s_2 that was assigned to the project p_1 is selected and the edge between them is flashing.

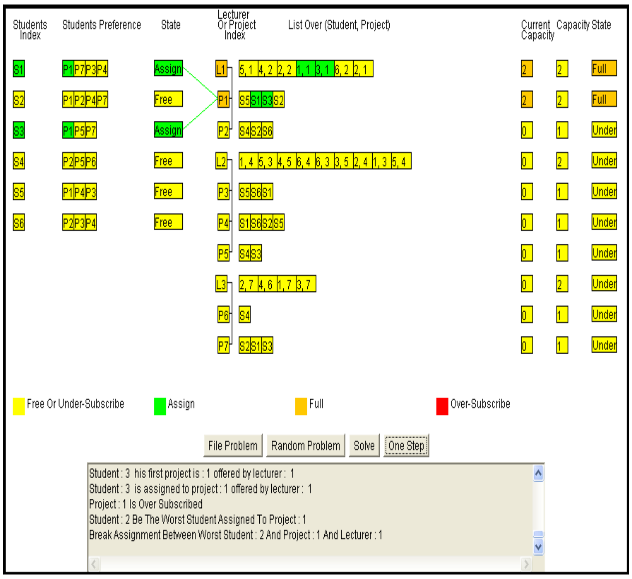

Figure 14: preference lists after break assignment between the student s_2 and the project p_1.

Figure 15: the stable matching.

be broken later when a project or a lecturer becomes over-subscribed and their boxes are colored red. As figure 13shows, the project p_1 becomes over-subscribed and the student s_2 is selected to break his assignment to the project p_1 at this moment the edge between them is flashing before breaking the assignment between the student s_2 and the project p_1. Figure 14 show the preference lists after breaking the assignment between the student s_2 and the project p_1.

Each iteration loop includes a free student applying to the first project on his/her preference list over the projects. After a number of iterations bounded by the overall length of the student preference lists, each student is assigned to at most one project and the assigned pairs constitute the stable match. The stable match is displayed as green boxes in the panel, the stable match is written in the lower part of Figure 15, the name of the student and the name of the best possible wishes project in his/her preference list.

VI. Conclusion

This paper has given an account of and the reasons for the efficient use of SPA-(s, p) model compared to previous models SPA and SPA-p. The present study was designed to determine the effect of the use of preference lists over pairs. One of the more significant findings to emerge from this study is that; SPA-(s, p) gives a larger stable matching. The second major finding was that the SPA-(s, p) is the senior of the two student project allocation models SPA and SPA-p. Part of our results had been published in [11].

References

[1] D.J. Abraham. Algorithmics of two-sided matching problems. Master’s thesis, University of Glasgow, Department of Computing Science, 2003.
[2] D.F. Manlove and G. O’Malley. Student project allocation with preferences over projects. In Proceedings of ACID2005:the 1stAlgorithms and Complexity in Durham workshop, volume 4 of Texts in algorithmics, pages 69- 80. KCL Publications, 2005.
[3] Abraham D.J and Irving R.W. and Manlove D.F. Two algorithms for the student- project allocation problem. Journal of Discrete Algorithms 5(1):pp. 73-90, 2007.
[4] M. Thorn, A constraint programming approach to the student-project allocation problem, BSc Honours project report, University of York,Department of Computer Science, 2003.
[5] C.Y. Teo, D.J. Ho, A systematic approach to the implementation of final year project in an electrical engineering undergraduate course, IEEETransactions on Education 41 (1) (1998) 25–30.
[6] J. Mestre, Weighted popular matchings, in Pro. of ICALP 2006, the 33rd Inter-national Colloquium on Automata, Languages and Programming, vol. 4051 of LNCS, pp 715–726, 2006.
[7] C. Shafer, Heath and J. Yang, "Using the Swan Data Structure Visualization system for Computer Science Education", Proceedings of the SIGCSE, ACM Press, 1996, pp. 140-144.
[8] H. Biermann, R. Cole, (1999), “Comic Strips for Algorithm Visualization”, NYU, New York, NY, Tech. Rep. 1999-778
[9] Charalampos Papamanthou, Konstantinos Paparrizos, and Nikolaos Samaras. “A Parametric Visualization Software for the Assignment Problem”. Yugoslav Journal of Operations Research, 15(1):147-158, 2005.
[10] M.D. Byrne, R. Catrambone and J. T. Stasko, “Evaluating animations as student aids in learning computer algorithm”, Computers and Education 33 (1999) 253-378.
[11] Mahmoud I. Moussa and Ahmed H. Abu El-Atta, “Student Project Allocation with Preference Lists over (Student, Project) Pairs”, The 2nd International Conference on Computer and Electrical Engineering (ICCEE 2009), Dubai, UAE.