International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 6, December 2020, pp. 6558~6573
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp6558-6573

Courses timetabling based on hill climbing algorithm

Abdoul Rjoub
Computer Engineering Department, Jordan University of Science and Technology, Jordan

Article Info

Article history:

Received Sep 14, 2019
Revised Feb 29, 2020
Accepted Jun 5, 2020

Keywords:

Course schedule
Hard and soft constraints
Hill climbing algorithms
Manual school timetable
Time slot

ABSTRACT

In addition to its monotonous nature and excessive time requirements, the manual school timetable scheduling often leads to more than one class being assigned to the same instructor, or more than one instructor being assigned to the same classroom during the same slot time, or even leads to exercise in intentional partialities in favor of a particular group of instructors. In this paper, an automated school timetable scheduling is presented to help overcome the traditional conflicts inherent in the manual scheduling approach. In this approach, hill climbing algorithms have been modified to transact hard and soft constraints. Soft constraints are not easy to be satisfied typically, but hard constraints are obligated. The implementation of this technique has been successfully experimented in different schools with various kinds of side constraints. Results show that the initial solution can be improved by 72% towards the optimal solution within the first 5 seconds and by 50% from the second iteration while the optimal solution will be achieved after 15 iterations ensuring that more than 50% of scientific courses will take place in the early slots time while more than 50% of non-scientific courses will take place during the later time's slots.

Corresponding Author:

Abdoul Rjoub,
Computer Engineering Department,
Jordan University of Science and Technology,
University Campus, Irbid 22110, P. O. Box. 3030, Jordan.
Email: abdoul@just.edu.jo

1. INTRODUCTION

In recent times, scheduling of timetabling [1, 2] has gained considerable importance and has witnessed rapid growth due to its significant role in the success of school [3, 4], university [5, 6] and examination timetable placements [7]. Given the different challenges facing each of these sectors, designing an appropriate algorithm for each will contribute to solving its challenges separately. In this paper, the issue of school timetable scheduling is addressed.

Usually, a school timetable is built at the beginning of each semester to organize the academic program of study by considering huge amounts of information to be filled in such as courses, classes, instructors, the number of lectures for each course through the week and finally, the maximum number of hours that are assigned to each instructor. These parameters should be taken in consideration when someone plans to build a timetable for any academic institution. Unfortunately, ensuring compatibility of these inputs is time consuming especially when done manually. In addition, the manual method requires human labor, the instructors' help sometimes, which means more arrangement efforts, more rehearsals in terms of writing and reviewing the required parameters and resulting partiality, which, in turn, may lead to problems in the work environment [8]. These challenges affect the teaching process and instructors' scientific efficiency and productivity. Hence, this is the main motivation behind the proposed technique to overcome the manual timetable scheduling and have it replaced by an automated one.

Countless efforts have been made in this area. Based on local search techniques [9], the process begins with some initial state that is created randomly or read from a file without satisfying any hard constraints [10]. Therefore, there is a need to satisfy such constraints to create a correct state and then attempt to achieve the soft constraints [11]. Hence, this would require more operations to convert the incorrect random state into an acceptably more correct state that, in turn, requires additional time. In this new approach, the hard constraints are already satisfied during the initial state as fulfillment of the soft constraints commences while, in the process, maintaining the hard constraints from any violations.

In [12, 13] a class-oriented representation of Timetable is used, which implicitly avoids classroom contentions, the instructor-oriented representation of timetable is used which implicitly avoids instructors time table contentions. In this paper, both Class-oriented and Instructor-oriented representations of a timetable, which avoid class clashes and instructor clashes are used in parallel.

Across the literature, various algorithmic classes have been proposed to address the scheduling problem at academic institutions. These algorithms include: Algorithms falling under the genetic algorithms class which are often used to tackle huge amounts of data, and simulated annealing algorithms which lend themselves fairly well to problems with large search spaces. Neither of these classes of algorithms will be leveraged in the current work. In the former class of algorithms large memory sizes are required [14, 15], while in the latter category certain disadvantages are commonly encountered. These deal with the fact that the optimal solution usually depends on the initial state where the speed of convergence is rather too slow than many other algorithms [16, 17]. Furthermore, in this work no consideration will be given to Tabu's search algorithm which is classified under the local search family and would require a large number of iterations for its convergence [18].

In this paper, a novel technique is proposed; this technique is made-up of two stages: In the first, an initial solution is created in a way that would avoid any clashes leveraging Class-oriented and Instructor-oriented representations with various other functions to make sure that no time slots are selected in ways that would end up in clashes. In the second stage, the hill-climbing algorithm, with some modifications, per the procedure outlined in section 3, is used, leveraging the number of advantages it entails including its low power requirements, its rather fast convergence and its reasonable requirements of low memory sizes [19].

The rest of this paper is organized as follows: section 2 details the common problem of school timetabling. Section 3 illustrates the modified Hill Climbing Algorithm. Section 4 elaborates the mechanics of the proposed technique, as section 5 explains the main results and offers pertinent analyses. Finally, section 6 provides the conclusions.

2. PROBLEM STATEMENT: DESCRIPTION AND REQUIREMENTS

The problem of school timetable scheduling has traditionally been defined by a process that would ensure integrity and consistency as we combine instructor, classroom and course times together into commonly feasible time slots, satisfying a certain group of constraints that would make the structure of the targeted school time table sound and correct. The basic factors of the school timetable-scheduling problem contain from three layers, these three layers are featured in the same diagram. Layer 1 contains the most essential components in the process, *viz.*, Instructors, Classes and Courses. Layer 2 features the most important hard constraints (Classroom clashes, Instructor clashes, lecture and day teaching loads) that make the overall structure of the timetable logically correct. Layer 3 contains a group of soft constraints (Instructors' preferences, Daily working loads (hours), desired Last Lecture and Preferences of the Classes) where these constraints are meant to enhance the School Timetable quality [20, 21].

2.1. Layer 1: instructors, classes, courses

The Instructors: *I (j); j= 1, 2, 3, … , Ị;* are an essential element in any school timetable-scheduling mechanism, where each one of the instructors has a parameter called *Specialty*, which is an important characteristic that can take values for Primary Level Instructor, Diploma and Bachelor's degrees or even those that are above. These values help the programming process to determine the class level an instructor can teach. More specifically, an instructor with a Primary Level Instructor specialty can only teach primary level classes, while that with a Diploma specialty can teach both primary and intermediate level classes and the instructor whose specialty is at the Bachelor's degree or above can teach all class levels. The Classes: *C (j); j= 1, 2, 3, … , Ç*; the classes constitute the second most important element in school timetable scheduling, with each class possess in a parameter called the *Class-Level*. This parameter can take on values of primary, intermediate or secondary, with the values indicating the specialty of the instructor that allows him to teach a particular class, and finally, the Courses: *K (j); j= 1, 2, 3, … , Ḳ;* which are the last element in the school timetable scheduling process, where each course has a parameter called *Course type*. This parameter has two values: scientific courses such as *Physics*, *Mathematics* and *Chemistry*, and non-scientific ones such as *Arts*, *Sports*, *History* and *Computer skills*. This parameter is usually weighed in more heavily in assigning scientific courses to early time slots, and those that are non-scientific courses to later time slots; something that would contribute more positively towards students' learning appetites. All of these elements are scheduled to take place within certain time slots during school day, TS (*j*) = *1, 2, 3, ... ,Ş*. The solution space of such a problem thus becomes:

$$SPACE = I \times C \times K \times TS \tag{1}$$

Assuming that the instructor teaches just one or two courses, a classroom takes some but not all courses, and it is often known which instructor teaches what course and at which classroom. In addition, lectures take place during a limited time frame (e.g. day), which inherently limits the actual timeslots. As a results, SPACE is reduced to V, where V $\subset$ SPACE, which represents only those lectures the time scheduling of which makes sense, such that

$$\boldsymbol{V_{i,c,k,s}} = \begin{cases} \mathbf{1}; & if\ instructor\ i\ teaches\ course\ k\ to\ class\ c\ \ in\ time\ slot\ s \\ \mathbf{0}; & otherwise \end{cases} \tag{2}$$

where $\boldsymbol{v_{i,c,k,s}}$ represents an element of $V_{i,c,k,s}$.

2.2. Layer 2: hard constraints

The second layer refers to the hard constraints, which encompasses those constraints that make the structure of the timetable sound and correct which has to be satisfied in all cases and at all times. These constraints include:

a. Classroom clash (conflict)

Classroom clash refers to a situation whereby we would allocate more than one instructor to the same classroom timetable in the same time slot. This problem occurs because the administrator cannot see the timetable for all classrooms at the same time. Consequently, the application of the proposed solution avoids this problem by using a Class-oriented representation of the timetable, which allows seeing the entire timetable for all classrooms all at the same time. Class clashes can be represented as:

$$\textstyle\sum_{i\,\in\,I}\left(v_{c1,k1,s1} + \bar{v}_{c1,k2,s1}\right) = 1; \quad \forall\, v \in \mathrm{V} \tag{3}$$

which means that for any classroom *c₁*, no two courses (*k₁*, *k₂*) are assigned to it at the same time slots_1 regardless of the instructor who gets assigned to that particular course. Instructors clash (conflict): This is quite the opposite of the classroom clash problem such that one cannot allocate more than one classroom in the same time slot for a given instructor's timetable. Such a problem usually occurs since the administrator cannot see the timetables for all instructors all at the same time. Here, again, the proposed solution avoids this problem using Instructor-oriented representation of a timetable such that it allows displaying the timetables for all instructors all at the same time. Instructor clash can be represented in the form:

$$\textstyle\sum_{k\in K}\left(v_{c1,i1,s1} + \bar{v}_{c2,i1,s1}\right) = 1; \quad \forall\, v\ \in V \tag{4}$$

This means that for any given instructor i_k, no two classrooms (c_i, c_j) are assigned to him/her in the same time slot s_k regardless of which courses he/she is assigned to. Finally, the Lectures (weekly) load: In every School policy, each course has a preset number of lectures per week per each classroom. This constraint readily implies that we cannot possibly increase or decrease the number of lectures for a course in any given classroom. Therefore, the required number of lectures for course $k_1$taught at classroom c_1 during the week regardless the time slot distribution or the particular instructor of the course is given by:

$$CKL \equiv \textstyle\sum_{d\in D}\sum_{s\in TS}\sum_{i\in I} v_{c1,k1}\,; \quad \forall\, v \in V \tag{5}$$

where ***D* (*j*) = *1, … , 5***; which represents the weekly school days. Similarly, the number of lectures has to be taught by instructor i_1 during the week regardless of courses get taught, the time slots or the classrooms is given by:

$$IL \geq \textstyle\sum_{d\in D}\sum_{c\in C}\sum_{k\in K}\sum_{s\in TS} v_{i1}\,; \quad \forall\ v \in V \tag{6}$$

b. Day load

The number of lectures per course for each classroom must be divided into five working days. For example, if a given class room caters for five *Math* course lectures, then each day can have no more than one lecture for the same course. Consequently, three cases will arise assuming that there are 5 school days (working days) in a given week:

Case 1: Class *c* has less than five lectures of course *k* during the week

$$\sum_{d=1}^{5} \sum_{s \in TS} \sum_{i \in I} v_{k1,c1} < 5; \quad \forall\, v \in V,\ \forall\ s \in TS,\ \forall\ i \in I \tag{7}$$

Then,

$$r_{k1,d} = \{0,1\}; \quad \forall\, d \in D \tag{8}$$

where,

$$R_{d,k,c} \equiv \begin{cases} 1; & if\ class\ c\ has\ a\ lecture\ of\ course\ k\ on\ day\ d \\ 0; & otherwise \end{cases} \tag{9}$$

Case 2: Class *c* has exactly five lectures of course *k* during the week

$$\sum_{d=1}^{5} \sum_{s \in TS} \sum_{i \in I} v_{k1,c1} = 5; \quad \forall\, v \in V,\ \forall\ s \in TS,\ \forall\ i \in I \tag{10}$$

Then,

$$r_{k1,d} = 1\,; \quad \forall\, d \in D \tag{11}$$

Case 3: Class *c* has more than five lectures of course *k* during the week

$$\sum_{d=1}^{5} \sum_{s \in TS} \sum_{i \in I} v_{k1,c1} > 5; \quad \forall\, v \in V,\ \forall\ s \in TS,\ \forall\ i \in I \tag{12}$$

Then,

$$1 \le r_{k1,d} \le 2\,; \quad \forall\, d \in D \tag{13}$$

such that

$$\sum_{d=1}^{5} r_{k1,d} = \sum_{d=1}^{5} \sum_{s \in TS} \sum_{i \in I} v_{k1,c1} \tag{14}$$

The final and definite constraint is that no free lectures are allowed to be assigned to any classroom before the last lecture in that particular classroom on a given day.

$$V_{c1} \neq 0; \quad \forall\ s < Class\ last\ lecture,\ \forall\, i \in I,\ \forall\, k \in K \tag{15}$$

2.3. Layer 3: soft constraints

Soft constraints refer to the parameters that are somewhat less important than their hard counterparts are. Here, although these would show up in the computer program, it would not be very necessary to account for all of them. Definitely, the more contained it is, the better the schedule. As mentioned earlier, layer 3 is made up of two constraints: Classroom preferences: Preferences of classrooms implicitly refer to the three constraints associated with classrooms: Scientific Courses: Scientific courses such as *Physics*, *Mathematics*, *Chemistry* and *Biology* are to be considered during the early time slots. Non-Scientific Courses: Non-Scientific courses such as *Arts, Sports, History and Computer skills* are to be considered for the latter time slots. The Instructor preferences which is an essential part of the teaching process addresses the instructors' convenience and comfort levels during the semester, such that the teaching effort is reflected more positively on the overall educational process. Hence, several instructor-centric facets are considered. These include an instructor's preference on whether he/she would rather start their daily teaching in the first lecture slot of each day or not. Also, an instructor's preference to call it an early quits, on any particular day of choice, and finally whether the instructor still happens to be a student of the institution himself/herself with certain convenience-driven needs that need to be accommodated by the timetable organizers.

3. EFFICIENT HILL CLIMBING ALGORITHM

The Hill Climbing Algorithm (HCA) is a mathematical technique that belongs to the category of local search algorithms [20-22]. It is a repetitive algorithm that starts with an initial population chosen from the problem solutions space. It then goes through a process that improves these solutions by incrementally changing one of the elements in order to create new solutions. In the process, solutions that have higher cost values are adopted in an effort to generate a new population and the process is repeated until no further improvements can be achieved [23].

Under the proposed technique, the Hill Climbing algorithm is modified to be used as an optimization algorithm, where the optimization process continues as long as the calculated heuristic value δ, i.e., cost value, of the last optimized solution is less than zero; other than that the process is stopped if this value does not change for eight consecutive executions, as this would indicate that no further improvement can be achieved, and the solution is finally reached [24]. The heuristic value is a linear function of soft constraints; such that it measures to what degree the current solution satisfies these constraints. It checks if the two constraints of class preferences are satisfied for all classes in the school, such that:

$$StCt^{c} = \begin{cases} 0; & if\ the\ two\ constraints\ of\ classes'preferences \\ & are\ satisfied\ for\ class\ c \\ -1; & if\ one\ of\ them\ is\ not\ satsfied\ for\ class\ c \\ -2; & if\ both\ of\ them\ are\ not\ satisfied\ for\ class\ c \end{cases} \tag{16}$$

where $StCt^{c}$ represents a measure of the class preference constraints for class c.

In addition, it checks if the constraints of the instructor's personal preferences, daily work hours and last lecture preferences are satisfied for all instructors, such that:

$$StCt^{i} = \begin{cases} 0; & if\ the\ instructor\ desires, daily\ work\ hours\ and\ last\ lecture \\ & are\ satisfied\ for\ instructor\ i \\ -1; & if\ one\ of\ them\ is\ not\ satsfied\ for\ instructor\ i \\ -2; & if\ two\ of\ them\ are\ not\ satisfied\ for\ instructor\ i \\ -3; & if\ all\ f\ them\ are\ not\ satisfied\ for\ instructor\ i \end{cases} \tag{17}$$

where $StCt^{i}$ represents the measurement of the instructor's preferences, daily work hours and last lecture preferences for instructor i. As a result, the heuristic parameterδ is given as:

$$\delta = \sum_{c=1}^{Ç} StCt^{c} + \sum_{i=1}^{I} StCt^{i} \tag{18}$$

The initial and optimal value forδ is 0. It is decreased by 1 each time any of the constraints is found to be violated for any classroom or instructor. Hard constraints are not considered as MHCA does not provide a solution unless all the hard constraints are satisfied.

The input vector that is required to run this technique is: $\langle Course\ Details,\ Class\ Details,\ Instructor\ Details \rangle$ as follows:
- Course details: such as course title, course type (Scientific or Non-Scientific).
- Class details: such as class name, class division, class level (Primary, Intermediate or Secondary) and the list of all courses that are taught in a given classroom.
- Instructor details: such as his/her ID number, name, number of lectures he/she can teach, instructor specialty (Primary level, Diploma or Bachelor's degree or above) and a list of all classrooms and the courses that are taught by this instructor.

4. THE PROPOSED TIMETABLE SCHEDULING TECHNIQUE

Once all required data will have been identified and accounted for, the creation process will commence in three separate phases. The first phase is the data organization phase (DOP) which organizes all the data that was accounted for earlier. The second phase is the initial state phase (ISP) which creates an initial solution for the timetable and the third phase is the satisfied soft constraints phase (SSCP) which improves the initial solution using MHCA to satisfy all of the soft constraints as it possibly can in order to achieve the "best" final solution [25].

4.1. The data organization phase (DOP)

When the system begins to build a school timetable, it commences by organizing the required data as a first task. In this phase (DOP), the system starts the creation of the timetable by loading the required data and setting up the data structures. Such required data in the school object splits into four types of objects:

- Instructor Object: contain variables such as Instructor-ID, Instructor-Total-Number-Of-Lectures and Class-Course-List that refers to the classes and courses which this instructor teaches and the Instructor-Number-Of-Last-Lecture.
- Class Object: contains variables such as Class-ID, Class-Total-Number-Of-Lectures, Class-Course-List that refers to each course with its instructors, Class-Last-Lecture that refers to the index of last lecture in the class timetable and Class-Number-Of-Last-Lecture that refers to the total number of last lectures.
- Course Object: contains variables such as Course-ID and Course-Type.
- Class-Course Object: contains some variables that link Class-ID, Course-ID, Number-Of-Lectures and Instructor-ID with the Instructor Object, Class object and Course object.

Once the required data is loaded and split into the said objects, the algorithm commences the school timetable generation. The ICTNL Function illustrates how many times each specific course is gone through for a specific instructor and how many times each specific course is gone through for a specific classroom. A loop selects the next object from Class-Courses list, and then finds the instructor object that has the same Instructor-ID. This is performed in order to add the number of lectures for the Class-Courses object to the total number of lectures of the instructor object and then the same procedure is repeated for the class object. It is quite evident that the complexity of this function is $O\left(CKO(\underset{\wedge}{\text{I}} + \text{Ç})\right)$. Where CKO is the number of class-course objects, which can be described as:

$$CKO = \sum_{d\in D}\sum_{s\in TS}\sum_{i\in I} v_{c,k}\ ; \quad \forall\ v \in V,\ \forall\ k \in K,\ \forall\ c \in C \tag{19}$$

$$= \sum_{k\in K}\sum_{c\in C} CKL\ ; \quad \forall\ k \in K,\ \forall c \in C$$

The total number of lectures for any instructor *i* is:

$$TL^{i} = \sum_{d\in D}\sum_{c\in C}\sum_{k\in K}\sum_{s\in TS}\ v_i\ ; \quad \forall\, v \in V,\ \forall\ d \in D,\ \forall\ c \in C,\ \forall\ k \in K\ ,\ \forall\ s \in TS \tag{20}$$

Similarly, the total number of lectures for any class *c* is:

$$TL^{c} = \sum_{d\in D}\sum_{k\in K}\sum_{i\in I}\sum_{s\in TS} v_c; \quad \forall\, v \in V,\ \forall\ d \in D,\ \forall\ k \in K,\ \ \forall\ i \in I,\ \ \forall\ s \in TS \tag{21}$$

From (10), the total number of lectures in the school is the summation of the total number of lectures of all classrooms, as given by:

$$\Omega = \sum_{c\in C} TL^{c}\ ; \quad \forall\, c \in C \tag{22}$$

As a result, the probability to schedule the lectures of the next instructor (*f+1*) is:

$$P^{f+1} = \left(\frac{1}{\underset{\wedge}{\text{I}} - f}\right)\left(\frac{\Omega - z - \sum_{1}^{f} TL^{f}}{\Omega}\right) \tag{23}$$

where *z* is the number of lectures reserved for instructor (*f+1*) in the process.

Once the total number of lectures is calculated, an ILL function will find the index for last lecture for each classroom, i.e., the time slot of the last lecture within the week, in order to initialize instructor's timetables and class timetables as well as that for the school as a whole. A new parameter is assigned $(O(\text{Ç}))$ to distributes the class lectures TL^{c} over the five weekly days even handedly to find the maximum time slots for one working day. Then, where there remain lectures unaccounted for, the ILL increases the maximum number of time slots by one, and distributes these lectures over there maining instructors starting from the first day of the week, such that:

$$ILL^{c} = \begin{cases} \frac{TL^{c}}{5}; & if\ mod\left(\frac{TL^{c}}{5}\right) = 0 \\ \frac{TL^{c}}{5} + 1; & otherwise \end{cases} \tag{24}$$

where ILL^{c} is the maximum number of time slots of class c for any working day. For the entire school, the index for the last lecture ILL^{s} is given by:

$$ILL^s = Max(ILL^c); \qquad \forall\ c \in C \tag{25}$$

The NLLI Function in general counts how many weekly lectures are scheduled to take place in time slot ILL^s for the whole school, and then it distributes these lectures amongst instructors even-handedly. This happens commensurate with the fairness criterion that warrants that the lecture distribution process is bias free devoid of any favoritism practices, per the soft constraints in section 2. As a result of this function, each instructor gets his/her own share of the Number of Last Lectures (NLL). First, the number of all lectures that take place ILL^s time slot across the school is given by:

$$NLL^s = \sum_{d=1}^{5} \sum_{c=1}^{Ç} ILL^c; \quad \forall\, c \in Ç \tag{26}$$

Then for all instructors, if instructor i teaches a class such that $ILL^c = ILL^s$, then the number of instructors who teach lectures in that classroom within that time slot is given by:

$$NoI = NoI + 1 \tag{27}$$

where there still exists cases of more lectures than there are instructors, then the remaining non-distributed lectures are distributed randomly across instructors, such that the random probability to select any one of them is:

$$RP^i = \frac{1}{NoI} \tag{28}$$

4.2. The initial state phase (ISP)

Initially, and as explained in [16, 17], the first step is to create a random initial solution and then perform additional operations on it with the objective of satisfying the hard constraints thereby arriving at the correct initial solution [18]. In fact, this has been a rather time-consuming method. In the proposed technique, the initial solution which is created in the initial state phase **(**ISP), keeps all hard constraints devoid from any violations so as to maintain integrity and correctness of the timetable structure throughout. The ISP process can be split into three main parts in a way that would underscore its rather important functions:

4.2.1. Common break array (CBA)

When an instructor is picked and a specific classroom is identified, a two-dimensional array called the common-break-array (CBA) is generated. First, the instructor timetable and the classroom timetable contain three types of time slots as shown in Figure 1(a) and 1(b): time slots reserved for the classroom (TSRC), time slots reserved by the instructor (TSRI) and also the remaining non-reserved time slots (NRTS). When the instructor timetable is overlapped with a classroom timetable, the common break array (CBA) will be generated as shown again in Figure 1(c). Now, the generated CBA has two types of Time Slots: a Non- Reserved time slot that is represented by Y/Yes and a reserved time slot (by either an instructor or a classroom) that is represented by an N/No as illustrated by (29).

$$CBA^c_{d,s} = TBL^i_{d,s} \bigcap TBL^c_{d,s} \tag{29}$$

where, $TBL^i_{d,s}$ is instructor i's timetable and $TBL^c_{d,s}$ is classroom c's occupancy timetable as given in (30) and (31), respectively.

4.2.2. Time slot selection (STS) function

The STS is an important function that offers several advantages in the proposed technique, amongst which that of minimizing the search space, per equation 30, by avoiding time slots with a label of (N) in the CBA; as a consequence, the execution time is further reduced. Furthermore, the STS function avoids simultaneous classroom and instructor conflicts (clashes) as it only searches for the set of time slots that still exhibit the label (Y) in the CBA, say YCBA, which means that these time slots are still Non-Reserved, as is given by (31).

$$\Omega^i\left(CBA^c_{d,s}\right) = \left|TBL^i_{d,s}\right| \times \left|TBL^c_{d,s}\right| \tag{30}$$

$$YCBA^c_{d',s'} : D' \rightarrow TS', \ d' \rightarrow s' \tag{31}$$

$$YCBA^{c}_{d',s'} \subseteq CBA^{c}_{d,s}$$

where $D'(d') = 1, \ldots, d̨$ is days that have $TS'(s') = 1, \ldots, s̨$ non reserved time slots.

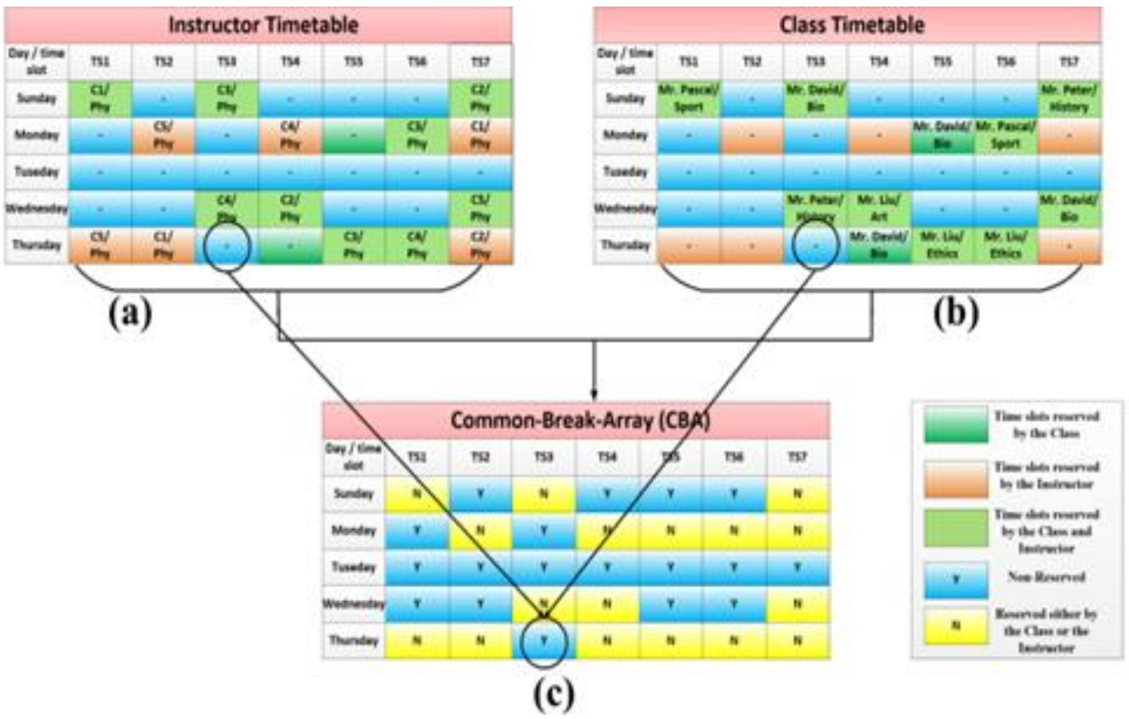

Instructor Timetable
Class Timetable
Common-Break-Array (CBA)
Day / time slot
TS1
TS2
TS3
TS4
TS5
TS6
TS7
Sunday
Monday
Tuesday
Wednesday
Thursday
(a)
(b)
(c)
Time slots reserved by the Class
Time slots reserved by the Instructor
Time slots reserved by the Class and Instructor
Non Reserved
Reserved either by the Class or the Instructor

Figure 1. Common-break-array (CBA) implementation

Once the CBA is generated per section 4.2.1, the (STS) adopts one random Non-Reserved Time Slot from the CBA in order to check if a particular timeslot is appropriate (valid) for the class/course. When this occurs, the course will be assigned to this time slot. As a result, this time slot will be mapped to the classroom and instructor timetables and updated in the CBA with an (*N*); otherwise this time slot will be updated by the CBA with a (*U*/Under demand). This procedure will be repeated until all time slots start exhibiting either (*N* or *U*). Figure 2 illustrates an example that schedules a *Phy* Course at three lectures per week. S0: Designate the CBA array that is created per section 4.2.1. S1: Divide the CBA into five nodes to represent the working days per week. S2: Remove all time slots with the label N (Reserved Time Slots). S3: In order to schedule the first lecture of the Phy course, the STS function designates TS3 for Thursday, TS3 will be implicitly updated with label (*N*) and then removed from search space. Note that this day becomes empty, and, consequently, the STS function neglects any further consideration of it and the search space is restricted only to four days. In S4: The STS continues to schedule the second lecture of the *Phy* course. It designates as TS4 for Sunday; TS4 is then updated implicitly with label (*N*) and then will, again, be removed from search space. S5: The STS designates TS2 for Tuesday in order to schedule the last lecture of the *Phy* course. TS2 is finally implicitly updated with a label of (*N*) which will consequently be removed from search space. S6: Displays the end result of the CBA process. In fact, such subset of CBA would only be available if the corresponding subsets of the instructor and classroom timetables are both available, such that

$$YCBA^{c}_{d',s'} = YTBL^{i}_{d',s'} \bigcap YTBL^{c}_{d',s}{}' \tag{32}$$

where

$YTBL^{i}_{d',s'} \subseteq TBL^{i}_{d,s}$ represents all available non-reserved time slots for instructor i's timetable which is filled by a$(-)$denotation.

$YTBL^{c}_{d',s} \subseteq TBL^{c}_{d,s}$ represents all available non-reserved time slots in class c's timetable which is filled with a$(-)$denotation.

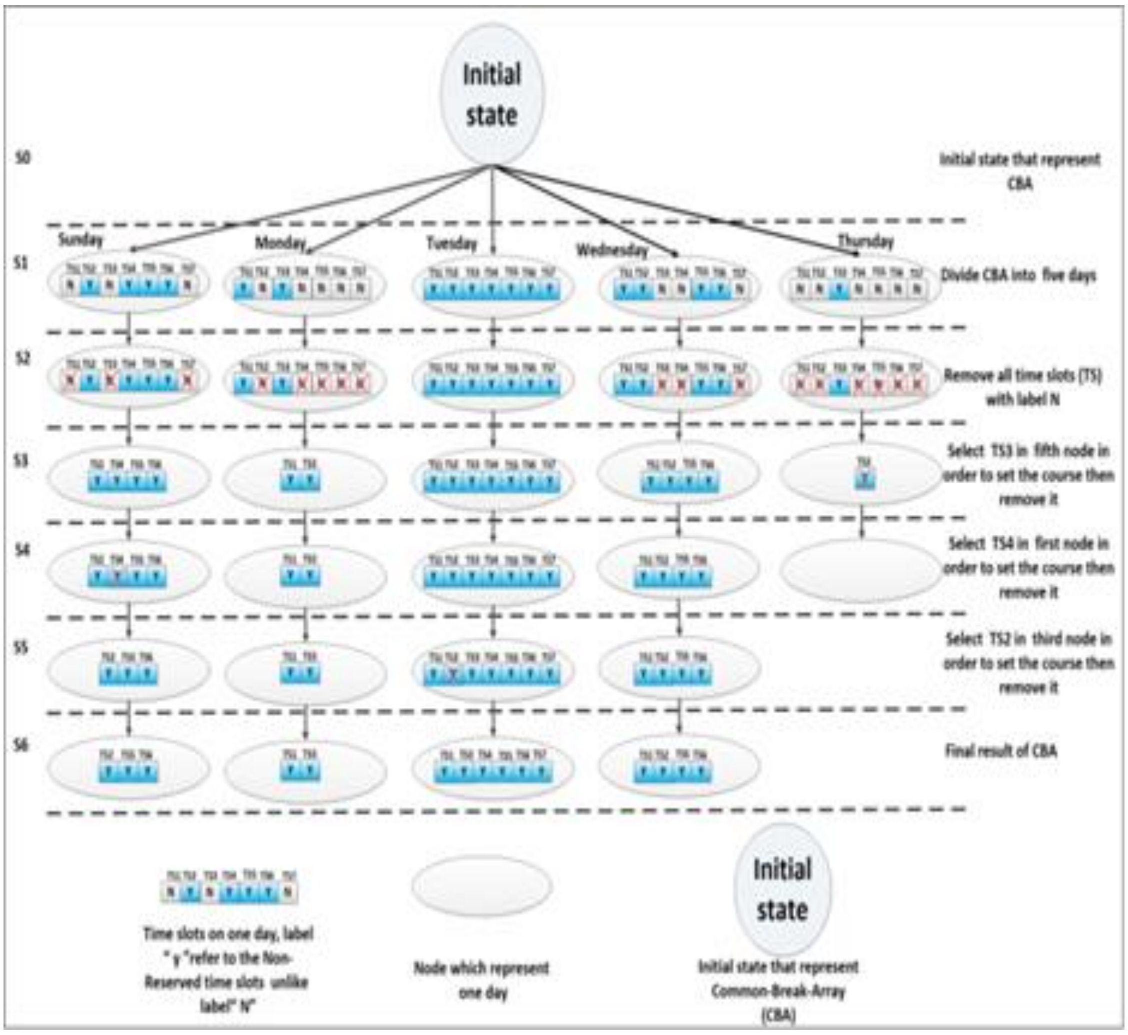

Initial state
S0
Initial state that represent CBA
Sunday
Monday
Tuesday
Wednesday
Thursday
S1
Divide CBA into five days
S2
Remove all time slots (TS) with label N
S3
Select TS3 in fifth node in order to set the course then remove it
S4
Select TS4 in first node in order to set the course then remove it
S5
Select TS2 in third node in order to set the course then remove it
S6
Final result of CBA
Time slots on one day, label " y "refer to the Non-Reserved time slots unlike label" N"
Node which represent one day
Initial state
Initial state that represent Common-Break-Array (CBA)

Figure 2. Select time slot (STS) function

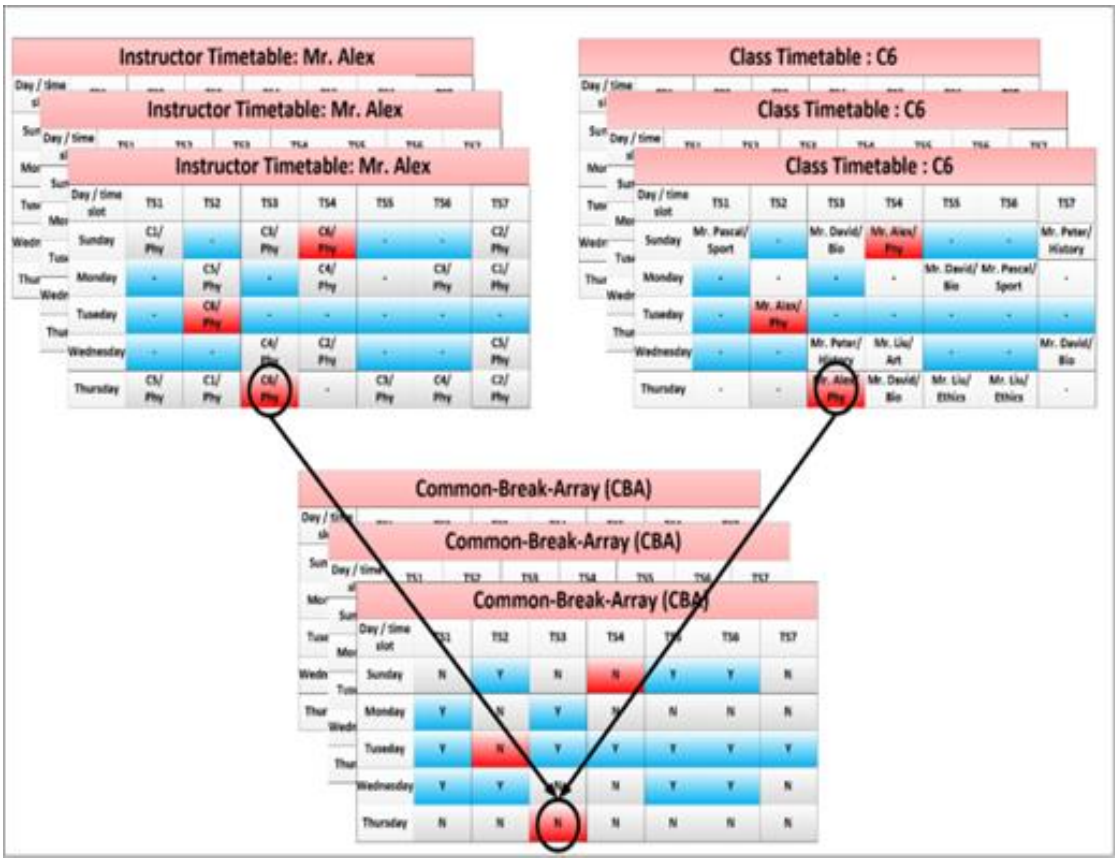
So, instead of searching $\sum_{d=1}^{5} s_d$, it is needed to search $\sum_{d=1}^{\mathrm{d}} s'_d$, where s_d is the total number of taught time slots during day d, and s'_d is the number of available non-reserved time slots for day d. The relative percent improvement can be measured as

$$Relative\ PercentImprovement\ (\varepsilon) = \frac{\left|\sum_{d=1}^{5} s_d - \sum_{d=1}^{\mathrm{d}} s'_d\right|}{\sum_{d=1}^{5} s_d} \times 100\% \qquad (33)$$

The worst case will manifest itself during the initial state where all timeslots for both the instructor and the classroom timetables would show as non-reserved status. On the other extreme, the best case will appear when both timetables reflect the same timeslot, in which case the relative percent improvement would approach 100%. As a result, the relative percent improvement achieved is:

$$\frac{\left|\sum_{d=1}^{5} s_d - \sum_{d=1}^{\mathrm{d}} s'_d\right|}{\sum_{d=1}^{5} s_d} \times 100\% < 100\% \qquad (34)$$

Now reverting to the *Phy* course which has three lectures per week, the STS function, in this case, will be repeated three times in the process of updating the three tables (Instructor Timetable, Class Timetable and CBA). In this example, the STS function commits TS4 for Sunday, TS2 for Tuesday and TS3 for Thursday, with a final result as shown in Figure 3, noting that the red slots are used to denote the time slots that have been changed.



Instructor Timetable: Mr. Alex

Day / time slot	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Sunday	C1/ Phy	-	C3/ Phy	C6/ Phy	-	-	C2/ Phy
Monday	-	C5/ Phy	-	C4/ Phy	-	C3/ Phy	C1/ Phy
Tuseday	-	C6/ Phy	-	-	-	-	-
Wednesday	-	-	C4/ Phy	C2/ Phy	-	-	C5/ Phy
Thursday	C5/ Phy	C1/ Phy	C6/ Phy	-	C3/ Phy	C4/ Phy	C2/ Phy

Class Timetable : C6

Day / time slot	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Sunday	Mr. Pascal/ Sport	-	Mr. David/ Bio	Mr. Alex/ Phy	-	-	Mr. Peter/ History
Monday	-	-	-	-	Mr. David/ Bio	Mr. Pascal/ Sport	-
Tuseday	-	Mr. Alex/ Phy	-	-	-	-	-
Wednesday	-	-	Mr. Peter/ History	Mr. Liu/ Art	-	-	Mr. David/ Bio
Thursday	-	-	Mr. Alex/ Phy	Mr. David/ Bio	Mr. Liu/ Ethics	Mr. Liu/ Ethics	-

Common-Break-Array (CBA)

Day / time slot	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Sunday	N	Y	N	N	Y	Y	N
Monday	Y	N	Y	N	N	N	N
Tuseday	Y	N	Y	Y	Y	Y	Y
Wednesday	Y	Y	N	N	Y	Y	N
Thursday	N	N	N	N	N	N	N

Figure 3. Final result after setting 3 lectures of phy course

4.2.3. Swap time slots function for two instructors (STSI)

When the STS function fails to select appropriate time slots from CBA, then it would be essential to execute the STSI function. This function deals with potentially emerging cases of two instructors showing at the same time and strives to swap around the time slots between them. As another example, Figure 4 outlines the process of scheduling a Math course with five lectures per week for instructor (Mr. *Jack*) and classroom (C6). The first four lectures for this Math course are already scheduled through the CBA process using the STS function (those colored as green), where the last lecture is shown to have failed, where it would require the help of the STSI function. The following steps serve to outline this process.

In S1, the process begins by loading Mr. Jack's and C6 timetables. S2 selects a Non-Reserved time slot from Mr. *Jack* timetable and fetches the corresponding course from the C6 timetable, in which case the Time Slot would be TS4 as colored in red where the corresponding course is Art). This is fulfilled by:

$$\langle d^{*}, s^{*}\rangle = YTBL_{d,s}^{i1} \bigcap TBL_{d,s}^{c} \tag{35}$$

where i_1 represents the instructor to be scheduled, which represents Mr. Jack in this case. The process then fetches the instructor who teaches this course as shown in S3, who happens to be *Mr. Liu*). After loading *Mr. Liu*'s timetable, the algorithm picks one Non-Reserved time slot from *Mr. Liu*'s and the C6 timetables that overlaps in same time slot; here, the time slot happens to be Ts2 where it is colored in blue. This is summed up by:

$$\langle d^{s}, s^{s}\rangle = YTBL_{d,s}^{i2} \bigcap YTBL_{d,s}^{c} \tag{36}$$

where i_2 represents the instructor whose lecture will be swapped, which represents that for *Mr. Liu*, in this case. In S4, now swap the contents of TS4 and TS2 for both *Mr. Liu*'s and the C6 Timetables so that TS4 becomes Non-Reserved, where the *Art* course will be set for TS2. Finally, in S5, set the *Math* course into the timeslot TS4 for both Mr. Jack and C6 timetables. At the end of this process, the last lecture for the Math course will be scheduled, such that:

$$TBL^{c}_{d^*,s^*} = \langle i_1, k_{i1} \rangle \tag{37}$$

$$TBL^{c}_{d^s,s^s} = \langle i_2, k_{i2} \rangle \tag{38}$$

As a result, the timetable for *Mr. Jack* is now updated to yield:

$$TBL^{i1}_{d^*,s^*} = \langle c, k_{i1} \rangle \tag{39}$$

And the timetable for *Mr. Liu* is updated to become:

$$TBL^{i2}_{d^s,s^s} = \langle c, k_{i2} \rangle \tag{40}$$

$$TBL^{i2}_{d^*,s^*} = \langle - \rangle \tag{41}$$

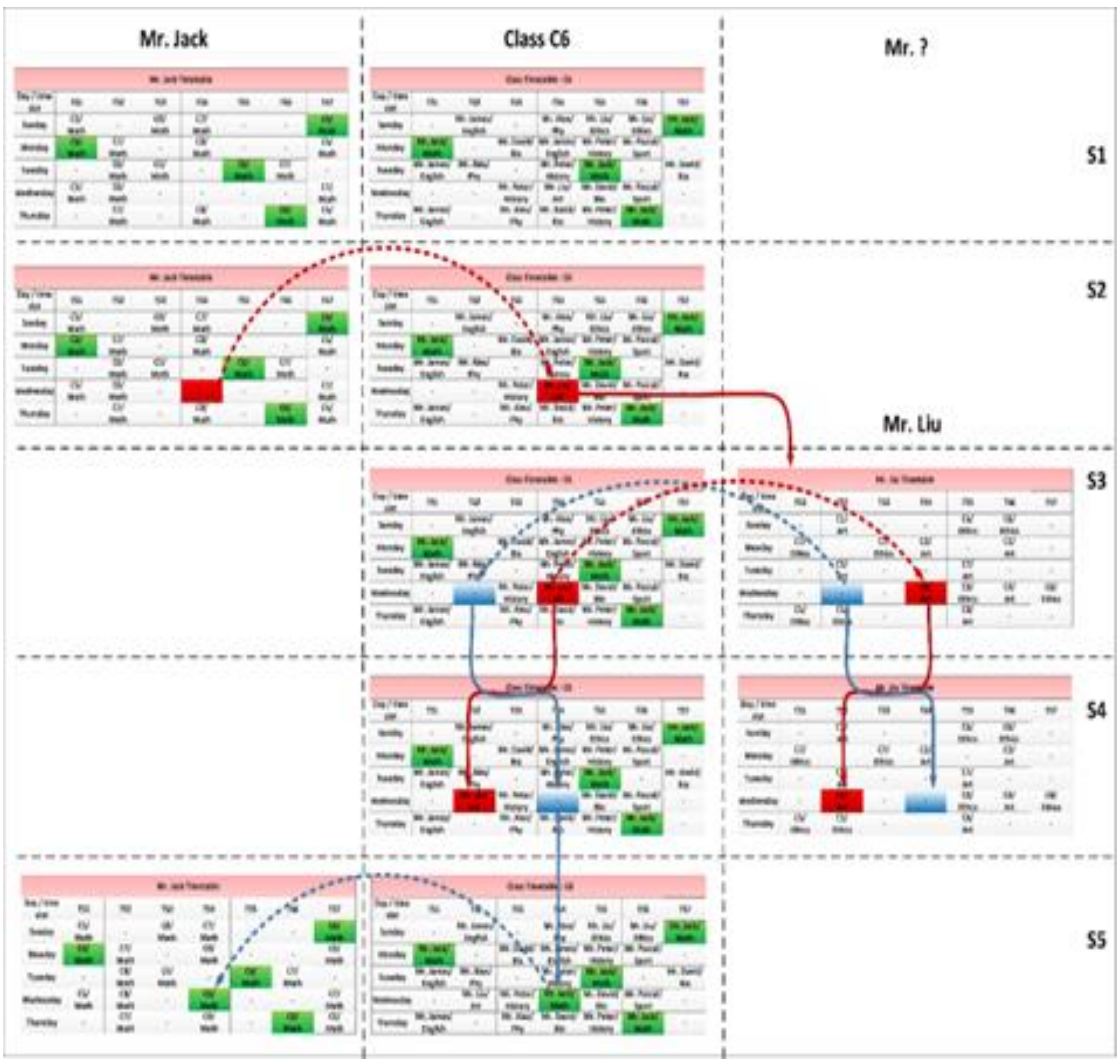
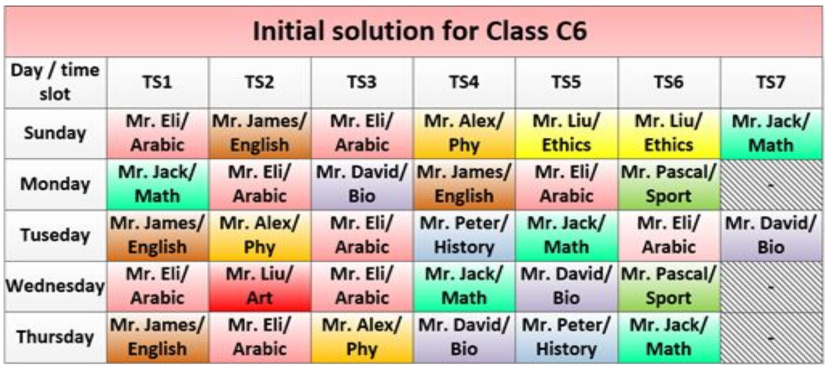


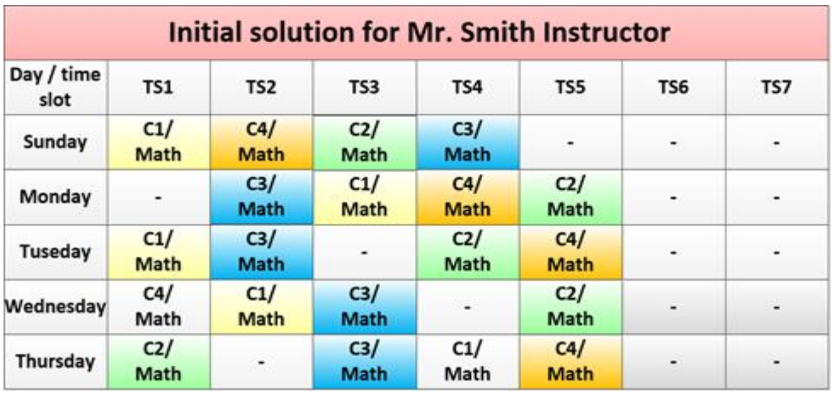
Figure 4. Swap time slots between two instructors (STSI) function

Going along the same procedure for all instructors, Figure 5 finally reflects the timetable for classroom C6 exhibiting all instructors for a whole week, while Figure 6 finally reflects the timetable for *Mr. Smith* alone. Towards the end of the ISP process, the number of instances for various operations is calculated such that the number of lectures to be scheduled, the number of iterations to contrive initial solutions and the number of set and swap actions in the various time slots in order to create all initial solutions would ultimately culminate in the total number of instances needed to generate the entire school initial state.



Initial solution for Class C6							
Day / time slot	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Sunday	Mr. Eli/ Arabic	Mr. James/ English	Mr. Eli/ Arabic	Mr. Alex/ Phy	Mr. Liu/ Ethics	Mr. Liu/ Ethics	Mr. Jack/ Math
Monday	Mr. Jack/ Math	Mr. Eli/ Arabic	Mr. David/ Bio	Mr. James/ English	Mr. Eli/ Arabic	Mr. Pascal/ Sport	-
Tuseday	Mr. James/ English	Mr. Alex/ Phy	Mr. Eli/ Arabic	Mr. Peter/ History	Mr. Jack/ Math	Mr. Eli/ Arabic	Mr. David/ Bio
Wednesday	Mr. Eli/ Arabic	Mr. Liu/ Art	Mr. Eli/ Arabic	Mr. Jack/ Math	Mr. David/ Bio	Mr. Pascal/ Sport	-
Thursday	Mr. James/ English	Mr. Eli/ Arabic	Mr. Alex/ Phy	Mr. David/ Bio	Mr. Peter/ History	Mr. Jack/ Math	-

Figure 5. Initial solution for class C6



Initial solution for Mr. Smith Instructor							
Day / time slot	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Sunday	C1/ Math	C4/ Math	C2/ Math	C3/ Math	-	-	-
Monday	-	C3/ Math	C1/ Math	C4/ Math	C2/ Math	-	-
Tuseday	C1/ Math	C3/ Math	-	C2/ Math	C4/ Math	-	-
Wednesday	C4/ Math	C1/ Math	C3/ Math	-	C2/ Math	-	-
Thursday	C2/ Math	-	C3/ Math	C1/ Math	C4/ Math	-	-

Figure 6. Initial solution for Mr. Smith instructor

4.3. Satisfied soft constraints phase (SSCP)

Even though the initial solution, as unveiled in section 4.2, satisfies all hard constraints, per the discussions of section 2, there would remain need to satisfy the soft constraints in order to increase algorithm efficiency. To do that there is imminent need to resort to the MHCA process. The SSCP phase is split into three parts: The L4 phase which distributes the total number of last lectures amongst instructors. The L5-L9 phase which is used to improve the instructor's timetable by accounting for the Instructor's preferences and the number of Daily work hours constraints as discussed in the soft constraints in section 2.The third phases panning L10-L13 is leveraged to improve the classroom timetable by considering the preferences in the classroom constraints as discussed in the soft constraints section of section 2.

4.3.1. The set scientific courses (SSC) function

The SSC function is an important function that deals with scientific courses and moves them to the first three time slots (early hours). Figure 7 illustrates an example of on improving Classroom C7 timetable using the SSC function:

- S0: The SSC function starts improvement by getting the C7 timetable.
- S1: SSC selects one day from the C7 timetable, say d^. In this example the day selected is Thursday.
- S2: SSC Separates all Courses in two main categories (*S: Math* and *Bio*) and colors them in pink while those (*NS: Sport, English, History, Arabic* and *Art*) are colored in blue, such that:

$$TBL^{c}_{d^{\wedge},s} = TBL^{c}_{d^{\wedge},s^{e}} + TBL^{c}_{d^{\wedge},s^{l}} \tag{42}$$

where s^e represents the early time slots, and s^l represents the late ones.

Further, the process separates the Time Slots in two segments: (TS1-TS3 and TS4- TS7), such that:

$$TBL^{c}_{d^\wedge,s} = TBL^{c}_{d^\wedge,s}(\Phi) + TBL^{c}_{d^\wedge,s}(\overline{\Phi}) \tag{43}$$

where Φ represents the scientific courses, and $\overline{\Phi}$ represents non-scientific courses.

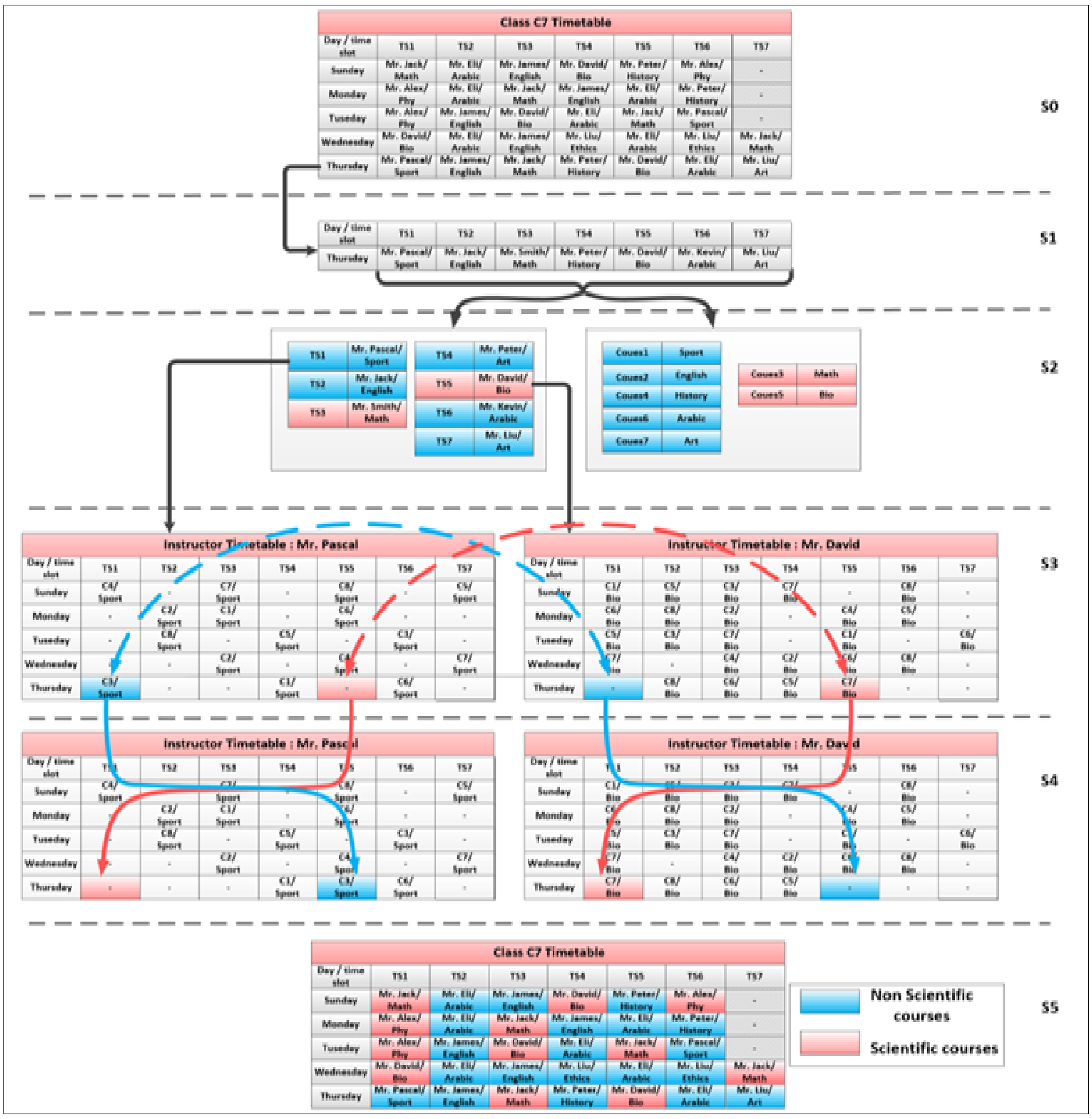


Figure 7. Set scientific courses (SSC) function

Then, SSC selects one S-Course from the second segment of the Time Slots (in this example is Bio in TS5), $\langle d^\wedge, s^{\Phi}\rangle$, and select one NS-Course from the first segment of Time Slots (Sport in TS1), $\langle d^\wedge, s^{\overline{\Phi}}\rangle$.

- S3: SSC fetches the corresponding instructor's timetable for each course; *Mr. Pascal* who teaches sports and *Mr. David* who teaches *Bio*. After that, it checks on whether TS5 is a Non-Reserved time slot in *Mr. Pascal's* timetable, and same for TS1 in *Mr. David's* timetable. When this happens; The SSC function goes right to S4 action. Otherwise, it starts out with the S2 action.

- S4: SSC swaps the courses between the TS1 and TS5 timeslots, so that the Bio course now takes place in TS1 while the Sport course in TS5, such that:

$$TBL^{i_\varphi}_{d^\wedge,s^\varphi} = TBL^{c}_{d^\wedge,s^\varphi}, \qquad iff\ TBL^{i_\varphi}_{d^\wedge,s^\varphi} \bigcap TBL^{c}_{d^\wedge,s^\varphi} = Y \tag{44}$$

$$TBL^{\overline{i}_{\varphi}}_{d^\wedge,s^{\overline{\varphi}}} = TBL^{c}_{d^\wedge,s^{\overline{\varphi}}}, \qquad iff\ TBL^{\overline{i}_{\varphi}}_{d^\wedge,s^{\overline{\varphi}}} \bigcap TBL^{c}_{d^\wedge,s^{\overline{\varphi}}} = Y \tag{45}$$

- S5: SSC updates classroom's C7 timetable.

By the end of the SSCP phase, the final solution for the entire School Timetable will be as generated. Moreover, the number of instances for each type of soft constraints are calculated such as: the number of scientific courses lectures to be considered in the early time slots, the number of non-scientific course lectures to be considered in the late time slots, the number of time slots to satisfy Instructor preferences, the number of time slot swaps required to satisfy the daily work hours constraint and the number of last lectures to be distributed amongst instructors.

5. RESULTS AND DISCUSSION

The work which was developed to contrive the algorithm underlying the proposed technique was based on Visual Studio and basically C#. It ran on personal computer of Intel Core i5, 4GB RAM, 0.5TB HDD using Windows 10. It was designed so that it would accept a variable number of instructors, classrooms, and courses. Various real examples were experimented with using this code taking into consideration primary and secondary schools in Jordan and mainly in the district of Irbid. The achieved results show that when primary schools were experimented with, different simulation parameters would be achieved than in the case of secondary schools. At primary school level, it was possible to perform a massive number of trials, where each run would cost 9 seconds of CPU time on average and actually consumedaround407.19 KB of the memory. Performance of this technique was measured with respect to four performance metrics:

a. The ratio of scientific lectures in the early time slots and the non-scientific lectures in the late time slots

In order to satisfy the constraint of scientific courses falling in early time slots, the proposed technique was tested for four schools with 10 runs for each school. Table 3 illustrates the distribution of course time slots. Here, it is evident that the smallest ratio of scientific courses in the early time slots was registered in School 1, with the largest value recorded in School 3. The smallest ratio of non-scientific courses in the late time slots was registered in School 1, with the largest ratio recorded in School 2.

These results signify that two primary factors actually have a pronounced bearing on this ratio:

- The ratio of scientific courses to non-scientific courses across the entire school.
- The ratio of early time slots in the school to the late time slots, as the early time slots are considered to be just the first three time slots, while the late might constitute the time slots that follow, depending, of course, on the overall number of time slots around the school.

Simulation results show that more than 75% of scientific courses were scheduled to convene during the early time slots and more than 25% of non-scientific courses were scheduled to take place during the late time slots. The simulations show also that the distribution of courses as a function of the time slots follows a normal distribution, where the x-axis represents the number of lectures as shown where the curves are centered on their averages. Some of these curves manifest overlapping characteristics in ways that would be attributed to aforementioned factors that govern the distribution; such behavior appears more clearly in the case of School 2, where the curve for the early scientific courses and that for the early non-scientific courses overlap fairly completely.

Results show the normal distribution of early scientific and late non-scientifc courses over the four schools that were subject of the study. As shown, 65.44% of scientific courses in all schools were distributed duringearly time slots, while 58.21% of non-scientific courses in all schools were distributed during thelate time slots.

b. The improvement in satisfying more soft constraints under each optimization, iteration as measured by a reduction in the heuristic value.

By going through one more iteration, in this case from 1 to 2, the heuristic value is shown to increase dramatically, where it is shown to increase by 50%. Now by jumping from iteration 2 to 4, the heuristic value increases by 17%. Incrementing the number of iterations, yet more, from 4 to 8, the simulation shows that the heuristic value remains fairly quite the same, which means that during these iterations, no changes are bound to take place. On the 9th iteration, the heuristic value is increased by 12% which readily implies that significant and less pronounced changes are taking place, where the heuristic value remains the same up until the 14th iteration. At the 15th iteration the heuristic value gets increased by 23% which means again that the quality of the "best" solution as represented by the heuristic value becomes rather limited. After the 15th iteration, the heuristic value remains the same at -17 which means that the best solution is reached at that point.

c. Acceptability

In order to show the validity, usability and quality of this tool, 20 instructors who were actually involved in the preparation of the school timetable in 6 schools, were solicited to attend to a questionnaire entailing 9 items where their responses to a special questionnaire designed for this purpose. These responses, as shown, readily reflect the satisfaction of the constituents involved with the outcome of the research work, as presented in this paper.

6. CONCLUSION

The work proposed in this research effort has tackled the manual school scheduling (timetable scheduling) problem by developing a system for administrators' uses at schools to help them develop the school timetable for each semester and have it readily for the upcoming semester in a more automated manner. The proposed system was developed using various supporting technologies including SQL Server 2008, Visual Studio .NET and ASP .NET. This system leveraged the Hill Climbing Algorithm which provided good results for different school sizes under various hard and soft constraints. The proposed technique was found to save more time and effort by contriving multiple alternate versions of the school timetable in minutes, with the end result of creating more handy timetables at reduced time costs.

ACKNOWLEDGEMENTS

This paper is funded by deanship of research, Jordan University of Science and Technology, grant no. 145/2018.

REFERENCES

[1] A. Hiendro, "Projectile-target search algorithm: a stochastic metaheuristic optimization technique," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3772-3778, 2019.

[2] Erwin, et al., "Hybrid Multilevel Thresholding and Improved Harmony Search Algorithm for Segmentation," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4593-4602, 2018.

[3] A. Schaerf, "Local search techniques for large high school timetabling problems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 4, pp. 368-377, 1999.

[4] I. Chorbev, et al., "Solving the High School Scheduling Problem Modelled with Constraints Satisfaction using Hybrid Heuristic Algorithms," *Greedy Algorithms Open Access Book, InTech Publisher*, 2008.

[5] T. Song, et al., "An iterated local search algorithm for the University Course Timetabling Problem," *Applied Soft Computing*, vol. 68, no. 7, pp. 597-608, 2018.

[6] J. A. Soria-Alcaraz, et al., "Iterated local search using an add and delete hyper-heuristic for university course timetabling," *Applied Soft Computing*, vol. 40, no. 5, pp. 581-593, 2016.

[7] Y. Bykov and S. Petrovic, "A step counting hill climbing algorithm applied to university examination timetabling," *Journal of Scheduling*, vol. 19, no. 4, pp. 479-492, 2016.

[8] S. Eftekhari and M. O. Sadegh, "The effect of load modelling on phase balancing in distribution networks using search harmony algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 3, pp. 1461-1471, 2019.

[9] S. Abdullah and H. Turabieh, "Generating university course timetable using genetic algorithms and local search," in *Third International Conference on Convergence and Hybrid Information Technology (ICCIT'08)*, vol. 1, pp. 254-260, 2008.

[10] A. Rahimi, et al., "Using meta-heuristics for project scheduling under mode identity constraints," *Applied Soft Computing*, vol. 13, no. 4, pp. 2124-2135, 2013.

[11] K. Paul and N. Kumar, "Cuckoo Search Algorithm for Congestion Alleviation with Incorporation of Wind Farm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4871-4879, 2018.

[12] P. A. Sonawane and L. Agha, "Hybrid Genetic Algorithm and TABU Search Almgorithm to Solve Class Time Table Scheduling Problem," *International Journal of Research Studies in Computer Science and Engineering*, vol. 1, no. 4, pp 19-26, 2014.

[13] S. M. J. Moghaddam and S. Alipour, "Resource allocation in cloud computing using advanced imperialist competitive algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, pp. 3286-3297, 2019.

[14] F. F. Yeng, et al., "The saturation of population fitness as a stopping criterion in genetic algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 4130-4137, 2019.

[15] A. K. Ariyani, et al., "Hybrid Genetic Algorithms and Simulated Annealing for Multi-trip Vehicle Routing Problem with Time Windows," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4713-4723, 2018.
[16] L. A. Bewoor, et al., "Comparative Analysis of Metaheuristic Approaches for Makespan Minimization for No Wait Flow Shop Scheduling Problem," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 1, pp. 417-423, 2017.
[17] R. Kumar and R. Kumar, "Optimizing requirement analysis by the use of meta-heuristic in search-based software engineering," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 4336-4343, 2019.
[18] A. Schaerf, "Tabu search techniques for large high-school timetabling problems," *Technical Report, Centre for Mathematics and Computer Science*, Amsterdam, Netherlands, pp. 363-368, 1996.
[19] I. A. Joundan, et al., "A new efficient way based on special stabilizer multiplier permutations to attack the hardness of the minimum weight search problem for large BCH codesIssam," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 2, pp. 1232-1239, 2019.
[20] W. Shao, et al., "An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem," *Applied Soft Computing*, vol. 61, no. 12, pp. 193-210, 2017.
[21] R. Raghavjee and N. Pillay, "Using genetic algorithms to solve the South African school timetabling problem," in *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 286-292, 2010.
[22] K. Nguyen, et al., "Simulated annealing-based algorithm for a real-world high school timetabling problem," *Second International Conference on Knowledge and Systems Engineering (KSE)*, pp. 125-130, 2010.
[23] A. Cerdeira-Pena, et al., "New approaches for the school timetabling problem," *Seventh Mexican International Conference on Artificial Intelligence*, pp. 261-267, 2008.
[24] Y. Zheng, et al., "Quantum-inspired genetic evolutionary algorithm for course timetabling," *Third International Conference on Genetic and Evolutionary Computing*, pp. 750-753, 2009.
[25] A. N. Fauziyah and W. F. Mahmudy, "Hybrid Genetic Algorithm for Optimization of Food Composition on Hypertensive Patient," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4673-4683, 2018.