

Constraints Aware and User Friendly Exam Scheduling System

Mohammad Al-Haj Hassan¹ and Osama Al-Haj Hassan²

¹Computer Science Department, Zarqa University, Jordan

²Computer Science Department, Isra University, Jordan

Abstract: Scheduling is a crucial task for schools, universities, and industries. It is a vital task for any system containing utilization of resources to fulfill a certain criterion. Utilization of such resources usually includes several conflicting constraints that scheduling has to take into account. Exam scheduling is an essential key for schools and universities in order for exams periods to be smooth. In this paper, we present an exam scheduling system that employs graph coloring scheduling technique. We focus on two aspects: First, the constraints our system handles, second, the user friendly interface of the system.

Keywords: Exam scheduling, user friendly, constraints, optimization, conflict, graph coloring.

Received September 16, 2015; accepted October 18, 2015; published on line January 28, 2016

1. Introduction

Scheduling is needed in various aspects of life such as reservations, project scheduling, timetabling, workforce scheduling, appointments, transportation scheduling, and scheduling in entertainment [10]. It is also a necessity in schools and universities in order to generate exam schedules [2].

The process of generating exam schedules is not a straight forward one. There are many constraints that should be taken into consideration such as available instructors in a time period, available halls and labs, number of concurrent exams. In fact, finding the optimal exam schedule that satisfies given constraints is considered NP-Hard problem [3].

There are many well known scheduling techniques, such as graph coloring [8, 12], fuzzy logic [2], simulated annealing [5], particle swarm [6], genetic algorithms [4], memetic algorithms [7], and ant colony [14]. The one we use in our system is a graph coloring scheduling algorithm which we already proposed in a previous work [8].

The contribution of this paper is to design exam scheduling system that embodies the following: First, our exam scheduling system covers as many constraints as possible which make our system generate accurate exam schedules; second, the design of our system is user friendly which makes it easy to understand and use by non tech savvy people.

The rest of the paper is organized as follows: First, in section 2 we discuss related work in scheduling research area. After that, in section 3 we comprehend on the various constraints we take into consideration in the exam scheduling process. Next to that, in section 4 we discuss the graph coloring algorithm we use in our system. Consequent to that, we propose in section 5 the

requirements of scheduling exams in real world scenarios. Following that, we present our design for the exam scheduling system in section 6. Then, we conclude our work in section 7.

2. Literature Review

Exam scheduling is a form of time tabling problem and it has been studied extensively in literature. Selemani *et al.* [12] use the widely known Recursive Largest First (RLF) algorithm to color a graph that represents different sections in Sokoine University of Agriculture. The work in [6] presents a survey of different particle swarm techniques for solving exam scheduling problem. Hosny and Al-Olayan [4] use a genetic algorithm to generate exam schedules. They consider two dimensional chromosome consisting of days as one dimension and exams as another dimension. The genetic algorithm they use relies on mutation operator and excludes crossover operator. The work in [14] uses an ant colony approach to generate exam schedules. The approach relies on constructing an initial solution comprising days, rooms, slots, and exams. Then, exams schedule is developed by tracking pheromone of ants trying to make a tour to find optimal exam schedules. In [13], a schedule is generated by searching among heuristics and this is achieved by using iterative local search and a set of move operators that tend to improve the quality of the outcome schedule. A survey of different exam scheduling techniques can be found in [1, 9]. A clonal selection algorithm that produces exam schedules is proposed in [14], where in this work, a set of solutions (antibodies) are developed and the affinity (fitness) of those solutions are calculated based on a fitness function. After that, the fittest antibodies are chosen to be cloned

with a certain degree of mutation in order to find better solutions. Sabar *et al.* [11] discuss a honey-bee mating optimization algorithm which is used to find near optimal exam schedules. The algorithm relies on queen (current best solution), drones (trial solutions), workers (heuristics), and brood (new solutions). The algorithm first generates a pool of solutions where the best one is chosen as the queen and the others are considered drones. Drones (trial solutions) mate with the queen (current best solution) using crossover and that generates new solutions.

3. Exam Scheduling Variables and Constraints

Our system takes into consideration several variables and constraints. This is of utmost importance so that the generated schedule meets the operation of real world scenarios. The following is a list of those variables and constraints:

- **Count of Days:** The count of days allowed for exam scheduling. This can be a specific number or it can be open such that our system uses the minimum number of days needed to generate a schedule.
- **Count of Time Slots:** The count of time slots during which exams can be scheduled.
- **Type of Exam:** The type of exam such as first, second, mid, or final exams.
- **Concurrent Exams:** A student cannot have more than one exam in the same time slot of a given day.
- **Count of Exams for Students:** The maximum count of exams held in the same day for one student.
- **Exam Position:** Indicates whether the system has to schedule exams using predefined day and time rules or the system has the freedom to schedule the exam in any day and time slot.
- **Conflicts:** Sometimes when a fixed number of days are specified, conflicts may arise such as count of exams in one day for a given student exceeds the allowed limit. So, this parameter indicates if a conflict is allowed. If conflicts are not allowed (Hard Constraint), then few exams may remain unscheduled.
- **Exams per Time Slot:** The maximum number of exams that can be scheduled in a given time slot.
- **Monitoring Tasks:** The maximum count of monitoring tasks that can be assigned to an instructor.
- **Concurrent Monitoring:** An instructor cannot have two simultaneous monitoring tasks.
- **Concurrent class with monitoring:** An instructor cannot have a monitoring task at the same time of his a class he or she teaches.
- **Concurrent Class with Student Exam:** A student cannot have an exam at the same day and time of a class he attends unless the exam pertains to that particular class.

- **Concurrent Exams:** An instructor cannot have two simultaneous exams. The same constraint applies for students.

It is worth mentioning that most of the above constraints are parametric. Only constraints that constitute predefined conditions will be imposed on our system rather than considering them parameters.

4. Exam Scheduling using Graph Coloring

Our exam scheduling system is based on our work in [8], wherein a novel technique for exam scheduling using graph coloring is proposed. In that work, we represented exam scheduling problem as an undirected weighted graph G that is an ordered pairs (V, E, W) , where V is graph nodes, E are edges between nodes, and W is a weight function that gives weight to edges. Here, a node corresponds to a section of a course and an edge between two nodes, together with its weight, pertains to number of common students between the two sections. Adjacent nodes are sections that share an edge with weight (number of common students) greater than zero. Example of an exam scheduling problem represented in a weighted undirected graph is shown in Figure 1.

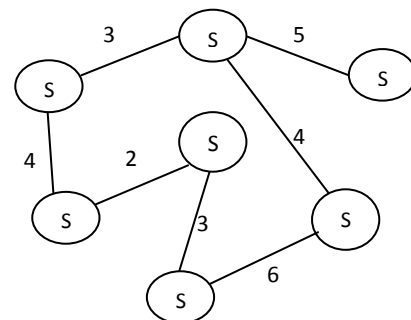


Figure 1. Sections represented as a graph.

Colors indicate available time slots in a given day. A color has concurrency limit which represents the number of exams that can concurrently be held at that time slot. This is usually controlled by the number of available halls/ labs in that time slot. For example an instance of a color is time slot (09-10) and concurrency limit for this color is 5 meaning that there are 5 available halls/ labs in time slot (09-10). The graph coloring problem is concerned with coloring graph G such that no two adjacent nodes have the same color. This is logical because adjacent nodes have students in common, and therefore, cannot be scheduled in the same day and time slot. The graph coloring algorithm starts by building adjacency matrix of sections as shown in Table 1. An entry in the adjacency matrix corresponds to count of common students between the two sections. Each section has a degree and a weight values. A degree of a section "S" refers to the count of sections with which the section "S" shares edges. A weight of a section "S" is the summation of weight

values on the edges “S” shares with other sections. Table 2 shows corresponding degrees and weights for the sections in Figure 1. The next step is to order sections list “secList” in descending order based on degree which is shown in Table 3. Now, some sections might share the same degree, therefore we order them in descending order based on their weight as illustrated in Table 4. After that we iterate over each section “S” in the ordered list such that the following steps are executed for each section:

- Find the first day and slot that does not violate the constraints listed in section 3. Assign that time slot to the section “S”.
- Find the adjacency list “adjS” of the section “S”.
- Order the sections of “adjS” based on the same degree and weight ordering explained previously.
- For each section “S2” in “adjS”, find the first day and slot that does not violate constraints listed in section 3. Assign that time slot to the section “S2”.

The graph coloring algorithm is shown in Algorithm 1.

Table 1. Adjacency matrix of graph in figure 1.

	S1	S2	S3	S4	S5	S6	S7
S1	0	2	4	0	0	0	0
S2	2	0	0	0	3	0	0
S3	4	0	0	3	0	0	0
S4	0	0	3	0	0	5	4
S5	0	3	0	0	0	0	6
S6	0	0	0	5	0	0	0
S7	0	0	0	4	6	0	0

Table 2. Degrees and weights of sections in figure 1.

	S1	S2	S3	S4	S5	S6	S7
Degree	2	2	2	3	2	1	2
Weight	6	5	7	12	9	5	10

Table 3. Sections of table 2 ordered based on degree.

	S4	S1	S2	S3	S5	S7	S6
Degree	3	2	2	2	2	2	1
Weight	12	6	5	7	9	10	5

Table 4. Sections with the same degree ordered based on weights.

	S4	S7	S5	S3	S1	S2	S6
Degree	3	2	2	2	2	2	1
Weight	12	10	9	7	6	5	5

5. Real World Scenario Requirements

In this section, we take the graph coloring algorithm mentioned in [8] as a base for our system that takes into consideration real world scenarios. In the original algorithm, conflicts are not allowed. However, in real world scenario, the school/university may force a requirement for the count of days in the schedule. In this case, conflicts may occur such as a student having count of exams in a given day greater than the allowed number. In original algorithm, the type of exam is really a general concept. In real world scenario, there are specifics that may differ according to type of exam. For example, during first, second, and mid exams, scheduling an exam during the same day and time slot

in which the section is taught is considered normal. This is not normally true for final exams where fixed time periods are available and no classes are held. Another example is related to hall availability. In final exams all halls are available while in first, second, and mid exams some halls are already occupied with classes. In the original algorithm, the concurrency limit is a general concept which means the count of exams that can be held in a given day and slot. This is normally translated to the count of available halls in that day and time slot. However, in real scenarios we might consider a case where multiple exams can be held in the same hall; and this redefines the concurrency limit definition. In our original work, we have not discussed any constraints related to assigning exam monitoring tasks for instructors. This is definitely a needed issue in real world scenarios. In the original work all sections are considered unique entities. However, in real scenarios we have “Shared Sections” which arise because of changes on degree requirements. When a major change occurs to degree requirements, this change has to be applied on new students. But, it cannot be applied on previous students who are committed to the previous degree plan. These results in having two or more sections that are assigned different course number and/or different section number because they belong to different degree plans. However, those sections are really the same unique section. So, for exam scheduling purposes, those sections have to be treated as one. One point to mention is that in real world scenarios, an instructor might request that his exam be held in a lab instead of a theory hall. This is not mentioned in our original work.

Algorithm 1: The graph coloring algorithm.

```

Construct Adjacency Matrix of sections in the section list
“secList”
Order sections of “secList” in descending order based on
degree
For sections in “secList” with the same degree
Order them in descending order based on weight
End For
For each section “S” in the ordered list “secList”
Assign to “S” the first day and time slot such that constraints
are not violated
Retrieve “adjS” which is the adjacency list of “S”
Order sections of “adjS” in descending order based on degree
For sections of “adjS” with the same degree
Order them in descending order based on weight
End For
For each section “S2” in “adjS”
Assign to “S2” the first day and time slot such that constraints
are not violated
End For
End For
    
```

6. Exam Scheduling System Design

In this section, we shed light on the design and features of our system. Our system is developed using Java. It interacts with a MySQL database which stores information of sections being taught in a given semester. The database contains the following tables:

- Course: Represents courses to which sections belong.
- Section: Represents sections of courses.
- Instructor: Represents teachers of sections.
- Student: Represents students who register sections.
- Hall: Represents halls and labs.

When the user runs the system, data is loaded from database. The main window has several menus and it looks like Figure 2.

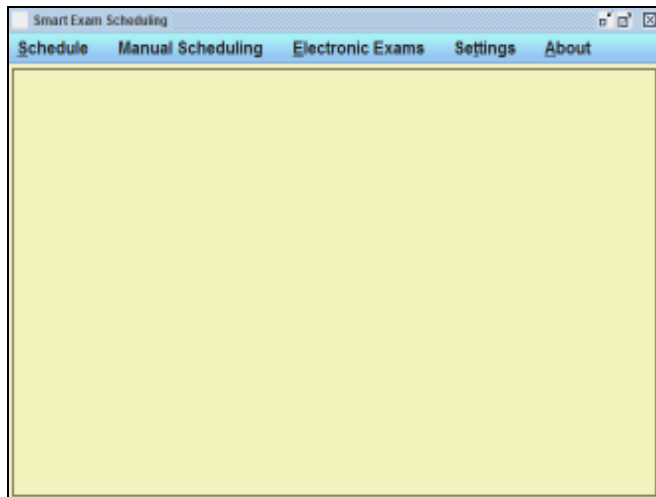


Figure 2. Scheduling system main window.

First, “Settings” menu enables user to control several parameters in the scheduling process. One of the menu items in the settings menu is “Parameters” which opens the window illustrated in Figure 3. It allows the user to enter maximum number of days allowed for schedule, maximum number of exams that can be held in a given time slot, maximum number of exams for a student in one day, and maximum number of monitoring tasks permissible for an instructor. The user can also select whether the desired schedule is with minimum number of days such that it contains no conflicts or is strict to the entered maximum number of days regardless of having conflicts. Also, the user can choose if all sections of a given course are to be scheduled in the same time slot or each section is scheduled on its own slot. In addition, the user can determine the type of exam such as first, second, mid, or final exam. Finally, scheduling can occur based on fixed rules coming from the registration department. These rules come in the form “A section that is taught on a given day and time would be scheduled in a given day and time”. On the contrary, the user can choose “dynamic” scheduling that uses the graph coloring algorithm explained in this paper to generate a schedule with fewest conflicts. The second option in settings menu is “Schedule Days”. Here, the user can choose the exact dates of schedule days. This can be done by selecting the date of the first day from a calendar and then clicking “Change Dates” which changes the dates of the remaining days accordingly.



Figure 3: Parameters window.

In addition, the user can use a calendar beside each day to change the date of that day. This is shown in Figure 4.

After the user finishes choosing settings, he or she can go through the process of electronic exams using the menu “Electronic Exams”. Clicking on the single option available in that menu causes the window in Figure 5 to show up. This window contains two lists. The list on the left contains sections that are originally taught in a hall, the list on the right contains sections that are originally taught in a lab. The user can use the two buttons to move sections between the two lists: This is handy in case we need to schedule an exam in a lab when it is originally taught in a hall and vice versa. The user can also manually schedule sections. This can happen when for example an instructor of a given section presents a special request to schedule one of his exams in a given day and time due to personal or urgent circumstances. This can be done by using “Manual Scheduling” menu which shows the window in Figure 6. The window contains two lists: The list to the left contains unscheduled sections and the list to the right contains scheduled sections, the second list is usually empty unless sections are manually scheduled which causes them to move to the list on the right. To manually schedule a section, the user selects it from list to the left and clicks on the button with right arrow. This opens a window showing schedule days, time slots, and available halls/ labs in that day/ slot. The user chooses the desired parameters and clicks “OK”. This causes the section to be manually scheduled and it will be transferred to the menu on the right.

If a scheduled section on the right list should be moved back to the unscheduled sections list, the section can be selected and the button with left arrow is clicked.

After settings are selected, electronic exams are chosen, manual scheduling is performed. The user is now ready to generate a new schedule using “Schedule” menu, the first option in this menu is “Generate New Schedule”, and clicking this option

causes the scheduling algorithm to run. The algorithm will take into consideration the settings, electronic exams, and manual scheduling previously chosen and then a new schedule are generated.

provides is the count of students having one, two, and three exams per each day. This gives an indication about quality of generated schedule since students having three exams in one day is normally not allowed and students having two exams in the same day should be kept as minimum as possible. But cases like that can happen if the user selected the settings that force the system to work within very few days which causes three exams issue to arise. In this case, students usually request deferring one of the three exams. This piece of information also gives an indication of how busy a building where exams are held during a given day. The second option in “Schedule” menu is “Store Schedule as HTML File” which causes generated schedule to be stored in a readable user friendly way. Examples of stored schedules are illustrated in Figures 7 and 8. It is clear that Figures 7 and 8 contain just parts of lengthy schedules as test samples.

The previous description shows the details of our scheduling system and the steps the user undertakes in order to generate a new schedule. We showed the aspects related to offering a flexible easy to follow and user friendly exam generation process.



Figure 4. Days control window.

One piece of information that the system

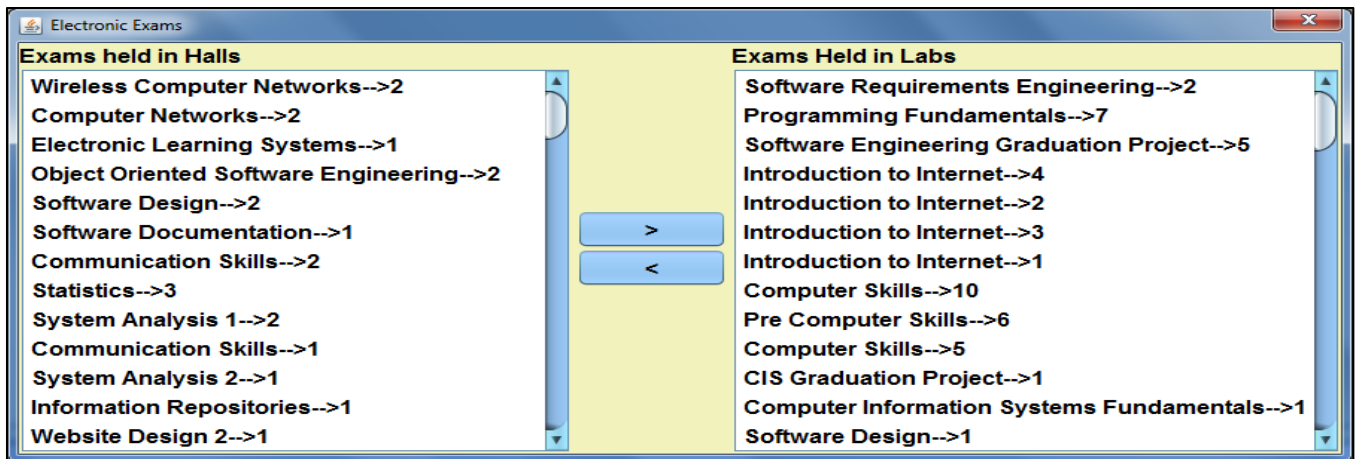


Figure 5. Electronic exams and paper exams.

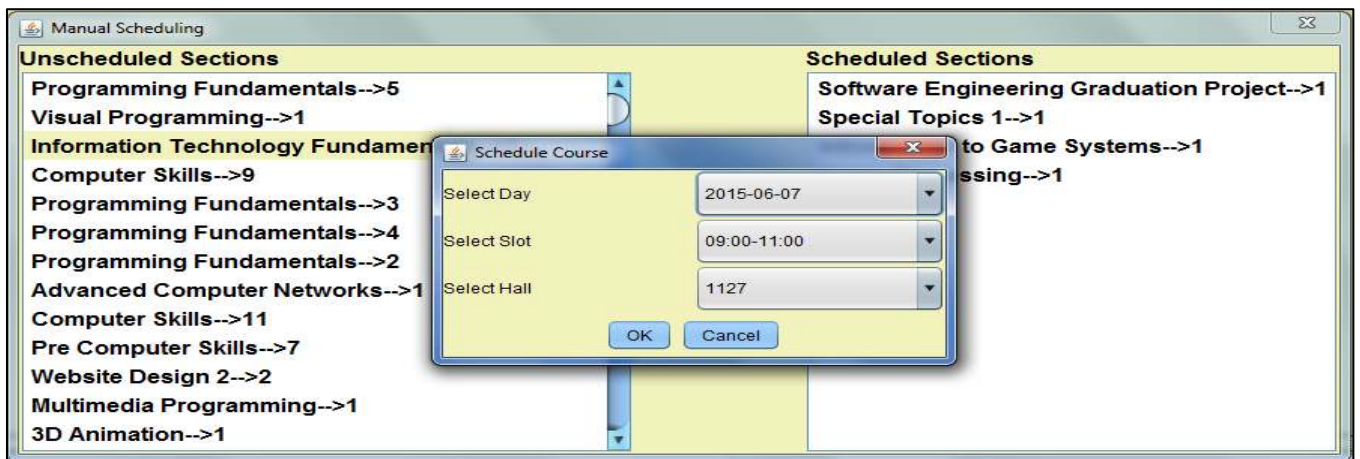


Figure 6. Manual scheduling.


Faculty of IT				Monday 7-12-2015			
Second Exam Schedule for 2015/2016 Semester							
Morning							
Day	Time	Course Number	Course Name	Section Number	Place	Student Count	Instructor/Monitor
Sunday 20-12-2015	11:00-12:00	1501321	Algorithms Design	1	H310	20	Mohammad Al-Haj Hassan
		1501112	Computer Programming 2	1	H312	17	Khaled Waleed
		1501110	Computer Programming 1	1	H313	47	Mohammad Al-Ghanem
		1504180	Web Programming 1	1	H315	14	Mohammad Rasmi
		1501230	Logic Design	1	H311	37	Yaser Al-Laham
		1501220	Discrete Mathematics	1	H316	15	Rozain Qadoura
	12:00-01:00	1501221	Data Structures	1	H311	20	EmanMashaqbah
	01:00-02:00	1501310	Advanced Programming	1	H313	30	Mohammad Al-Kaabi

Figure 7. Example of stored schedule.


Faculty of IT				Saturday 12/9/2015			
First Exam Schedule for 2015/2016 Semester							
Morning Session							
Day	Time	Course Number	Course Name	Section Number	Students Count	Place	Instructor/Monitor
Sunday 07/06/2015	9:00-11:00	603323	Software Engineering Tools	2	25	3310	Khalil Barhoom
		604371	Operations Research	1	9	4135	Suha Afaneh
		605242	Computer Architecture	3	22	4136	Abdulla Al-Ali
		603210	Software Engineering Fundamentals	1	18	4138	Nabeel Abu Hasheeh
		601321	System Analysis 2	1	12	4140	Bassam Sharjabi
		603441	Fault Tolerant Software	1	4	4142	Majdi Ghobn
		605481	Graphic Design	1	15	1127	Faris Abu Hashish
		602283	Graphic Design	1	15	4003	Khaled Kaabneh
		608314	Advanced Computer Networks	1	5	4315	Anas Abu Taleb
		605116	Programming Fundamentals	5	23	7209	Wael Toughoj
		605242	Computer Architecture	1	24	4135	Abdulla Al-Ali

Figure 8. Example of stored schedule.

7. Conclusions

In this paper a new exam scheduling system is proposed. The system covers several key constraints related to schedule days, schedule time slots, conflicts, students, and instructors. The system design is user friendly which allows users to generate a schedule in a flexible and easy process. Our system utilizes a graph coloring scheduling algorithm which provided a strong base for generating satisfactory exam schedules.

References

[1] Babaei H., Karimpour J., and Hadidi A., "A Survey of Approaches for University Course Timetabling Problem," *Computers & Industrial Engineering*, vol. 86, pp. 43-59, 2015.

[2] Cavdur F. and Kose M., "Fuzzy Logic and Binary-Goal Programming-Based Approach for Solving the Exam Timetabling Problem to Create a Balanced-Exam Schedule," *International Journal of Fuzzy Systems*, pp. 1-11, 2015.

[3] Gonsalves T. and Oishi R., "Artificial Immune Algorithm for Exams Timetable," *Journal of*

Information Sciences and Computing Technologies, vol. 4, no. 2, pp. 287-296, 2015.

[4] Hosny M. and Al-Olayan M., "A Mutation-Based Genetic Algorithm for Room and Proctor Assignment in Examination Scheduling," *Science and Information Conference*, pp. 260-268, 2014.

[5] Kalender M., Kheiri A., Ender A., and Burke E., "A Greedy Gradient-Simulated Annealing Selection Hyper-Heuristic," *Journal of Soft Computing*, vol. 17, no. 12, pp. 2279-2292, 2013.

[6] Larabi S. and Sainte M., "A Survey of Particle Swarm Optimization techniques for Solving University Examination Timetabling Problem," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 537-546, 2015.

[7] Lei Y., Gong M., Jiao L., and Zuo Y., "A Memetic Algorithm Based on Hyper-Heuristics for Examination Timetabling Problems," *International Journal of Intelligent Computing and Cybernetics*, vol. 8, no. 2, pp. 139-151, 2015.

[8] Malkawi M., Al-Haj Hassan M., and Al-Haj Hassan O., "A New Exam Scheduling Algorithm Using Graph Coloring," *International Arab Journal of Information Technology*, vol. 5, no. 1, pp. 80-86, 2008.

- [9] Pillay N., "A Survey of School Timetabling Research," *Annals of Operations Research*, vol. 218, no. 1, pp. 261-293, 2014.
- [10] Pinedo M., Zacharias C., and Zhu N., "Scheduling in the Service Industries: An Overview," *Journal of Systems Science and Systems Engineering*, vol. 24, no. 1, pp. 1-48, 2015.
- [11] Sabar N., Ayob M., and Kendall G., "Solving Examination Timetabling Problems using Honey-Bee Mating Optimization (ETP-HBMO)," in *Proceedings of Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, Dublin, Ireland, pp.399-408, 2009.
- [12] Selemeni M., Mujuni E., and Mushi A., "An Examination Scheduling Algorithm using Graph Colouring-The Case of Sokoine University of Agriculture," *International Journal of Computer Engineering & Applications*, vol. 3, no. 1, pp. 116-127, 2013.
- [13] Soria-Alcaraz J., Ochoa G., Swan J., Carpio M., Puga H., and Burke E., "Effective Learning Hyper-Heuristics for the Course Timetabling Problem," *European Journal of Operational Research*, vol. 238, no. 1, pp. 77-86, 2014.
- [14] Thepphakorn T., Pongcharoen P., and Hicks C., "An Ant Colony Based Timetabling Tool," *International Journal of Production Economics*, vol. 149, pp. 131-144, 2014.



Mohammad Al-Haj Hassan is a professor of Computer Science since the year 2002. He obtained his BSc and MSc from The University of Jordan in 1973 and 1977, respectively, and his PhD degree in computer science from Clarkson University at NY, USA, in 1983. His research interests include: Computer algorithms and applications, graph algorithms and applications, natural language processing, and machine learning. He worked in several universities inside and outside Jordan both in academic and administrative positions, and he has more than 35 publications and authored books.



Osama Al-Haj Hassan is an assistant professor of computer science. He has a BSc in computer science from Princess Sumaya University of Technology in 2002. He finished his master degree in computer science from New York institute of technology in 2004. He earned his PhD degree in computer science from University of Georgia/ USA in 2010. His research interests are in the areas of distributed systems, web 2.0, mashups, web services, and peer-to-peer networks.