Journal of Computer Science

Original Research Paper

A Developed Uncapacitated Scheduling Algorithm of Building Timetables for Different Exam Kinds

Hussein Al Bazar and Hussein Abdel-Jaber

Faculty of Computer Studies, Arab Open University, Dammam, Kingdom of Saudi Arabia

Article history
Received: 14-06-2020
Revised: 05-08-2020
Accepted: 22-08-2020

Corresponding Authors:
Hussein Al Bazar
Faculty of Computer Studies,
Arab Open University,
Dammam, Kingdom of Saudi Arabia
Email: halbazar@arabou.edu.sa

Abstract: Scheduling many exams into different timeslots and rooms while meeting all given requirements is a difficult task, which requires designing and proposing a scheduling algorithm to deal with the issue of timetabling examination. This paper proposes a scheduling algorithm for tackling such an issue. The proposed scheduling algorithm generates a satisfactory timetable of different module exams derived from different faculties such as the faculty of language studies, business studies and computer studies within a university. The generated timetable satisfies all given constraints such that no student shall attend more than one exam at the same day and time. Moreover, a user interface system is developed in order to enable different selections, for instance, exam sort, the number of sessions per day. Further, it can show the timetable of different exams of a module. The accuracy of every examination session and all sessions is calculated and it is found to be proven from the results the satisfaction of each examination session and all sessions together is increased.

Keywords: Scheduling Algorithm, Fitness Function, User Interface System, Examination Timetable, Hard and Soft Constraints, Uncapacitated Scheduling

Introduction

The problem of timetabling of different entities have shown interest in many areas (e.g., people, academic institutions, factories, machines, sports and so on). Timetabling allocates a number of entities into different timeslots and places by taking into account a bundle of constraints; including hard and soft constraints (Vatansever and Arici, 2019). For instance, assigning the exams of academic institutions into a number of timeslots and rooms based on available resources and meeting the given constraints. In a university, there are usually several faculties. In each faculty, there are several departments and in each department, many students can register and enrol for different modules. As a result, it is significant to create a number of schedules for students, such as a student's timetable, the timetable of examinations and so forth. When the number of students that is enrolled for different modules increases, the difficulty of building timetables increases, particularly, building timetables for examinations. Moreover, when the number of hard and soft constraints is enormous and limited available resources emerge, building a timetable is considered a difficult task. Hard and soft constraints should be met in order to build a quality timetable. Hard constraints differ from one academic institution to another. When building an examination timetable, hard constraints should be met. Examples on such constraints are presented as follows:

- Exams should be allocated to particular given timeslots such that these timeslots are limited (hard constraint)
- A student can only sit for a single exam within every timeslot (hard constraint)
- The module's exams of the same level are scheduled within the same track into different days unless some other modules are considered as perquisites for certain modules (hard constraint)

Soft constraints are required, but are not crucial (Qu *et al*., 2009) and are not guaranteed to be met due to the difficulty of exams in the scheduling problem. Thus, a number of soft constraints is breached. Additionally, the differences of soft constraints emerge from one institution to another according to their necessity and sort (Burke *et al*., 1995) and these constraints are conflicting with each other. This can enable students to obtain sufficient time for studying their exams. Examples on soft constraints are given by (Qu *et al*., 2009) as follows: The conflicting exams are equally distributed as large as possible and this enables students to obtain sufficient time to study. The

DOI: 10.3844/jcssp.2020.1139.1149

exams should be held on the same day, at the same time or in one place. Further, exams should be successive where the biggest exams should be held on earlier dates. The priority of exams is required to be satisfying and for every timeslot, a specified number of exams and/or students is achieved. Conditions of time such as exams not need to be in particular timeslots. In the same day, conflicting exams should be allocated close to each other by dividing exams on analogous places. The solely exams that share the same time interval can be held within the same room. Requirements of resource such as rooms.

Creating an examination timetable is not a simple and easy task since building an optimal schedule by satisfying all given constraints is presented as an NP-Hard problem (Gonsalves and Oishi, 2015). The problem of examination timetabling is introduced into two different types: Incapacitated and capacitated. The room capacities are not assumed in the incapacitated type, while the capacity of rooms are assumed as a hard constraint in the capacitated type.

There is a number of heuristic processes for solving the issues of the timetabling examination. Such processes include the different scheduling methods, which comprise: Genetic Algorithms (GAs) (Ross and Corne, 1994; Hosny and Al-Olayan, 2014; Dener and Calp, 2018), Ant Colony (Thepphakorn *et al*., 2014), Simulated Annealing (Kalender *et al*., 2013; Kirkpatrick *et al*., 1983), FastSA that is a new variant of simulated annealing algorithm (Leite *et al*., 2019), Tabu Search algorithms (Clark, 1993; Hertz, 1991), Graph Coloring Techniques (Selemani *et al*., 2013; Malkawi *et al*., 2008; Carter, 1986; Abou Kasm *et al*., 2019), Fuzzy Logic (Cavdur and Kose, 2016), Memetic Algorithm (Lei *et al*., 2015; Burke *et al*., 1996), Particle Swarm (Marie-Sainte, 2015), automatic scheduling algorithm depending on hash and priority (Ji and Ma, 2020) and many more. Moreover, (Novita *et al*., 2020) build a computer application that aims to establish a schedule of different exams at the State Polytechnic of Sriwijaya (Carter and Laporte, 1996). Novita *et al*. classify the examination timetabling issue into four different groups, which are: Classification approaches, sequential approaches, generalised search techniques and constraint satisfaction methods. Some of these methods relate to the domain of GAs, Simulated Annealing and Tabu Search. However, the problems can be described according to a number of students that have module's exams within the same days and times and according to the importance of managing such exams within the same level when needed.

This paper attempts to solve the following problems:

- Scheduling of exams
- The flexibility in making different selections are related to the types of exams and the number of their sessions in a single day

The research questions of this paper are comprised of the followings:

- How to decrease the conflict of students' module exams?
- How students can at most sit for only one exam within a time interval?
- How can different options be managed into a schedule of exams when choosing the type of exam and the number of periods for an exam within one day?

The objectives of this paper are given as follows:

- To build a schedule of exams that reduces the conflict of module's exams
- To provide a schedule of exams that allows students to sit for at most a single exam within a given timeslot
- To produce a user interface system, which enables producing dissimilar choices, such as the type of exam to be given (Midterm Assessment (MTA) exam or final exam) and the number of daily exam sessions
- To display the timetable of the module's exams by the user interface system in order to be extracted into different formats

The contribution of this paper is to produce a scheduling algorithm for building an examination timetable in order to meet any encountered hard constraints. In fact, this can decrease the conflict of module's exams for students. Further, the paper develops a user interface system in order to present the result for scheduling module's exams. The outlines of the paper are organised as follows: Section II presents the secondary research (related research) of different scheduling algorithms that tackle the examination timetabling problem. In section III, details of the proposed algorithm are provided. The conclusions of this research are summarised in section IV.

Related Research

Ishak *et al.* (2016) present a Hybrid Genetic Algorithm (HGA) in order to deal with the problem of the university examination timetable, which allocates a number of exams into a number of vacant timeslots based on meeting a set of constraints. A number of solutions is introduced for this problem, in which low-level heuristic methods are used as the representation of five domain specific knowledge. Such methods are applied to easily create a timetable through the initial population. The key modules of the genetic operators in GA are tested and the best mixture of the genetic operators are used for building the Pure Genetic Algorithm (PGA). The HGA is produced by using the PGA based on three local optimisation techniques (e.g., move exam, swap exam and interchange timeslot) and it aims at enhancing the produced solutions. Exams

and timeslots are organised according to the three local optimization techniques using a number of equations in case the alteration reduces the penalty cost function. The results of the HGA can be improved by producing a trade-off between the local search and global search (Wan and Birch, 2013). A software system of the examination timetabling is implemented by (Chunbao and Nu, 2012) in order to execute efferent, precise and vigorous solutions for tackling different restrained timetabling issues. Chunbao and Nu (2012) demonstrate some fundamental characteristics of the software system, particularly, the paper test of the conflictive analysis method, which is able to produce a largely efficient data model with the intention to importantly enhance interactivity and the efficiency of the search.

Burke *et al*. (1996) declare that the presence of modularity through various UK institutions appears in an important growth of its complexity and in further difficulties of universities' administrators who ought to come up with a solution without any assistance based on a computer. Evolutionary methods are used to automatically solve this problem and to exhibit a lot of promises due to its efficient optimisation capabilities (Burke *et al*., 1996). Nevertheless, hybrid evolutionary techniques can produce more effective results. A memetic algorithm is proposed as a hybrid method to form an evolutionary technique, which combines the methods of the local search (Burke *et al*., 1996). Papaioannou *et al*. (2017) use the graph colouring technique in order to propose solutions for solving the timetabling problems effectively in higher educations. The aim of the graph colouring technique issue is to allocate different colours to the vertices in the graph and hence, the neighbouring vertices can obtain different colours (Papaioannou *et al*., 2017). It is assumed that the aim of timetabling issues in higher education, such as timetabling of lectures and exams, it is allocated day/time slots for examination or teaching. Therefore, the largest number of students can attend their lectures and sit for their exams by producing minimum conflicts. The fundamental motivation of this research relates to the critical issue of an efficient examination timetabling and courses that are normally emerging in different departments of the University of Patras in Greece (Papaioannou *et al*., 2017). Nonetheless, the examination timetables and lectures are created based on a number of heuristic methods that perform efficiently, but leaving a number of rooms for further enhancements. Papaioannou *et al*. (2017) develop a scheduling application by using a simple colouring method and MATLAB programming language. Such an application possesses the following inputs: Courses and constraints. The output of this application is an efficient timetable of lectures and exams.

The complex problem that is incurred through an examination timetable in universities, e.g., in big universities, increases for several grounds such as the massive sizes of universities, the increase in the elasticity of students' curricula and the focus on having a large set of constraints and aims (Alvarez-Valdés *et al*., 1997). An algorithm is introduced to deal with the above problem and it is intended to be applied in the University of Spain (Alvarez-Valdés *et al*., 1997). An incorporation of different heuristics methods that are based on the Tabu Search algorithm provides a preliminary solution such that no student has two exams at the same time. After that, an equal spacing between exams in the examination period is enhanced. This algorithm is inserted within a package to be used by administrators of different faculties and to propose a number of solutions based on different interested parties, which comprise departments, students and administrators (Alvarez-Valdés *et al*., 1997). Botangen (2014) develops and evaluates a web-based timetable application for a cooperative preparation of the class timetables at the Central Luzon State University. The development stage incorporates an algorithm for performing an automatic plotting and checking the conflicts between the timetable components according to their availability by considering a number of preferences and constraints. When using the timetable application, four academic university units can act cooperatively in order to build their class timetables. A comparison is conducted among the prospective problems with respect to the classes scheduling within the academic units. The classes scheduling apply an application based on five academic units, which are not applying the application (Botangen, 2014).

A study produced by (Akbulut and Yılmaz, 2013) attempt to solve the problem of the examination scheduling in many universities. Due to the large numbers of students and courses, a difficulty has arisen in scheduling exams through the midterm and final examination weeks. Additionally, Akbulut and Yılmaz mention that several works regarding the exams should be planned beforehand via the department at the end of every semester. Moreover, (Akbulut and Yılmaz, 2013) produce a scheduling system for different exams within the same rooms at the same time. In fact, timetabling is an essential task for academic institutions and industries when each system contains a number of resources that can be used for accomplishing a specific criterion (Hassan and Hassan, 2016). Timetabling should consider the use of resources according to a number of conflicting constraints. This introduces a system for exams timetabling that uses the graph colouring scheduling method. This method concentrates on two parts, which comprise: The constraints that are dealt by the system and the friendly interface of the system that is used by users (Hassan and Hassan, 2016). Scheduling of exams for a given list of courses are assigned to different timeslots by allocating them to a number of timeslots (e.g., conflicting exams (Kadry and Ghazal, 2016). Many

researchers introduce dissimilar methods in order to solve different examination scheduling problems. A model is proposed by (Kadry and Ghazal, 2016) in order solve the problem of examination schedule.

A model for the examination management system is provided by (Shelke *et al*., 2018) in order to assess the way of accessing the information that is related to an examination for a specific student within a determined class. In particular, this system is developed to compute the conventional method of performing different exams (Shelke *et al*., 2018). The project of the examination management system provides the facility for students to view their examination schedule, view the examination syllabus and observe the locations of their exams. The system represents a web-based system that is assessed by using real data via any available users (Shelke *et al*., 2018). The design and implementation of many examination scheduling systems are depicted for the Universiti Utara Malaysia (UUM) (Abdul-Rahman *et al*., 2017). Students who are registered in courses should be assigned to dissimilar timeslots and halls by avoiding any conflicts during the process of building the examination schedule. Moreover, a consideration of lectures' conflict is taken into account through this study (Abdul-Rahman *et al*., 2017). The developed system aims at providing an examination schedule without clashes and increasing the satisfactions of students and lecturers (Abdul-Rahman *et al*., 2017). The examination schedule is created by default where the developed system possesses the Graphical User Interface (GUI), which assists the user to input, create and alter the schedule. The GUI is used by means to enable a user in allocating or reallocating different courses into appropriate timeslots within the stage of creation and alteration that are directed by several heuristics methods.

Moreira (2008) introduces a method for tackling the issue of automatically creating the schedule of examinations. The use of the matrix model that is selected for the benefits of this model is produced based on the use of the Meta Heuristic algorithm, which applies the Genetic Algorithm into it (Holland, 1992). In fact, the model is used to build different examinations' schedules for various academic organisations of the higher education. The issue of scheduling relates to the procedure of performing an ideal level of particular constraints (Elen and Çayiroğlu, 2010). A solution is proposed for course timetabling issues in different universities through the automation of the student affairs based on the use of the Genetic Algorithm (Elen and Çayiroğlu, 2010). An examination schedule of the National University of Singapore was created for the fall semester in the academic year 2001/2002 by applying the UTTS Exam. This exam forms a scheduling program for building a timetable for the university exams (Lim *et al*., 2002). In particular, this type of schedule represents a scheduling program for creating a university examination timetable (Lim *et al*., 2002). The system has several components, which comprise the hybrid centralised cum decentralized scheduling method, the combined approach scheduling method and the entire procedure that are required to create an ultimate schedule (Lim *et al*., 2002).

Furthermore, (Hambali *et al*., 2020) apply the Genetic Algorithm and Simulated Annealing (Kirkpatrick *et al*., 1983) in order to form a Heuristic Approach (HA) that can tackle the issue of course scheduling in the Federal University Wukari (FUW). The implementation of the HA is given by considering soft and hard constraints along with the endurance of the fittest. A notice of the complexity of space and the interval has occurred. This leads to match between the number of rooms and the number of courses. Moreover, an approach is produced by (Mandal *et al*., 2020) to partially schedule the selected exams into rooms and timeslots and to sequentially enhance the vector of solution for partial exams. The following set of exams are scheduled as the procedure continues to proceed further. The procedure discontinues until all exams are successfully scheduled. Partial graph heuristic orderings with a modified great deluge algorithm (Abuhamdah, 2012) are used to resolve the problem of examination timetabling. Therefore, this approach uses Partial Graph Heuristic orderings with a modified Great Deluge algorithm (PGH-mGD).

The Proposed Scheduling Algorithm

This section is divided into the following subsections: Subsection I demonstrates the implementation environments and a discussion around the proposed solution. Subsection II highlights detailed explanations about the examination scheduling algorithm. Subsection III exhibits the fitness calculation function. The implementation part of the proposed algorithm is given in subsection IV. Finally, the obtained results are discussed in subsection V.

Implementation Environments and the Proposed Solution Discussion

In this section, the environment that is used for the implementation process and the proposed algorithm related to the exams' schedule are presented in detail. The main goal of this algorithm is to generate the exam's schedule for all academic faculties and to assure for the selected schedule that no conflicting modules are held. This implies that a student will not have two exams, for instance, at the same session or on the same day.

The implementation's environment of the proposed scheduling algorithm consists of three different academic faculties, which are computer studies, business studies and language studies at the Arab Open University. These faculties have more than one academic track for their students. For example, there are four different tracks in

the faculty of computer studies. In the faculty of business, there are three academic tracks. Additionally, the modules within students' tracks are divided into two main categories. The first category represents shared modules where all tracks include these modules within its study plan (in this study, the types of these modules are represented as a notation of common modules). The second category represents the independent modules, which only belong to a single track. Further, the tracks of modules are categorised by their level and through this solution, modules are divided into three levels (1, 2 and 3) according to students' registered years. For all faculties, one centralised database is applied by the proposed solution where this database includes all details of every module and its tracks for every faculty such as the module code, module name, module level, prerequisite module(s) and track identification ID number.

In addition to the previous implementation features, some inputs of data should be entered by the user before starting up with the scheduling algorithm by using a developed interface. The developed interface is discussed in detail in the following part. By using this interface, studnets should select the examination type (MTA or Final), the start and the end date of the examination schedule, the number of sessions per day (up to 3 sessions). Additionally, a set of constraints is considered to ensure that the main aims of this algorithm are efficiently and successfully achieved. In the first constraint, the algorithm should not select more than one module from the same level (1, 2 and 3) within the same examination session. In the second constraint, the algorithm does not select more than two or three modules (based on the user input) from the same level on the same day. However, they are distributed through different sessions. Further, a schedule of exams is created in order to reduce the conflict that is encountered in the module examination.

Exams Scheduling Algorithm Discussion

For each indicated faculties, the main steps of the proposed scheduling algorithm execute the generating process, which is outlined as follows:

1. Three lists of variable lengths are created for each academic faculty (computer studies, business studies and language studies). An extra particular list of all modules and prerequisite module(s) is created. The later list is used for the process of calculating the fitness, which is further discussed later.
2. For each academic faculty, all modules that are stored into the systems' main database are fetched into their corresponding lists that are previously created in Step 1.
3. A special function, namely, '*generate-schedule*' is used to process each of the faculty's own list according to its name. This function preforms the following procedures:
   - Generate and store a new copy of the lists of modules in order to keep the original lists without any modifications
   - A sub function, namely, '*constrained-sum*' is invoked from the '*generate-schedule*' function where this function is used to generate and return a list with randomly generated integers. This process indicates that these integers are used to determine the number of modules that are selected for each examination session. The values of random integers are ranged from the values (1, 2 and 3) and the total sum of these integers should be equal to the total number of modules for each academic faculty. In this stage, the department that is passed through to the '*generate-schedule*' function and the returned random integers are stored as a local variable list
   - For each session within an examination schedule, a number is randomly selected from the previously created random values (i.e., the local variable list). This selected number, for instance, represents an X number that is used to determine the number of times a loop is iterated. During each iteration (i.e., for each examination session), a module is randomly selected from the list of stored and copied modules for the currently selected faculty
   - The selected modules are then passed and used by a function, namely, '*calculate-fitness*', which is used to calculate the fitness percentage value for this session and return the percentage value of non-conflicting modules only. The fitness calculation process and the calculation of the fitness percentage value are further discussed in the following part
   - If the fitness percentage value of this session that is iterated reaches 100% and for all exams' session at that day, no more than three modules are selected from within the same level. After that, the generated session is approved and stored as an examination session
   - If the calculated fitness percentage of this session is found to be less than 100% or more, then three modules from the same level are selected between all sessions on that day. After that, the previous process is repeated for this session and different modules are selected until the calculation of the fitness value's percentage provides a percentage value of 100%. Moreover, this process is repeated up to N times, which represents a random number that is selected by the user before the generated algorithm starts with its processes. If after N times the problem is unresolved, the overall process starts from the first stage
   - Each examination session that calculates its fitness percentage to reach 100% according to the stated requirements is approved and stored.

- For each approved session, all selected modules that belong to this session are deleted from the list of modules (the list is generated from the original list of modules). This process is required in order to ensure that the selected modules are not selected for a different session and no repeated modules are selected for the overall full examination schedule
- After completing the previous steps, a single list is generated based on a condition, which states that all modules for that faculty should be within the range of 1 to 3 (the number of modules for this sessions) and should achieve 100% for the fitness value

4. The previous steps are repeated for each academic faculty. Additionally, a final approved list is provided for each faculty (i.e., computer, business and language faculties) and all lists are merged together into one final list according to the final generated examination schedule
5. 5The final generated examination schedule is passed through to the *calculate-fitness* function where this process is required to calculate the overall fitness per day and not per session
6. Steps 3, 4 and 5 are repeated N times and hence, different fitness values per schedule are calculated and generated from Step 5. The best calculated fitness value is selected as an approved examination schedule based on the proposed solution. This value is presented as an HTML page with its ability to be extracted as a PDF file. Moreover, a report is generated to provide a detailed information regarding the fitness calculation for each faculty on a daily basis. Figure 1 illustrates the overall steps of the proposed algorithm

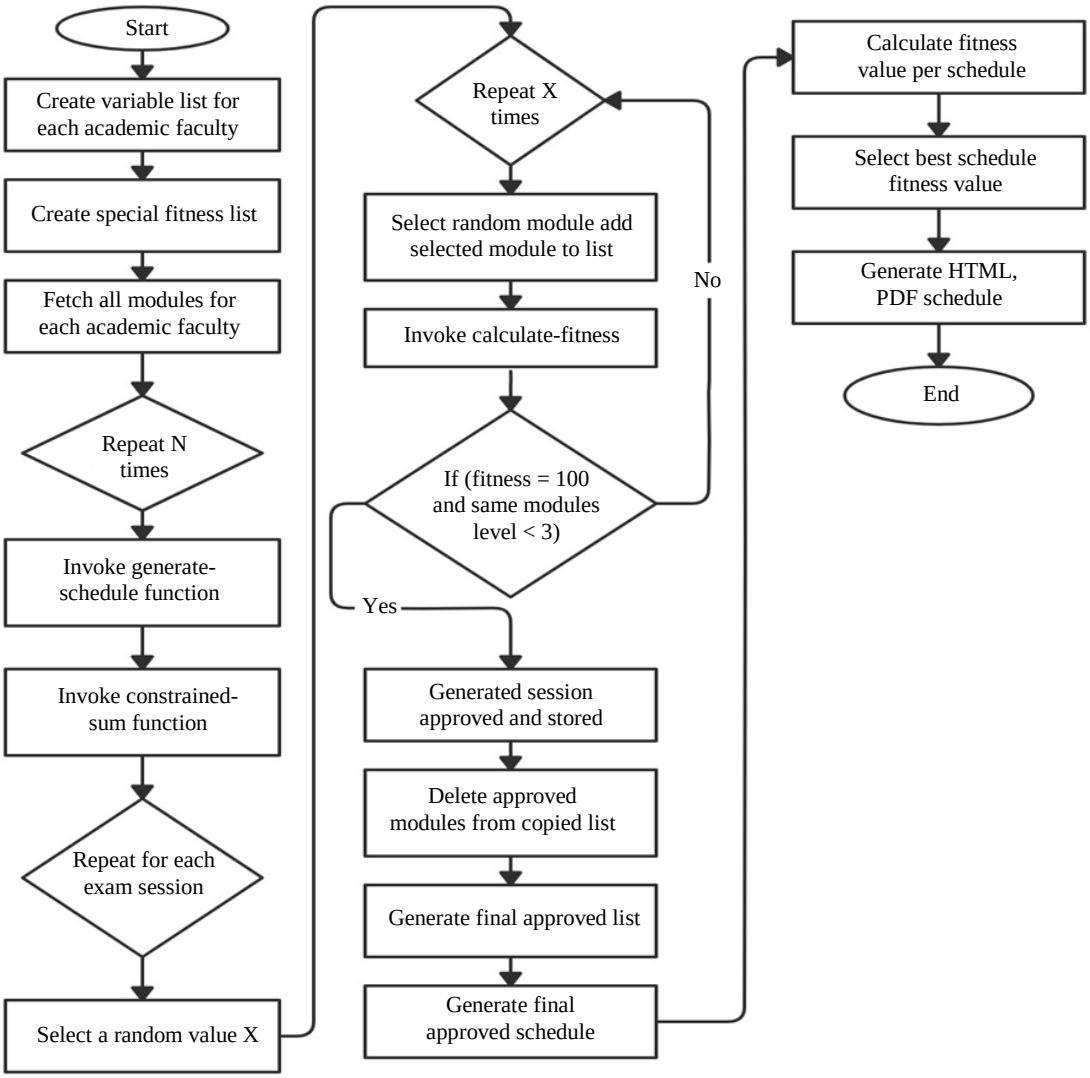

Fig. 1: The main steps of the examination scheduling algorithm

DOI: 10.3844/jcssp.2020.1139.1149

Fitness Calculation Function

The fitness function requires three types of determined parameters in order to perform its intended task correctly. The first parameter is a list that contains a number of codes for all modules (which fitness is calculated for it), the notation modules-list is used to refer to this list. The second parameter represents the prerequisite dictionary that contains each module code and its prerequisite module(s) if any. Finally, the last parameter represents a track dictionary that contains all modules' codes and a track for which each module belongs to that track due to some existing faultiest such as computer studies have four different tracks. The fitness calculation function starts with the process of creating three different lists where each list is used for a particular level (level 1, level 2 and Level 3) of modules. By using the above-mentioned modules-list, each of these new lists are filled with modules that belong to that level only. The second step is to create six empty sets such that two sets are used for each level of the modules' list. One of the two sets contains the modules' codes without any conflicts (from within the same level) and the other set of modules contains conflicts on the schedule (from within the same level).

The next step in this function is to check the length (the number of Level 1's modules). If there is one element, it is added to the set of Level 1's modules, which does not contain any conflicts since there is no option to have any conflicts in such case. Nevertheless, if Level 1's modules possess more than one element (i.e., modules), a loop is iterated through all the elements in order to provide comparisons among the entire elements (modules). This loop, checks as to whether or not these modules are prerequisite to each other. If they are, both of these modules are added to the set of no conflicts since the two modules cannot be registered within the same semester by a student. If the modules are not prerequisite, one extra check is applied as to whether or not the two modules are derived from a number of common modules (which shared between many tracks) or from different tracks. If the checked modules belong to the common list, the elements are added to the second set, which represents Level 1's modules with conflict. The previous step is applied for each level of the modules separately and leads to obtain 6 sets for which each set contains its corresponding elements (modules) as previously mentioned.

After that, the fitness percentage value is calculated by computing the sum of length (the number of elements) for each of the three sets that belong to each module level with no any conflicts. For example, (the set of Level 1 with no conflicts) + (the set of Level 2 with no conflicts) + (the set of Level 3 with no conflicts). Prior to the calculation of the fitness percentage, the set of Level 1 modules should be checked. If all modules within this level are added to the conflict set and the set of no conflict is empty, then one of those modules is removed through to the no conflicting set and the value '1' is added to the total sum of the fitness since one course is considered as a no conflicting course. Finally, the fitness percentage value is calculated as follows:

> Fitness-percentage = (Total-Fitness/Totla-Exams) * 100 where the Total-Fitness denotes the sum of modules with no conflicts through the entire sessions.

The Total-Exams denote the sum of modules per one session or per a single day. For example, when the fitness percentage value is calculated for each session, the Total-Exams value represents the number of modules of this session only and its maximum value is '3'. Moreover, when the fitness percentage value is calculated per one day, the Total-Exams are assigned the value of the total number of modules for all sessions on that day.

Proposed Algorithm Results and Discussion

This section discusses comprehensive details on the implementation of the proposed scheduling algorithm, findings and results discussions, including the overall extra functions of the proposed solution.

The proposed algorithm achieves the main functions that represent the ability to generate examination schedule with best distributions for faculties' modules, zero conflicts in the generated schedule (a student will not have two exams, for instance, at the same session or on the same day) and a fitness of 100% for the overall generated schedule. Further, the algorithm takes less time during its process where this improves the maintainability and usability by making it more user-friendly through a simple and an easy use of a user interface. As depicted from Fig. 2, users are allowed to pass the required parameters by using the provided interface before this algorithm begins to operate. Users can select the examination type in case of AOU's examination types (MTA and final exams), exam duration, start and end dates of exams, duration per day and the number of sessions per day. Once the required parameters are entered, the algorithm begins to process in order to generate the required exam's schedule.

Based on Fig. 3, an example of the generated MTA exam's schedule is provided. This schedule demonstrates that each examination session contains only one module from the same level (level 1, level 2 and level 3) for all of the academic faculties (computer, business and language faculties).

The entire scheduled sessions include different levels of modules and no one includes two modules from the same level. For instance, some sessions include three modules from three levels (Levels 1, 2 and 3) such as session 1 of the first day of the examination schedule for

DOI: 10.3844/jcssp.2020.1139.1149

the faculty of computer studies. Some sessions include two modules from two different levels (Levels 2 and 3) as shown on Day 2, Session 1, for the faculty of computer studies. Additionally, it can be illustrated that two or three modules are selected and distributed from the same levels within the available examination sessions when checking a full schedule on a particular day. The selected modules from the same level of the same day do not refer to the same track. For example, the three selected modules from Level 3 of the first day for computer studies (T316, T318 and TM366) refer to three different tracks (Fig. 3).

At the same time, the scheduling algorithm generates multiple schedules for each academic faculty according to the N value that is entered by the user. A report about the fitness calculation is extracted and printed along with the chart diagram. In fact, this report demonstrates the fitness calculation for reaching the academic faculty generated schedule. Further, it highlights the calculation of the fitness conflict percentage among the generated exams' schedules. For each academic faculty, the algorithm selects the generated schedule with a fitness of 100% and finally, it integrates all selected schedules into one schedule (Fig. 3).

Furthermore, Fig. 4 illustrates an example of a generated fitness report where 100% is achieved for the computer science faculty in the generated schedule Number 2. The target percentage is achieved for the business faculty in the generated schedule Number 3 and Number 1 for the language faculty, which is integrated into a final approved schedule as previously indicated.

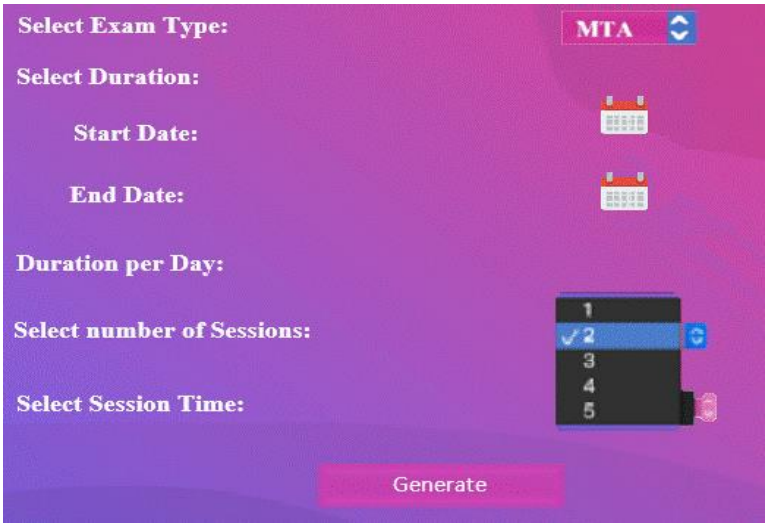

Fig. 2: The main parameters' interface of the examination scheduling algorithm

AOU MTA Exam schedule

Date	Department	Session 1 12:00-02:00	Session 2 03:00-05:00	Session 1 00:06-08:00
2019-11-19	Business	B205A	ECO101	BUS101 B326
	CS	T215A M109 T316	TM298 T318	TN366 T277
	English	E302A	AA100A	U214B
2019-11-20	Business	B123 B327	BU310 LB170 B205B	ACC300
	CS	TM355 TM240	TM356	TM111
	English	AA100B	TR102	E302B EL230 EL122
2019-11-21	Business	MKT331 BUS110	B325 B292	BUS115 MKT332
	CS	M269 TM103	MT129	T216B MT131 TM352
	English	EL117	A230B	U214A
2019-11-23	Business	B207A ACC302	SYS380 B124	B122
	CS	TM105 TM351	MT101	TT284
	English	A230A	EL121N	E304A

Fig. 3: An example of generated schedule of the MTA examination at the Arab Open University

DOI: 10.3844/jcssp.2020.1139.1149

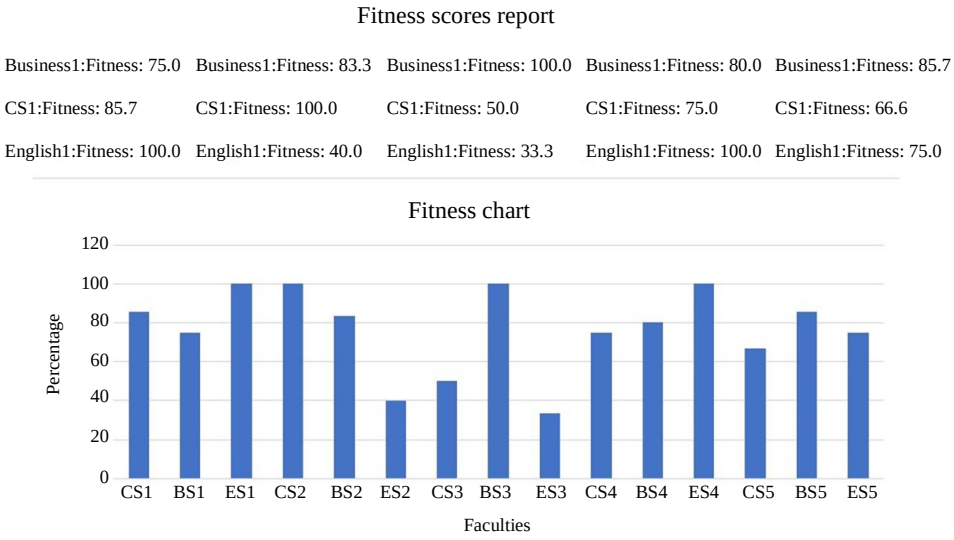

Fig. 4: An example of the calculated fitness report

Table 1: The main functions of the proposed solution

Key points	Exam planner	College time table	Proposed solution
Easy to use	Yes	NA	Yes
Space required (size)	NA	Yes	Yes
Automatically produces high-quality timetables that meet all requirements.	Yes	NA	Yes
Produces clash-free results and improved spacing of exams for students.	Yes	Yes	Yes
Flexibility	Yes	NA	Yes
Generating process started instantly		Yes	Yes
Cross platform	Yes	NA	Yes

Results and Discussion

The proposed scheduling algorithm aims at achieving the entire targeted main goals, which are efficiently provided. All of the required activities are easily performed by using a simple user interface that can pass the required preliminary parameters through to the algorithm before starting the process of generating the examination schedule. In order to improve the level of users' convenience, usability and simplicity are taken into consideration during the development of the proposed algorithm. As previously discussed, it is found to be proven from the obtained results that the proposed algorithm is simply implemented. The schedule generating process can represent an automated process by exploiting the processing power of the currently available computer systems. Further, the results demonstrate that the generated schedule can be successfully generated with no conflicts where the selected schedule is based on the required constrains. In fact, such constraints assure that no modules from the same level are selected on the same session.

The same levels of modules are selected on the same day, but not within the same exam session on that day and the most important constrain is that the fitness percentage for each session of modules should reach 100%. Consequently, the algorithm is considered accurate when achieving the targeted aims.

The main functions of the proposed algorithm are provided in Table 1. This table provides a summarised comparison between the proposed algorithm and the two similar solutions that are used for generating the examination schedule. Both solutions comprise the exam planner and college timetable. The main key points of the provided table are based on the usability, solution size, procedure automation, flexibility and immediate executional time. In this table, the entire key points are efficiently achieved based on the proposed solution.

Conclusion

A scheduling algorithm is produced to solve the problem of the examination schedule in a particular university. The proposed algorithm introduces a timetable of different examination modules that prevent each student from sitting for more than one exam on the same day and time. All hard constraints are applied based on the implementation of the proposed algorithm.

Additionally, a user interface system is developed to make it simple for selecting different options such as the type of an exam, the number of exam's sessions per day and also the user interface system that displays the examination timetable. The proposed algorithm decreases the number of examination conflicts and enables a student to only perform one exam within a specific day and time. Further, the proposed algorithm demonstrates an efficient accuracy for all examination sessions and sections. To sum up, it is found to be proven that the results of the accuracy of each individual session and for the entire sessions are both satisfactory.

Acknowledgement

The authors of this manuscript would like to express their appreciations and gratitude to the Arab Open University in the Kingdom of Saudi Arabia for supporting this research.

Author's Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Abdul-Rahman, S., Benjamin, A. M., Omar, M. F., Ramli, R., Ku-Mahamud, K. R. & Jabbar, W. K. A. (2017). Designing and Implementation a Web-Based Architecture for an Examination Timetabling System. Journal of Engineering and Applied Sciences, 12**,** 7299-7305.

Abou Kasm, O., Mohandes, B., Diabat, A., & El Khatib, S. (2019). Exam timetabling with allowable conflicts within a time window. Computers & Industrial Engineering, 127, 263-273.

Abuhamdah, A. (2012). Modified great deluge for medical clustering problems. International Journal of Emerging Sciences, 2(3), 345-361.

Akbulut, A., & Yilmaz, G. (2013). University exam scheduling system using graphcoloring algorithm and rfid technology. International Journal of Innovation, Management and Technology, 4(1), 66.

Alvarez-Valdés, R., Crespo, E., & Tamarit, J. M. (1997). A tabu search algorithm to schedule university examinations. Qüestiió, 21(1-2), 201-215.

Botangen, K. A. W. (2014). Web-Based Class Scheduling for a Collaborative Preparation of Block-Based Schedules. International Proceedings of Economics Development and Research, 81, 161.

Burke, E. K., Newall, J. P., & Weare, R. F. (1996, August). A memetic algorithm for university exam timetabling. In international conference on the practice and theory of automated timetabling (pp. 241-250). Springer, Berlin, Heidelberg.

Burke, E., Elliman, D., Ford, P., & Weare, R. (1995, August). Examination timetabling in British universities: A survey. In International Conference on the Practice and Theory of Automated Timetabling (pp. 76-90). Springer, Berlin, Heidelberg.

Carter, M. W. (1986). OR practice—a survey of practical applications of examination timetabling algorithms. Operations research, 34(2), 193-202.

Carter, M. W., & Laporte, G. (1996). Recent developments in practical examination timetabling. Selected papers from the first international conference on practice and theory of automated timetabling (pp. 3-21).

Cavdur, F., & Kose, M. (2016). A 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, 18(1), 119-129.

Chunbao, Z., & Nu, T. (2012). An intelligent, interactive & efficient exam scheduling system (IIEESS v1. 0). Proceeding of the Practice and Theory of Automated Timetabling (PATAT), Norway, 437-450.

Clark, D. (1993). Exam scheduling by Tabu Search. Australian Soc. Ops Res. Bulletin,, 12**,** 5-9.

Dener, M., & Calp, M. H. (2018). Solving the exam scheduling problems in central exams with genetic algorithms. Mugla Journal of Science and Technology, 4, 102–115.

Elen, A., & Çayiroğlu, İ. (2010). SOLVING OF SCHEDULING PROBLEM WITH HEURISTIC OPTIMIZATION APPROACH. Teknoloji, 13(3).

Gonsalves, T., & Oishi, R. (2015). Artificial Immune Algorithm for exams timetable. Journal of Information Sciences and Computing Technologies, 4(2), 287-296.

Hambali, A. M., Olasupo, Y. A., & Dalhatu, M. (2020). Automated university lecture timetable using Heuristic Approach. Nigerian Journal of Technology, 39(1), 1-14.

Hassan, M. A. H., & Hassan, O. A. H. (2016). Constraints aware and user friendly exam scheduling system. Int. Arab J. Inf. Technol., 13(1A), 156-162.

Hertz, A. (1991). Tabu search for large scale timetabling problems. European journal of operational research, 54(1), 39-47.

Holland, J. H. (1992). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence. MIT press.

Hosny, M., & Al-Olayan, M. (2014, August). A mutation-based genetic algorithm for room and proctor assignment in examination scheduling. In 2014 Science and Information Conference (pp. 260-268). IEEE.

Ishak, S., Lee, L. S., & Ibragimov, G. (2016). Hybrid Genetic Algorithm for University Examination Timetabling Problem. Malaysian Journal of Mathematical Sciences, 10(2), 145-178.

Ji, X., & Ma, K. (2020). Toward Automatic Scheduling Algorithm with Hash-Based Priority Selection Strategy. In Soft Computing for Problem Solving (pp. 35-42). Springer, Singapore.

Kadry, S. & Ghazal, B. (2016). New algorithm to solve examination timetable problem.

Kalender, M., Kheiri, A., Özcan, E., & Burke, E. K. (2013). A greedy gradient-simulated annealing selection hyper-heuristic. Soft Computing, 17(12), 2279-2292.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

Lei, Y., Gong, M., Jiao, L., & Zuo, Y. (2015). A memetic algorithm based on hyper-heuristics for examination timetabling problems. International Journal of Intelligent Computing and Cybernetics.

Leite, N., Melício, F., & Rosa, A. C. (2019). A fast simulated annealing algorithm for the examination timetabling problem. Expert Systems with Applications, 122, 137-151.

Lim, A., Ang, J. C., Ho, W. K., & Oon, W. C. (2002, July). UTTSExam: a university examination timetable scheduler. In AAAI/IAAI (pp. 1004-1005).

Malkawi, M., Hassan, M. A. H., & Hassan, O. A. H. (2008). A New Exam Scheduling Algorithm Using Graph Coloring. International Arab Journal of Information Technology (IAJIT), 5(1).

Mandal, A. K., Kahar, M. N. M., & Kendall, G. (2020). Addressing Examination Timetabling Problem Using a Partial Exams Approach in Constructive and Improvement. Computation, 8(2), 46.

Marie-Sainte, S. L. (2015). A survey of particle swarm optimization techniques for solving university examination timetabling problem. Artificial Intelligence Review, 44(4), 537-546.

Moreira, J. J. (2008). A system for automatic construction of exam timetable using genetic algorithms. Tékhne-Revista de Estudos Politécnicos, (9), 319-336.

Novita, N., Ganiardi, M. A., Ariyanti, I., & Khairunnisa, D. (2020). The Design Of The Semester Exam Scheduling Application At The State Polytechnic Of Sriwijaya. In Journal of Physics: Conference Series (Vol. 1500, p. 012123).

Papaioannou, E., Athanassopoulos, S., & Kaklamanis, C. (2017). EFFICIENT COURSE AND EXAM SCHEDULING USING GRAPH COLORING. International E-Journal of Advances in Education, 3(7), 51-59.

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. Journal of scheduling, 12(1), 55-89.

Ross, P., & Corne, D. (1994). Applications of genetic algorithms. AISB Quaterly on Evolutionary Computation, 89, 23-30.

Selemani, M. A., Mujuni, E., & Mushi, A. (2013). An Examination Scheduling Algorithm using Graph Colouring-The Case of Sokoine University of Agriculture. International Journal of Computer Engineering & Applications, 2(1/3), 116-127.

Shelke, S., Yadav, I., & Srivastava, J. (2018). College examination system. International Journal of Advance Research, Ideas and Innovations in Technology. 4, 1445-1454.

Thepphakorn, T., Pongcharoen, P., & Hicks, C. (2014). An ant colony based timetabling tool. International Journal of Production Economics, 149, 131-144.

Vatansever, S., & Arici, N. (2019). Creating The Best Session Plan With Artificial Immune System for Common Exam In Secondary Education Institutions. Artificial Intelligence Studies, 2(1), 1-14.

Wan, W., & Birch, J. B. (2013). An improved hybrid genetic algorithm with a new local search procedure. Journal of Applied Mathematics, 2013.