Appl Intell (2011) 34: 299–310
DOI 10.1007/s10489-009-0196-5

Scatter search technique for exam timetabling

Nashat Mansour · Vatche Isahakian · Iman Ghalayini

Published online: 30 September 2009

Abstract At universities where students enjoy flexibility in selecting courses, the Registrar's office aims to generate an appropriate exam timetable for numerous courses and large number of students. An appropriate, real-world exam timetable should show fairness towards all students, respecting the following constraints: (a) eliminating or minimizing the number of simultaneous exams; (b) minimizing the number of consecutive exams; (c) minimizing the number of students with two or three exams per day (d) eliminating the possibility of more than three exams per day (e) exams should fit in rooms with predefined capacity; and (f) the number of exam periods is limited. These constraints are conflicting, which makes exam timetabling intractable. Hence, solving this problem in realistic time requires the use of heuristic approaches. In this work, we develop an evolutionary heuristic technique based on the scatter search approach for finding good suboptimal solutions for exam timetabling. This approach is based on maintaining and evolving a population of solutions. We evaluate our suggested technique on real-world university data and compare our results with the registrar's manual timetable in addition to the timetables of other heuristic optimization algorithms. The experimental results show that our adapted scatter search technique generates better timetables than those produced by the registrar, manually, and by other meta-heuristics.

Keywords Evolutionary algorithm · Exam timetabling · Meta-heuristics · Multi-criteria optimization · Scatter search

N. Mansour (✉) · V. Isahakian · I. Ghalayini
Dept. of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon
e-mail: nmansour@lau.edu.lb

I. Ghalayini
e-mail: iman.ghalayini@lau.edu.lb

V. Isahakian
Dept. of Computer Science, Boston University, Boston, MA, USA
e-mail: vatchei@hotmail.com

1 Introduction

Timetabling of final exams for large numbers of courses and students is done at universities every semester. Often, such exam timetables lead to conflicts and/or unfairness for many students. For example, conflicts occur where simultaneous exams are scheduled for the same student, and unfairness to a student refers to consecutive exams or more than two exams on the same day. A good exam timetable would aim to minimize conflicts and the unfairness factors. Such a timetable is usually also subject to the constraints of having predefined number of exam time slots and limited-capacity classrooms.

The university timetabling problem is known to be NP-complete [14, 40] and a number of heuristic techniques and algorithms have been proposed for providing good sub-optimal solutions. Examples of these algorithms and techniques are: graph coloring [9, 11]; clustering and clique algorithms [12, 25]; constraint-based techniques [32, 38]; neighborhood search based techniques [1]; tabu search [16, 22]; simulated annealing and its great deluge variant [8, 28, 42]; genetic and evolutionary algorithms [7, 13, 15, 41, 43]; ant and artificial immune algorithms [17, 26]; multi-criteria techniques [34]; hybrid heuristics and hyper-heuristics [2, 4, 5, 35, 36, 39]; multi-phase technique that involves search and great deluge algorithms [33].

An excellent, recent survey of exam timetabling techniques is presented in Qu et al. [37].

Clearly, the exam timetabling literature includes a wide variety of approaches and algorithms and different versions of these algorithms. It also presents a variety of objectives and constraints for the problem [37] and, thus, different ways to assess solution quality. This variety is based on theoretical assumptions as well as diverse requirements found, in reality, among different institutions even within the same country [6]. Hence, techniques that address real-world problems based on real-world data would be useful for bridging the gap between research and practice [30]. In this context, of particular importance is the recent International Timetabling Competition (ITC2007), which aimed to narrow the gap between researchers and practitioners by testing their techniques on real-world models [31]. In ITC2007, exam timetabling solutions are considered feasible if they satisfy all hard constraints. Then, the quality of the solution is measured in terms of soft constraints satisfaction. Each soft constraint violation is weighted be a penalty coefficient. Hard constraints include: no exam conflicts, no room capacity is exceeded, and no period length is violated. Soft constraints include: 2 exams in a row, 2 exams in a day, spread of exams, number of larger exams appearing later in the timetable, and period- and room-related constraints.

In this work, we formulate the exam timetabling problem based on real-world university data with specific objectives and constraints that aim to address conflicts and unfairness factors. These specific objectives and constraints differ from the ITC2007's model and are described in detail in the next section. We propose a Scatter Search technique (SS) for solving the problem. Scatter search is a fairly recent approach [19, 20] that combines features from previous meta-heuristics. Scatter search is an evolutionary framework that has been adapted to hard optimization problems in a variety of settings [23]. It operates on a set of solutions and combines these solutions to create new ones. In contrast to other evolutionary approaches, scatter search is founded on the premise that systematic design and methods for creating new solutions provide benefits beyond those derived from mere randomization. It uses strategies for search diversification and intensification. Scatter search is adapted, for the first time, in this paper for the exam timetabling problem and is compared with other meta-heuristic approaches based on the same real-world objectives and constraints. The experimental results clearly show that SS provides good solutions.

The rest of the paper is organized as follows. Section 2 presents the problem formulation and the objective function. Section 3 describes the SS technique for solving the final exam timetabling problem. Section 4 presents and discusses the experimental results. Section 5 concludes the paper.

2 Exam timetabling problem and objective function

Every semester, university students enroll in individual courses for which they have to sit for exams at the end of the semester. Usually, different students, even in the same major, select (some) different courses or sections of courses. The exam timetabling problem consists of assigning a total of A exams to Π ($= B \times E$) exam periods and R classrooms, where B is the number of exam days and E is the number of exam periods per day. The objective is to minimize the conflict and the unfairness factors, which are: (i) The number of students having simultaneous exams, (ii) The number of students having consecutive exams, and (iii) The number of students taking three or more exams on the same day.

In this work, we assume the following:

(i) A limited number of exam periods, Π, is predefined.
(ii) A limited predefined number of classrooms, R, are available for exams.
(iii) Room capacities ($\Psi_1, \Psi_2, \ldots, \Psi_R$) are used in assigning exams to rooms. Also, more than one exam can be assigned to the same room at the same time, if they fit.
(iv) The last period of one day and the first period of the next day are considered to be consecutive.
(v) The user shall be provided with the flexibility of assigning weights to the different conflicts and unfairness factors.

Let $c(v)$ be the scheduled period of exam v, and $\xi = \{1, 2, \ldots, \Pi\}$ be the set of ordered, available periods. The weight, w_{ij}, represents the number of students enrolled for both exams i and j. The objective function, OF, is given in terms of the following factors:

(i) S_{SE}, the total number of students having conflicting simultaneous exams. That is, $S_{\mathrm{SE}} = \sum w_{ij}$ for all i and j such that $c(i) = c(j)$.
(ii) S_{CE}, the total number of students taking consecutive exams. That is, $S_{\mathrm{CE}} = \sum w_{ij}$ for all i and j such that $|c(i) - c(j)| = 1$.
(iii) $S_{3\mathrm{E}}$, the total number of students taking three exams per day. That is, $S_{3\mathrm{E}}$ is the total number of students, each taking exams in the periods $c(i)$, $c(j)$, and $c(k)$ such that $c(i) \operatorname{div} E = c(j) \operatorname{div} E = c(k) \operatorname{div} E$, where div refers to integer division.
(iv) $S_{4\mathrm{E}}$, the total number of students having more than three exams per day.
(v) $\rho_{rk} = 1$ if the capacity of room r is exceeded in period k, i.e. if room r was assigned a larger number of students than Ψ_r (capacity of room r); otherwise, it is equal to zero.

Therefore, the objective function to be minimized can be expressed as:

Table 1 Summary of useful notations

Symbol	Description
A	Total number of exams taken by all students
B	Number of exam days
E	Number of exam periods per day
Π	Number of available exam periods
R	Number of classrooms available
ξ	The set of available exam periods; $\lvert\xi\rvert = \Pi$
$c(i)$	Period to which exam i is assigned
S_{SE}	Total number of students taking conflicting simultaneous exams
S_{CE}	Total number of students taking consecutive exams
$S_{3\mathrm{E}}$	Total number of students taking 3 exams per day
$S_{4\mathrm{E}}$	Total number of students taking four or more exams per day
ρ_{rk}	Tells if capacity of room r is violated at period k
Ψ_r	Capacity of room r
α	Weighting factor related to the importance of S_{SE} in OF
φ	Weighting factor related to the importance of S_{CE} in OF
σ	Weighting factor related to the importance of $S_{3\mathrm{E}}$ in OF
β	Weighting factor related to the importance of $S_{4\mathrm{E}}$ in OF
γ	Weighting factor related to the importance of the room capacity term in OF
w_i	Number of students enrolled for exam i
w_{ij}	Number of students enrolled for both exams i and j

$$\mathrm{OF} = \alpha S_{\mathrm{SE}} + \varphi S_{\mathrm{CE}} + \sigma S_{3\mathrm{E}} + \beta S_{4\mathrm{E}} + \gamma\left(\sum_{1\le k\le\Pi}\sum_{1\le r\le R}\rho_{rk}\right) \quad (1)$$

where $\alpha, \varphi, \sigma, \beta$ and γ are user-defined weights for simultaneous exams, consecutive exams, multiple exams, and room capacity violations, respectively. The inner summation in the last term of expression (1) gives the total number of rooms violated in a period, whereas, the outer summation gives to the total number of rooms violated in all periods. The useful symbols used are summarized in Table 1.

We note that different values can be assigned to the weights in OF. These weights, which are similar to the concept of institutional model index of ITC2007 [31], allow flexibility in using our proposed solution algorithms to suit the user's particular choices or requirements for different instances of the problem.

3 Scatter search technique for exam timetabling

Scatter Search (SS) is an evolutionary metaheuristic approach that can be applied to hard optimization problems [18, 19]. It operates on a population of candidate solutions which is small compared to other evolutionary algorithms. It tries to diversify solution generation by not including the same solution twice in the solution pool. It has the ability to combine solutions and create new ones by using diversification strategies that do not merely rely on randomization [21, 23, 29]. Scatter search starts with generating a random set of candidate solutions that includes a good amount of diversity. From this initial set, we select a small subset, called the Reference Set (RefSet). Then, SS iterates through four methods: (a) Subset Generation method that generates subsets (of specifies sizes) of candidate solutions found in the RefSet; (b) Solution Combination method that generates new solutions based on combining the solutions defined in the subsets of the previous step; (c) Improvement method that aims to improve the quality of the combined solutions, if possible; (d) RefSet Update method that updates the limited number of candidate solutions found in the RefSet by replacing some or all of them by solutions that are selected from the improved candidates. Figure 1 shows the basic SS paradigm and the following subsections describe how we adapt SS to solve the exam timetabling problem.

3.1 Solution representation

The basic data structures used throughout the implementation of the SS technique consist of one vector and two matrices. The vector is a one dimensional array CRS(A) that defines the basic exam timetable, Where A is the number of exams. The element CRS(i) represents the exam period allocated to exam i where the value ranges from $\{1, \ldots, \Pi\}$. The first matrix ROOMS(A, Π) defines the basic exam timetable for room allocations. The element ROOMS(i, j) allows for a value ranging from $\{1, \ldots, R\}$. A value of x in ROOMS(i, j) implies the assignment of exam i to period j in room x. Every generated exam timetable is represented by CRS and ROOMS.

3.2 Diversification generation method

The population generation method aims to generate good starting point solutions for solving the exam timetabling problem. It tries to strike a balance between diversification (randomization) and intensification. To achieve this, we implement frequency memory and controlled randomization. Controlled randomization is favored with respect to pure randomization based on previous published experiences [10, 24].

We divide the total number of exam periods into 4 equal-size sub-ranges, i.e., each sub-range corresponds to $\Pi/4$ exam periods, assuming that Π is a multiple of 4. Each solution in P is constructed by applying the following steps to all exams i in $\{1, 2, \ldots, A\}$:

- Select a sub-range j randomly (from 1–4), where the probability of selecting j is inversely proportional to its frequency count. Initially, when the frequency count is zero for all sub-ranges, a sub-range is randomly selected.
- Generate an exam period randomly from the sub-range j.

Fig. 1 Outline of scatter search

1. Use the Diversification Generation Method to generate a set of candidate solutions P, each solution is referred to as y.
2. Apply the Improvement Method to y. The improved solution is referred to as x. If $x \notin P$ then x will replace y in P, otherwise, keep y in P. Repeat this step for all y in P.
3. Build RefSet $= \{x_1, \ldots, x_b\}$ with b solutions from P and sort them according to their objective function values such that x_1 is the best solution and x_b the worst. Set NewSolutions = TRUE.

while (NewSolutions) do

4. Generate NewSubsets, from solutions in RefSet.
   NewSolutions = FALSE.
   while (NewSubsets $\neq \emptyset$) do
      5. Select the next subset s in NewSubsets.
         Apply Solution Combination method to s to obtain a new solution y.
      6. Apply the Improvement Method to y and obtain x.
      7. if (x is not in RefSet and $f(x) < f(x_b)$) then
         8. Make $x_b = x$ and reorder RefSet.
         9. Set NewSolutions = TRUE.
         end if
      10. Delete s from NewSubsets.
   end while

end while

The number of times a sub-range j is chosen to generate an exam period for exam i is stored in a matrix freq(i, j), with $i = 1, 2, \ldots, A$ and $j = 1, 2, 3, 4$. This frequency value influences the generation of exam periods in future iterations. For every exam i in a candidate solution, the diversification generation method starts by determining a sub-range value by using controlled randomization combined with the inverse frequency. That is, sub-range(i) is the value j that corresponds to the minimum value of freq(i, j) for $j = 1, 2, 3, 4$, which favors sub-ranges that were not previously selected for this particular exam. Then, exam i is assigned to a randomly selected period within sub-range(i) and freq(i,sub-range(i)) is incremented. Also, a room is randomly selected from the set of available rooms (without replacement) and is assigned to exam i. These steps are repeated for every one of the A exams in a candidate solution.

The whole procedure is also repeated until $|P|$ candidate solutions are produced. Every solution generated is checked for duplication, since one of the characteristics of SS is not to start exploring the solution space from the same initial point solutions. If a duplicate is found, it is rejected and the procedure will be repeated to produce a replacement.

3.3 Improvement method

The improvement method follows the generation of a new solution using the diversification generation method or the combination method. The improved solution replaces the original one in P. Our improvement method is based on neighborhood search. Specifically, it is based on 'FIRST' move, where you move an element to the first location that improves the solution.

The improvement method starts by sorting the exams in terms of decreasing number of enrolled students, w_i, for $i = 1, 2, \ldots, A$. It selects an exam (assigned to period i) with the largest weight and reassigns it to the first period j that would decrease the value of OF, where j ranges between 1 and Π. This procedure is known as the Largest Enrollment First heuristic and has been found to be a favorable one with respect to other similar heuristics [3]. Whether period j is found or not, the algorithm considers the next exam in the sorted order. These steps are repeated until all the exams are visited. Once all the exams are visited, the same steps are repeated again as long as there is a change in an exam which leads to an improvement in the OF.

Every time an exam is reassigned to a new period, we try to reassign it a room using the following procedure. Randomly select a room, without replacement, from the range $\{1, 2, \ldots, R\}$, swap the exams assigned to the current room and the selected one, and check the change in the OF value. If the change is negative, then accept the room reassignment; otherwise, repeat the random room selection step. If none of the selected rooms lead to a decrease in the OF value, the initial room assignment is kept.

3.4 Reference set update method

The Reference Set update Method consists of two cases. The first case is to determine the RefSet from the initial population P. The second case is to maintain the RefSet with

"Best" solutions generated from the improvement method. The concept of "Best" does not only represent solutions with good quality but also represents solutions with good diversity.

After creating the initial population P using the diversification generation method, we determine the reference set by selecting solutions from the initial population. The number of solutions in the reference set should be relatively smaller than the entire population of P; we use RefSet Size $= 0.1 \times |P|$.

We divide the RefSet into two subsets HQRefSet for high quality solutions and DivRefSet for diverse solutions, such that RefSet = HQRefSet ∪ DivRefSet, and assign two counters: b_1 for HQRefSet and b_2 for DivRefSet having $|\text{RefSet}| = b_1 + b_2 = b$. We select b_1 best solutions from the initial population and assign them to the HQRefSet; that is, these solutions have the smaller values for OF. We remove these solutions from the initial population. b_2 DivRefSet Items are, then, selected from the initial population based on diversity and not quality. To measure the diversity of a solution, we use a heuristic function $\partial(x', x'')$ that gives an approximate distance between two solutions x' and x''. The distance $\partial(x', x'')$ is defined as the number of different exam assignments (to periods) that appear in solution x' and solution x''.

To select the b_2 (DivRefSet) solutions from P, for all the remaining solutions x in P, we define $\partial_{\min}(x)$ as the minimum distance between solution x and all the solutions x' in the HQRefSet. After ordering the solutions based on the minimum distance $\partial_{\min}(x)$, we select the b_2 Items that have the highest distance values and add them to the DivRefSet. This results in a RefSet count of $(b_1 + b_2)$, which is equal to $0.1 \times |P|$. Once we construct the Reference Set, the initial population items are no longer used.

In the second case, following the sequence of subset generation, combination and improvement methods, the generated solution needs to be examined for possible addition to the reference set for future use in subset generation. The Reference Set Update method is responsible for maintaining the reference set. The criterion for accepting a newly generated solution, x, is based on the following:

- Check whether the OF of x is better than X_{worse}, the worst solution in the HQRefSet. If it is, then add it to replace X_{worse} and reorder the HQRefSet based on their OF values.
- If the OF of x is greater than X_{worse}, then check if it can be added to the DivRefSet. If the distance $\partial_{\min}(x)$ to all the solutions in the HQRefSet is greater than that of the worst (least diverse) solution in DivRefSet, then add x to DivRefSet to replace the current worst solution.
- If the $\partial_{\min}(x)$ is greater than that of the worst solution of DivRefSet, i.e. x is a bad diverse solution, then reject x.

3.5 Subset generation method

For the Subset generation method, we adopt the design suggested by Glover [18], which refers to SubsetType = 1 and 2 with the following characteristics.

- SubsetType 1: all 2-element subsets. A 2-element subset is derived from the combination of all possible pairs of exam timetable solutions, (S_1, S_2), in the RefSet to generate new subsets.
- SubsetType 2: 3-element subsets derived from the 2-element subsets by augmenting each 2-element subset to include the best solution not in this subset. That is, a 3-element subset is derived from the combination of the 2-element subsets with the addition of the best solution S_3 such that $S_3 \notin \{S_1, S_2\}$.

All 2-element and 3-element subsets are considered in the next Solution Combination Method. Generating these subsets is time-consuming. This is why we refrain from generating higher order combinations such as Subset Types 3 and 4.

3.6 Solution combination method

This method uses the subsets generated using the subset generation method with the aim of creating new solutions. In designing the solution combination method, the objective is to strike a balance between solution intensification and diversification. The combination method is a greedy heuristic procedure that is based on a scoring mechanism and relies on randomization as a tie breaker.

First, a blank timetable S is defined, to which exam periods will be assigned. We select exams, i for $i = 1, 2, \ldots, A$, one by one in decreasing order of the number of students (w_i) enrolled for these exams. Then, we consider solutions S_1 and S_2 from the 2-element subsets generated in the previous step. The two periods, p_1 and p_2, and associated rooms to which exam i is assigned in S_1 and S_2, respectively, are assessed. The partial objective function values due to both assignments (p_1 and p_2) in S are computed. The partial objective function includes all choices made so far. The period (and room) that corresponds to lower objective function value is accepted for S. This process is repeated for all exams. In case of ties in the scores, we break the tie by selecting a period randomly.

The same process is performed for 3-element subsets by choosing for S from three exam periods found in the three solutions making up a subset.

The way the solution combination method works implies that using both SubsetTypes 1 and 2 offers more diverse combinations. Despite the intensification tendency that results from including the best solution in most of the 3-element subsets, diversity is not much reduced, since the

solution combination method considers one exam at a time and selects the period (and associated room) that yields the best partial objective function. A particular period that gives the best partial objective function at some instance in this process for a particular 3-element subset might fail to be selected in another 3-element subset. Therefore, we will obtain a larger pool of combined solutions to update the RefSet with.

4 Experimental results

4.1 Experimental procedure

In this section, we present the results of SS, and compare them with those of simulated annealing (SA), genetic algorithm (GA), 3-phase simulated annealing (3PSA), and extensive manual timetabling. Manual timetabling is accomplished by a few staff members of the registrar's office who work for several days on a timetable, which is given to instructors and students for feedback. After this feedback, the initial timetable is adjusted to produce the final version of the manual timetable, the results of which are reported in this section. These algorithms are applied to real data of the Lebanese American University (LAU) obtained from four semesters, which lead to four subject problem instances, described in Table 2.

The exam timetables produced by the algorithms are evaluated in terms of the five metrics: number of students taking simultaneous exams (S_{SE}), number of students taking consecutive exams (S_{CE}), number of students taking three exams per day ($S_{3\mathrm{E}}$), number of students taking more than 3 exams per day ($S_{4\mathrm{E}}$), and total number of rooms assigned with more students than their capacities over all exam periods. We apply the algorithms to the four subject problems with typical number of exam periods (at the university where data is obtained), namely $\Pi = 32$ or 36 periods for $B = 8$ or 9 days and $E = 4$ periods per day. Also, we consider the results of the algorithms for tighter and more relaxed number of exam periods, specifically for $\Pi = 20, 24, 28, 36, 40$. We use $|P| = 100$, RefSet size $b = 10$, and the following values for the coefficients of OF: $\alpha = 100$; $\varphi = 1$; $\sigma = 10$; $\beta = 150$; $\gamma = 200$. These coefficient values are chosen in accordance with the university administration's objectives specified in the following order of importance: not allowing room capacity violation; not allowing more than three exams per day for a student; minimizing the number of conflicting exams; limiting the possibility of scheduling three exams per day; limiting the number of consecutive exams for a student.

4.2 Background on GA, SA, and 3PSA

Genetic algorithms mimic the mechanics of natural evolution. That is, they are based on natural populations' reproduction and selection operations to achieve efficient and robust optimization. Through their artificial evolution, successive generations search for beneficial adaptations in order to find a good sub-optimal exam timetable [28]. Each generation consists of a population of chromosomes, where each chromosome represents a timetable. Such candidate solutions are represented in the same way as that of the SS algorithm, which is described in Sect. 3.1. The initial generation consists of randomly-created exam timetables. Each timetable acquires a fitness level that is by the value of the objective function, OF. The Darwinian principles of reproduction and survival of the fittest and the genetic operations of crossover and mutation are used to create a new offspring population from the current population. The reproduction operation involves selecting, in proportion to fitness, a chromosome from the current population of chromosomes, and allowing it to survive by copying it into the new population. Then, two mates are randomly selected from this population, and crossover and mutation are carried out to create two new offspring chromosomes. Crossover involves swapping two randomly located sub-chromosomes (i.e., parts of exam timetables) of the two mating chromosomes. Mutation is applied to randomly-selected exams, where these exams are randomly reassigned to different time slot and room. The offspring population replaces the parent population, and the process is repeated for many generations with the aim of minimizing the objective function of the candidate solutions.

A simulated annealing algorithm is based on an analogy to the physical annealing of a solid in statistical mechanics. The solid corresponds to a single candidate solution. To coerce this solution (i.e., exam timetable), which is represented in the same way as in Sect. 3.1, into a low-energy state the material is heated to a high temperature (high-energy

Table 2 Subject problems

Subject problem	#Exams A	#Rooms R	#Students	#Student enrollments
F06 (Fall 2006)	473	37	3709	13824
S07 (Spring 2007)	472	37	4383	16241
F07 (Fall 2007)	537	37	3911	15392
S08 (Spring 2008)	634	37	3659	15750

state) and then cooled very slowly, allowing it to come to thermal equilibrium at each temperature. The high-energy state is given by a an initially randomly generated exam timetable. At freezing temperatures, the exam timetable is expected to acquire a good sub-optimal value (low-energy state) for its objective function, OF [28]. The adaptation of the exam schedule at each temperature in the cooling schedule is simulated by the Metropolis algorithm. At each temperature, regions in the solution space are searched; an iteration of the Metropolis algorithm starts with proposing a random reassignment (perturbation) to an exam in the candidate timetable and evaluating the resultant change in the objective function. If the change is negative, corresponding to a downhill move in the energy landscape, the reassignment is accepted and the new lower energy configuration becomes the starting point for the next perturbation. If the change is positive, corresponding to an uphill move, the proposed reassignment may be accepted with a temperature-dependent probability. Metropolis iterations are repeated many times at each temperature, until equilibrium. The temperature is lowered in each cooling step by using a chosen schedule. After many cooling steps, a freezing point is reached where perturbations do not improve the solution and are no longer accepted. At this point, the exam timetable is expected to be near-optimal.

The 3-phase simulated annealing technique implements SA in 3 phases [27]. In the first phase, SA aims to assign blocks of exams to the time slots in a way that minimizes S_{SE}. In the second phase, SA reorders the exam blocks for the objective of minimizing S_{CE}, S_{3E}, and S_{4E}. In the last phase, SA is used to assign the exams to rooms with the objective of not exceeding room capacities.

4.3 Results and discussion

Tables 3–6 give the results of the four techniques (SS, SA, GA, 3PSA) and of manual timetabling for the four subject problems for six different exam periods: $20 \leq \Pi \leq 40$ (exam days B: 5–10). The manual results are reported only for typical numbers (used in practice) of 32 or 36 exam periods. Also, Fig. 2 shows a bar chart of the average values of the four evaluation metrics (S_{SE}, S_{CE}, S_{3E}, S_{4E}) given as ratios with respect to the minimum value; the minimum value is normalized to one and the average is taken over the values

Table 3 Results for subject problem F06, $20 \leq \Pi \leq 40$

		20	24	28	32	36	40
S_{SE}	SS	30	6	1	0	0	0
	SA	37	13	0	0	0	0
	GA	30	5	1	0	0	0
	3PSA	35	5	2	0	0	0
	Manual				54		
S_{CE}	SS	1314	961	701	412	351	183
	SA	1476	1027	881	496	304	154
	GA	1339	973	723	436	337	190
	3PSA	1241	1000	786	583	484	364
	Manual				1300		
S_{3E}	SS	131	49	31	7	0	0
	SA	122	68	30	10	2	0
	GA	132	52	29	7	0	0
	3PSA	133	56	21	16	0	0
	Manual				58		
S_{4E}	SS	1	0	0	0	0	0
	SA	1	0	0	0	0	0
	GA	0	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual				3		
#Rooms exceeding capacity	SS	0	0	0	0	0	0
	SA	0	0	0	0	0	0
	GA	0	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual				2		

Table 4 Results for subject problem S07, $20 \leq \Pi \leq 40$

		20	24	28	32	36	40
S_{SE}	SS	25	5	1	0	0	0
	SA	33	11	1	0	0	0
	GA	24	5	0	0	0	0
	3PSA	32	10	2	1	0	0
	Manual				58		
S_{CE}	SS	1524	931	628	540	241	120
	SA	1438	1161	692	526	331	205
	GA	1449	949	624	551	247	132
	3PSA	1347	1097	776	623	439	445
	Manual				1289		
S_{3E}	SS	158	62	29	12	2	0
	SA	146	71	30	12	4	1
	GA	157	64	30	11	2	0
	3PSA	135	74	31	12	5	0
	Manual				60		
S_{4E}	SS	2	0	0	0	0	0
	SA	1	0	0	0	0	0
	GA	1	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual				2		
#Rooms exceeding capacity	SS	0	0	0	0	0	0
	SA	0	0	0	0	0	0
	GA	0	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual				5		

that correspond to the typical Π values (32 in Tables 3–5 and 36 in Table 6). From all these results, we develop the following findings:

(a) SS produces good exam timetables. This finding is based on the low values obtained for the evaluation metrics: S_{SE}, S_{3E} and S_{4E} relative to the total number of students. For the typical values of Π, S_{SE} ranges between 0% and 0.4% and S_{3E} ranges between 0.2% and 0.5% of the total number of students. S_{CE} is reasonable since it ranges between 11% and 15% of the total number of students. Furthermore, in Tables 3–6, S_{4E} is zero for all cases except for $\Pi = 20$. SS also manages to fit students in the available rooms for all test cases. Moreover, SS allows some reduction in the number of exam days while keeping the values of the evaluation metrics reasonable using the same number of rooms. In our examples, SS allows a reduction of at least one day (i.e., 4 exam periods).

(b) SS produces much better solutions than those of extensive manual timetabling. This is true for all values recorded in Tables 3–6, for the typical Π values.

(c) SS gives better (or equal) results than those of 3PSA for all the values of the first four evaluation metrics (S_{SE}, S_{CE}, S_{3E}, S_{4E}) for the typical Π values. The graph of the average values in Fig. 2 confirms this assessment. In terms of these evaluation metrics in Tables 3–6, the overall results of SS are better 48% of the time, equal 35% of the time, and worse 17% of the time.

(d) SS gives better (or equal) results than those of SA for all the values, except one, of the four evaluation metrics that correspond to the typical Π values. In terms of these evaluation metrics in Tables 3–6, the overall results of SS are better 55% of the time, equal 32% of the time, and worse 13% of the time.

(e) SS gives better (or equal) results than those of GA for all the values, except one, of the four evaluation metrics that correspond to the typical Π values. In terms of these evaluation metrics in Tables 3–6, the overall results of SS are better 32% of the time, equal 44% of the time, and worse 24% of the time.

(f) The four techniques: SS, SA, GA, and 3PSA perform the same in terms of meeting room capacities.

Table 5 Results for subject problem F07, $20 \leq \Pi \leq 40$

		20	24	28	32	36	40
S_{SE}	SS	41	14	11	7	6	6
	SA	58	27	10	7	6	6
	GA	40	16	7	7	6	6
	3PSA	45	19	7	7	6	6
	Manual				56		
S_{CE}	SS	1785	1328	965	576	468	245
	SA	1828	1440	1221	626	504	307
	GA	1696	1320	990	580	464	247
	3PSA	1575	1224	1030	701	552	469
	Manual				630		
S_{3E}	SS	171	87	50	20	6	1
	SA	192	102	69	23	8	4
	GA	180	88	47	22	6	0
	3PSA	193	89	46	28	11	4
	Manual				40		
S_{4E}	SS	1	0	0	0	0	0
	SA	2	1	1	0	0	0
	GA	1	0	0	0	0	0
	3PSA	1	0	0	0	0	0
	Manual				2		
#Rooms exceeding capacity	SS	0	0	0	0	0	0
	SA	0	0	0	0	0	0
	GA	0	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual				0		

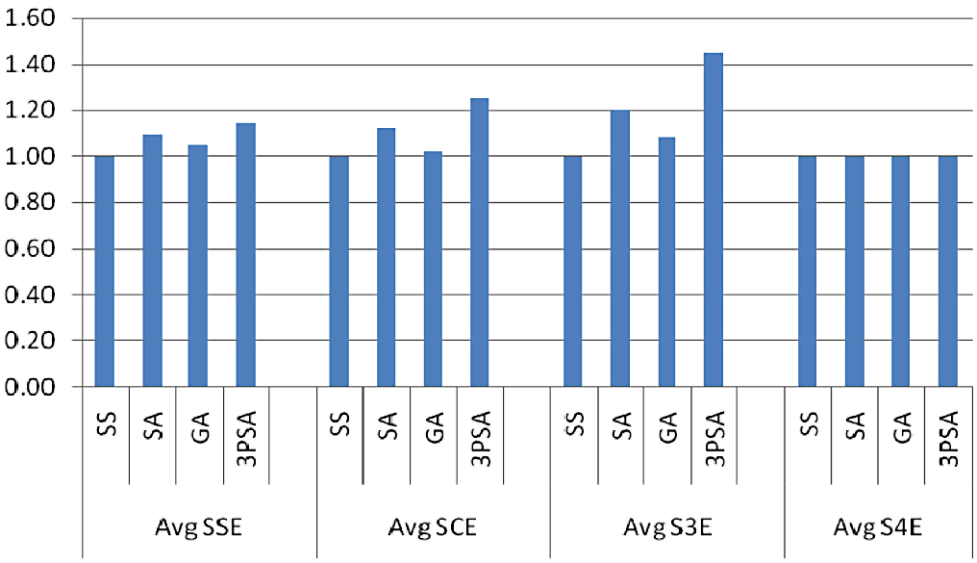

Fig. 2 Normalized average results, for typical Π values

(g) The effect of the coefficient values in OF (1) is salient in the results, where the emphasis is on reducing S_{SE} and S_{4E} and on preventing room capacity violation. From the results that are based on the typical Π values, it is clear that the four meta-heuristic algorithms lead to zero values for S_{4E} and #Rooms due to the high values of their

Table 6 Results for subject problem S08, $20 \leq \Pi \leq 40$

		20	24	28	32	36	40
S_{SE}	SS	67	35	22	18	14	16
	SA	79	40	24	19	16	14
	GA	70	25	21	18	15	15
	3PSA	71	33	23	15	16	16
	Manual					58	
S_{CE}	SS	1854	1426	1001	858	550	297
	SA	2108	1487	1232	874	686	425
	GA	1877	1015	1013	849	561	291
	3PSA	1950	1025	1019	892	697	596
	Manual					720	
S_{3E}	SS	155	72	47	19	10	2
	SA	191	95	50	29	14	4
	GA	158	74	48	20	13	3
	3PSA	193	75	51	24	15	5
	Manual					45	
S_{4E}	SS	0	0	0	0	0	0
	SA	0	0	0	0	0	0
	GA	0	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual					2	
#Rooms exceeding capacity	SS	0	0	0	0	0	0
	SA	0	0	0	0	0	0
	GA	0	0	0	0	0	0
	3PSA	0	0	0	0	0	0
	Manual					0	

coefficients. S_{SE} is zero for two subject problems and non-zero for the other two. In the latter cases, the limited number of days, the density of course/exam interaction (due to students enrolling in a number of courses), and the relatively moderate coefficient value allowed some exams to be scheduled simultaneously. This small number of simultaneous exams has to be resolved by the concerned students with their instructors. But, we have also experimented with different coefficient values, as shown in Table 7. This table shows if higher relative values are assigned to the coefficient of simultaneous exams, the number of such exams drop. However, as expected, the numbers of consecutive exams and 3-exams per day grow. These results demonstrate the adaptability of the SS algorithm to different user requirements.

(h) As expected, the SS and other techniques produce better results as Π increases and worse results as Π decreases.

(i) Table 8 gives the range of execution times of the four algorithms for all the subject problems over all values of Π, in minutes. These times show that the improvement in solution quality produced by SS in comparison with the other meta-heuristics comes at a price, which is longer execution time. But, this is not a critical disadvantage, since exam timetabling software is usually run off-line.

Table 7 SS results for subject problem S08 with different OF's coefficient values

Coefficient values	S_{SE}	S_{CE}	S_{3E}	S_{4E}	#Rooms exceeding
$\alpha = 100; \varphi = 1; \sigma = 10;$ $\beta = 150; \gamma = 200.$	14	550	10	0	0
$\alpha = 200; \varphi = 1; \sigma = 5;$ $\beta = 50; \gamma = 50$	6	733	29	0	0

Table 8 Execution times (minutes)

SS	SA	GA	3PSA
23–52	4–7	15–25	4–5

5 Conclusion

We have proposed a scatter search technique for producing good sub-optimal solutions for the final exam timetabling problem, given specific objectives and constraints. The technique has been evaluated using real university data for four semesters and has been compared with extensive manual timetabling as well as three meta-heuristics, namely, genetic algorithm, simulated annealing, and 3-phase simulated annealing. Our experimental results show that the SS technique produces good exam timetables that satisfy the stated objectives and are certainly better than those produced by extensive manual timetabling. The results also show that SS tends to give better exam timetables than those produced by the other three meta-heuristics.

Acknowledgements This work was partially supported by the Lebanese American University and the Lebanese National Council for Scientific Research. The authors thank the anonymous reviewers whose comments led to improving the presentation of the paper.

References

1. Abdullah S, Ahmadi S, Burke EK, Dror M (2007) Investigation Ahuja-Orlins large neighbourhood search for examination timetabling. OR Spectrum 29(2):351–372
2. Ahmadi S, Barone R, Cheng P, Cowling P, McCollum B (2003) Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. In: Proceedings of multidisciplinary international scheduling: theory and applications (MISTA 2003), Nottingham, pp 155–171
3. Asmuni H, Burke EK, Garibaldi JM, McCollum B (2005) Fuzzy multiple heuristic orderings for examination timetabling. In: Burke EK, Trick M (eds) Proceedings of the 5th international conference on the practice and theory of automated timetabling. Lecture notes in computer science, vol 3616. Springer, Berlin, pp 334–353
4. Azimi ZN (2005) Hybrid heuristics for examination timetabling problem. Appl Math Comput 163(2):705–733
5. Bilgin B, Ozcan E, Korkmaz EE (2007) An experimental study on hyperheuristics and exam timetabling. In: Burke EK, Rudova H (eds) Practice and theory of automated timetabling: selected papers from the 6th international conference. Lecture notes in computer science, vol 3867. Springer, Berlin, pp 394–412
6. Burke EK, Elliman DG, Ford PH, Weare RF (1996) Examination timetabling in British universities: a survey. In: Burke EK, Ross P (eds) Practice and theory of automated timetabling: selected papers from the 1st international conference. Lecture notes in computer science, vol 1153. Springer, Berlin, pp 76–90
7. Burke EK, Newall JP (1999) A multi-stage evolutionary algorithm for the timetable problem. IEEE Trans Evol Comput 3(1):63–74
8. Burke E, Bykov Y, Newall J, Petrovic S (2004) A time-predefined local search approach to exam timetabling problems. IIE Trans Oper Eng 36(6):509–528
9. Burke EK, Kingston JH, de Werra D (2004) Applications to timetabling. In: Gross J, Yellen J (eds) The handbook of graph theory. Chapman Hall/CRC, London, pp 445–474
10. Campos V, Glover F, Laguna M, Marti R (2001) An experimental evaluation of a scatter search for the linear ordering problem. J Glob Optim 21:397–414
11. Carter MW, Laporte G, Lee SY (1996) Examination timetabling: algorithmic strategies and applications. J Oper Res Soc 47(3):373–383
12. Carter M, Johnson DG (2001) Extended clique initialization in examination timetabling. J Oper Res Soc 52(5):538–544
13. Cheong CY, Tan KC, Veeravalli B (2009) A multi-objective evolutionary algorithm for examination timetabling. J Sched 12:121–146
14. Cooper TB, Kingston JH (1996) The complexity of timetable construction problems. In: Burke EK, Ross P (eds) Practice and theory of automated timetabling: selected papers from the 1st international conference. Lecture notes in computer science, vol 1153. Springer, Berlin, pp 283–295
15. Côté P, Wong T, Sabourin R (2005) Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem. In: Burke E, Trick M (eds) Proceedings of the 5th international conference on the practice and theory of automated timetabling. Lecture notes in computer science, vol 3616. Springer, Berlin, pp 294–312
16. Di Gaspero L, Schaerf A (2000) Tabu search techniques for examination timetabling. In: Proceedings of the 3rd international conference on the practice and theory of automated timetabling, Germany, pp 176–179
17. Dowsland KA, Thompson J (2005) Ant colony optimization for the examination scheduling problem. J Oper Res Soc 56:426–438
18. Glover F (1998) A template for scatter search and path relinking. In: Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D (eds) Artificial evolution. Lecture notes in computer science, vol 1363. Springer, Berlin, pp 13–54
19. Glover F, Laguna M, Martí R (2000) Fundamentals of scatter search and path relinking. Control Cybern 29(3):653–684
20. Glover F, Laguna M, Martí R (2002) Scatter search. In: Ghosh A. Tsutsui S (eds) Theory and applications of evolutionary computation: recent trends. Springer, Berlin, pp 519–529
21. Glover F, Laguna M, Martí R (2003) Scatter search and path relinking: advances and applications. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer Academic, Dordrecht, pp 1–36
22. Kendall G, Mohd Hussin N (2005) A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology. In: Burke E, Trick M (eds) Proceedings of the 5th international conference on the practice and theory of automated timetabling. Lecture notes in computer science, vol 3616. Springer, Berlin, pp 270–293
23. Laguna M, Martí R (2003) Scatter search methodology and implementations in C. Kluwer Academic, Dordrecht
24. Laguna M, Armentano VA (2005) Lessons from applying and experimenting with scatter search. In: Sharda R, Voß S, Rego C, Alidaee B (eds) Metaheuristic optimization via memory and evolution tabu search and scatter search. Springer, Berlin, pp 229–246
25. Lotfi V, Cerveny R (1991) A final-exam-scheduling package. J Oper Res Soc 42:205–216
26. Malim MR, Khader AT, Mustafa A (2006) Artificial immune algorithms for university timetabling. In: Burke EK, Rudova H (eds) Proceedings of the 6th international conference on practice and theory of automated timetabling, Brno, Czech Republic, pp 234–245
27. Mansour N, Tarhini A, Ishakian V (2003) Three-phase simulated annealing algorithms for exam scheduling. In: Proceedings of the 2nd ACS/IEEE international conference on computer systems and applications, Tunis
28. Mansour N, Timani M (2007) Stochastic search algorithms for exam scheduling. Int J Comput Intell Res 3(4):353–361

29. Martí R, Laguna M, Campos V (2005) Scatter search vs. genetic algorithms: an experimental evaluation with permutation problems. In: Rego C, Alidaee B (eds) Metaheuristic optimization via adaptive memory and evolution: tabu search and scatter search. Kluwer Academic, Dordrecht, pp 263–282
30. McCollum B (2007) A perspective on bridging the gap between research and practice in university timetabling. In: Burke EK, Rudova H (eds) Proceedings of the 6th international conference on the practice and theory of automated timetabling. Lecture notes in computer science, vol 3867. Springer, Berlin, pp 3–23
31. McCollum B, Schaerf A, Paechter B, McMullan P, Lewis R, Parkes A, Di Gaspero L, Qu R, Burke E (2009) Setting the research agenda in automated timetabling: the second international timetabling competition. INFORMS J Comput (accepted)
32. Merlot LTG, Boland N, Hughes BD, Stuckey PJ (2003) A hybrid algorithm for the examination timetabling problem. In: Burke EK, Causmaecker PD (eds) Proceedings of the 4th international conference on the practice and theory of automated timetabling. Lecture notes in computer science, vol 2740. Springer, Berlin, pp 207–231
33. Muller T (2008) ITC2007: solver description. In: Proceedings of the 7th international conference on the practice and theory of automated timetabling, Montréal
34. Petrovic S, Bykov Y (2003) A multiobjective optimisation technique for exam timetabling based on trajectories. In: Burke EK, Causmaecker PD (eds) Proceedings of the 4th international conference on the practice and theory of automated timetabling. Lecture notes in computer science, vol 2740. Springer, Berlin, pp 179–192
35. Pillay N, Banzhaf W (2009) A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. Eur J Oper Res 197(2):482–491
36. Qu R, Burke EK (2009) Hybridisation within a graph based hyper-heuristic framework for university timetabling problems. J Oper Res Soc 60:1273–1285
37. Qu R, Burke EK, McCollum B, Merlot LTG, Lee SY (2009) A survey of search methodologies and automated system development for examination timetabling. J Sched 12:55–89
38. Reis LP, Oliveira E (2001) A language for specifying complete timetabling problems. In: Burke EK, Erben W (eds) Practice and theory of automated timetabling: selected papers from the 3rd international conference. Lecture notes in computer science, vol 2079. Springer, Berlin, pp 322–341
39. Ross P, Marin-Blazquez JG, Hart E (2004) Hyper-heuristics applied to class and exam timetabling problems. In: Proceedings of the 2004 congress on evolutionary computation, pp 1691–1698
40. Schaerf A (1999) A survey of automated timetabling. Artif Intell Rev 13(2):87–127
41. Terashima-Marin H, Ross P, Valenzuela-Rendon M (1999) Clique-based crossover for solving the timetabling problem with GAs. In: Schoenauer M et al (eds) Proceedings of CEC'99 conference. IEEE Press, Washington, pp 1200–1206
42. Thompson J, Dowsland K (1998) A robust simulated annealing based examination timetabling system. Comput Oper Res 25:637–648
43. Ulker O, Ozcan E, Korkmaz EE (2007) Linear linkage encoding in grouping problems: applications on graph coloring and timetabling. In: Burke EK, Rudova H (eds) Practice and theory automated timetabling: selected papers from the 6th international conference. Lecture notes in computer science, vol 3867. Springer, Berlin, pp 347–363

Nashat Mansour is a Professor of Computer Science at the Lebanese American University. He received B.E. and M.Eng.Sc. degrees in Electrical Engineering from the University of New South Wales, and M.S. in Computer Engineering and Ph.D. in Computer Science from Syracuse University. His research interests include: application of metaheuristics and data mining to realworld problems; protein structure prediction; software testing.

Vatche Isahakian is a Ph.D. student at the Dept. of Computer Science, Boston University. He received B.S. and M.S. degrees in Computer Science from the Lebanese American University. His research interests are in real-time systems and exam timetabling.

Iman Ghalayini is an M.S. student at the Dept. of Computer Science and Mathematics, Lebanese American University. She received B.S. in Computer Science from the Lebanese American University. Her research interests are in exam timetabling and protein structure prediction.