

Stochastic Search Algorithms for Exam Scheduling

Nashat Mansour and Mazen Timany

*Computer Science Program, Lebanese American University,
Mme Curie st., Beirut, Lebanon, 1102-2801
E-mail: nmansour@lau.edu.lb*

Abstract: Scheduling final exams for large numbers of courses and students in universities is an intractable problem. Where scheduling is done manually, conflicts and unfairness are inevitable. Conflicts occur when 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 schedule should aim to minimize conflicts and the two unfairness factors based on user-assigned weights to these three factors and subject to some constraints such as classrooms' number and capacities. In this work, we use a modified weighted-graph coloring problem formulation and adapt two stochastic search algorithms for solving the problem. The two algorithms are a simulated annealing algorithm (SA) and a genetic algorithm (GA). We also propose an improvement to a 'good' clustering-based heuristic procedure, known as FESP, by using simulated annealing procedures. The improved heuristic is referred to as FESP-SA. Then, we empirically compare the three proposed algorithms and FESP using realistic data. Our experimental results show that SA and GA produce good exam schedules that are better than those of FESP heuristic procedure. Also, SA and GA allow a reduction in the number of exam days without much aggravating conflicts and unfairness. However, SA is more favorable since it is faster than GA.

Keywords: Clustering heuristics, exam scheduling, genetic algorithms, simulated annealing, timetabling.

1. Introduction

In universities where scheduling of final exams for large numbers of courses and students is done manually by the Registrar's Office, many students complain about conflicts or unfairness in their exams schedule. 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 schedule would aim to minimize conflicts and the two unfairness factors based on weights that are user-assigned to these three factors. Such a schedule may also be subject to constraints,

such as predefined number of days and limited-capacity classrooms.

Several approaches and techniques for solving the exam scheduling problem have been developed. Graph coloring has been the basis of a few solutions (Balakrishnan, 1991; Carter, Laporte and Chinneck, 1994; Wood, 1968; Mehta, 1981). Integer programming has also been used (Arani, Karwan and Lotfi, 1988; Descroches, Laporte, and Rousseau, 1978). Clustering and extended clique algorithms have been reported (Carter and Johnson, 2001; Leong and Yeong, 1987; Lotfi and Cerveny, 1991). Several researchers have used versions of genetic and simulated annealing algorithms (Erben, 2000; Ergul, 1996; Thompson and Dowland, 1996; Tarhini and Mansour, 1998; Corne, Fang and Mellish, 1993). Tabu search has been suggested in a few papers (Di Gaspero and Schaerf, 2000; Kendall and Mohd Hussin, 2005; White and Xie, 2000). Ant colony optimization has been adapted by Thompson and Dowland (2005). Multi-criteria based heuristics have been proposed by Burke, Bykov, and Petrovic (2000). Local search and hybrid algorithms have been presented (Burke et al., 2004; Merlot et al., 2003). Deris, Omatu, and Ohta (2000) have modeled exam scheduling as a constraint satisfaction problem. Further, some automated software tools have been developed for exam scheduling (Beynon, Ward and Maad, 2000; McCollum and Newall, 2000).

Previous work has been based on a variety of problem models and solution procedures for exam scheduling. Different approaches have dealt with different objectives and constraints. In particular, previous genetic and simulated annealing algorithms have been based on different design choices to adapt the basic paradigms (Goldberg, 1989; Kirkpatrick, Gelatt, and Vecchi, 1983) to the exam scheduling problem. Further, the solutions of the reported procedures are not usually compared with others for evaluation purposes.

In this work, we first formulate the exam scheduling problem as a modified weighted-graph coloring problem based on specific objectives and constraints. These objectives and constraints are integrated into the objective function. Then, we propose a simulated annealing algorithm (SA) and a hybrid genetic algorithm (GA) for solving the

problem. Both algorithms adapt basic paradigms by using simple design choices. We compare SA and GA to a good clustering-based heuristic solution procedure, FESP (Lotfi and Cerveny, 1991) using realistic data. FESP handles the different scheduling requirements in four phases: Phase 1 to handle exam conflicts, Phases 2 and 3 to handle unfairness, and Phase 4 to assign classrooms to scheduled exams. The experimental results clearly show that SA and GA provide better solutions than FESP. This finding has motivated us to replace heuristics of Phases 3 and 4 of FESP with simulated annealing algorithms; the resulting algorithm is referred to as FESP-SA. The experimental results also demonstrate the advantages of FESP-SA over FESP. Further, these results demonstrate that SA and GA allow a reduction in the length of the exam period at minimal cost.

The rest of the paper is organized as follows. Section 2 presents the problem formulation and the objective function. Sections 3, 4, and 5 describe SA, GA, and FESP-SA, respectively. Section 6 presents the experimental results. Section 7, contains our conclusions.

2. Exam Scheduling Problem and Objective Function

Given that A exams are to be taken by students over B days, where E exam periods can be done per day, the exam scheduling problem consists of assigning A exams to Π ($=B * E$) exam periods, within specified classrooms. 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 having two or more exams on the same day.

In this work, we assume the following conditions and constraints:

i) The user should be provided with the flexibility of assigning weights to the three conflict and unfairness factors.

ii) A limited number of exam periods, Π , is predefined.

iii) A limited predefined number of classrooms, R , are available for exams (first feasibility condition).

iv) Room capacities ($\Psi_1, \Psi_2, \dots, \Psi_R$) are taken into consideration in assigning exams to rooms (second feasibility condition). Also, more than one exam can be assigned to the same room at the same time if they fit.

v) The last period of one day is considered to be consecutive to the first period of the next day.

Scheduling problems can be represented by graphs (Wood, 1968). Let $G(V_G, E_G)$ be a graph in which: vertex $v_i \in V_G$ represents an exam to be scheduled and $|V_G| = A$; vertex weight w_i represents the number of students taking exam v_i ; edge $e \in E_G$ joining two vertices v_i and v_j represents the existence of students taking both exams v_i and v_j ; weight of edge e , w_{ij} , represents the number of students taking both exams v_i and v_j .

The exam scheduling problem can be expressed as a

modified weighted-graph coloring problem, where we color the vertices of a graph using a specified maximum number of colors (exam periods), Π , such that an objective function (OF1 in Equation 1 below) is minimized and the constraints (listed above) are satisfied. A solution to the exam-scheduling problem is henceforth denoted as the configuration C . Note that each color corresponds to an exam period and all vertices having the same color represent the exams that are assigned to the same period. This problem model is a modification to classical graph coloring, since its objective is not to minimize the number of colors. Instead, we have a predefined maximum number of colors Π . But, both models, classical and modified, aim to prevent adjacent vertices from having the same color.

Let $c(v)$ be the color of vertex v , and $\xi = \{c_1, c_2, \dots, c_\pi\}$ be the set of ordered, available colors; that is, $|\xi| = \Pi =$ maximum number of available colors, and $(c_i - c_{i-1}) = 1$ for $i=2,3,\dots, \Pi$. The objective function, OF1, is given in terms of the following factors:

(i) S_{SE} , the total number of students having conflicting simultaneous exams. That is, $S_{SE} = \sum w_{ij}$ for all i and j such that $c(i) = c(j)$

(ii) S_{CE} , the total number of students having consecutive exams. That is, $S_{CE} = \sum w_{ij}$ for all i and j such that $|c(i) - c(j)| = 1$.

(iii) S_{ME} , the total number of students having two or more exams per day. That is, $S_{ME} = \sum w_{ij}$ for all i and j such that $c(i)$ and $c(j)$ refer to exam periods on the same day

(i.e. $c(i) \text{ div } E = c(j) \text{ div } E$).

(iv) $\rho_{ik} = 1$ if the capacity of room i is exceeded in period k , i.e. if room i was assigned a larger number of students than Ψ_i (capacity of room i); otherwise, it is equal to zero.

Specifically,

$$OF1 = \alpha * S_{SE} + \varphi * S_{CE} + \sigma * S_{ME} + \gamma *$$

$$\sum_{1 \leq k \leq \Pi} \sum_{1 \leq i \leq R} \rho_{ik} \tag{1}$$

where α, φ, σ and γ are user-defined weights for simultaneous exams, consecutive exams, multiple exams, and room capacity violations, respectively. The inner summation in $\sum_{1 \leq k \leq \Pi} \sum_{1 \leq i \leq R} \rho_{ik}$ gives the total number of

rooms violated in a period, whereas the outer summation gives the total number of rooms violated in all periods. The useful symbols used are summarized in Table 1.

Note that different values can be assigned to the weights in OF1. These weights are important for producing exam schedules. They might be contradictory; that is, by increasing one of these weights, say α , the solution will improve in minimizing one factor (S_{SE}) while it might increase the other factors. These weights will allow flexibility in using our proposed solution algorithms to suit the user's particular choices or requirements for different instances of the problem.

Table 1 : Summary of useful symbols

Symbol	Meaning
1) A	The number of exams, taken by students in a semester, that are to be scheduled
2) B	The number of exam days
3) E	The number of exam periods per day
Π	The maximum number of available exam periods (maximum number of colors)
R	The maximum number of available classrooms
ξ	The set of available colors (available exam periods); $ \xi = \Pi$
c_j	An available color in ξ (i.e., exam period)
$c(i)$	The period to which exam i is assigned (The color of vertex V_i)
II. C	The system configuration; i.e., exams schedule
S_{SE}	The number of students having conflicting simultaneous exams
S_{CE}	The number of students having consecutive exams
S_{ME}	The number of students having two or more exams per day
ρ_{ik}	Equals 1 if the capacity of room i is exceeded in period k
Ψ_i	Capacity of room i ; the number of seats in room i
α	A weighting factor related to the importance of S_{SE} in OF1
ϕ	A weighting factor related to the importance of S_{CE} in OF1
σ	A weighting factor related to the importance of S_{ME} in OF1
γ	A weighting factor related to the importance of ρ_{ik} in OF1
w_i	The number of students taking exam i
w_{ij}	The number of students taking both exams i and j

3. Simulated Annealing Algorithm

Simulated annealing is based on ideas from physics and is analogous to the physical annealing of a solid (Kirkpatrick, Gelatt and Vecchi, 1983). To coerce some material into a low-energy state, it is heated and then cooled very slowly, allowing it to come to thermal equilibrium at each temperature. At each temperature in the cooling schedule, the Metropolis algorithm simulates the behavior of the system.

The simulated annealing algorithm (SA) simulates the natural phenomenon by a search (perturbations) process in the solution space (energy landscape) optimizing some cost function (energy). It starts with some initial solution at a high (artificial) temperature and then reduces the

temperature gradually to a freezing point. In the following subsections, we describe how simulated annealing is adapted for solving the exam scheduling problem; an outline of the SA algorithm is given in Figure 1.

```

Initial configuration = Random color assignment;
Determine initial temperature T(0);
Determine freezing temperature Tf ;
while (T(i) > Tf and not converged) do
    repeat  $\Pi * A$  times
        Generate_function();
        save_best_sofar();
        T(i) =  $\theta * T(i)$ ;
    endwhile
procedure Generate_function()
perturb();
if ( $\Delta OF1 \leq 0$ ) then
    update() /* accept */
else
    if (random() <  $e^{-\Delta OF1 / T(i)}$ ) then
        update() /* accept */
    else
        reject_purturbation();
    
```

Figure 1 : Outline of the SA algorithm for the exam-scheduling problem.

3.1 Solution representation and energy function

The system to be coerced into a low-energy state in the exam-scheduling problem is represented by the configuration C , which is implemented as an array of records and corresponds to a candidate exam schedule. The size of the array is equal to A . Each record includes: a value $c(i)$, which represents the period to which an exam i is assigned, and a classroom. Recall that we have A exams and $c(i)$ takes a color-value (period) assigned to a graph vertex (exam) i . The color-value of $c(i)$ is some c_j , where $c_j \in \xi$.

The system energy is given by OF1 (Equation 1). In the annealing process, C goes through many changes until it reaches an optimal or sub-optimal configuration at freezing temperature. However, the initial configuration is randomly constructed. That is, the colors, c_k , assigned to $c(i)$, $i=1, 2, \dots, A$, in C are randomly selected from the set ξ and a room is randomly selected from the set of available classrooms.

3.2 The Metropolis step and feasibility

An iteration of the Metropolis step, `Generate_function()`, consists of a perturbation operation, an accept/reject criterion, and a thermal equilibrium criterion. Perturbation to the configuration C is done by randomly (without replacement) selecting a record element with color c_i and assigning it another color c_j (j in the range 1 to Π) and randomly reassigning the classroom.

The acceptance criterion checks the change in OF1 due

to the perturbation. If the change decreases the objective function, the perturbation is accepted and C is updated. However, if the perturbation causes the objective function to increase, it is accepted only with a probability $e^{-\Delta OF1 / T(i)}$. The main advantage of this Monte Carlo algorithm is that the controlled uphill moves can prevent the system from being prematurely trapped in a bad local minimum-energy state. Note that for lower temperature values $T(i)$, the probability of accepting uphill moves becomes smaller; at very low (near-freezing) temperatures, uphill moves are no longer accepted. The perturbation-acceptance step is repeated $IT * A$ times at every temperature after which thermal equilibrium is considered to be reached.

Perturbations can make C infeasible if they violate the predetermined room capacities. But, the formulation of OF1 accounts for this infeasibility problem. The last term in OF1 can be assigned a large weight, γ , so that infeasibility is severely penalized. Thus, infeasible exam schedules will be prevented at low temperatures.

3.3 Cooling schedule

The initial temperature $T(0)$ is the temperature that yields a high initial acceptance probability of 0.93 for uphill moves. The freezing point is the temperature at which such a probability is very small (2^{-30}), making uphill moves impossible and allowing only downhill moves. The cooling schedule used in this work is simple: $T(i+1) = \theta * T(i)$, with $\theta = 0.95$.

As the annealing algorithm searches the solution space, the best-so-far solution (with the smallest OF1) found is always saved. This guarantees that the output of the algorithm is the best solution it finds regardless of the temperature it terminates at. Convergence is then detected when the algorithm does not improve on the best-so-far solution for a number of temperatures, say 20, in the colder part of the annealing schedule.

4. Genetic Algorithm

Genetic Algorithms are based on the idea of natural evolution (Goldberg, 1989). They simulate natural populations' reproduction and selection operations to achieve optimal results. Through artificial evolution, successive generations search for fitter adaptations in order to solve a problem. Each generation consists of a population of chromosomes, also called individuals, and each chromosome represents a possible solution. The Darwinian principle of reproduction and survival of the fittest and the genetic operations of recombination (crossover) and mutation are used to create a new offspring population from the current population. The process is repeated for many generations with the aim of maximizing the fitness of the individuals. In the following subsections, we describe how a genetic algorithm (GA) is adapted for solving the exam scheduling problem. An outline of this GA is given in Figure 2. This GA is hybridized with a local hill-climbing procedure in order to improve the final solution.

```

Random generation of initial population, size POP;
Evaluate fitness of individuals;
repeat
Rank individuals and allocate reproduction trials;
for i = 1 to POP step 2
    Randomly select two parents from list of
        reproduction trials;
    Apply crossover and mutation;
endfor
Apply hill-climbing procedure to offspring;
Evaluate fitness of offspring;
    save_best_so_far();
until convergence;

```

Figure 2 : Outline of the GA algorithm for the exam scheduling problem.

4.1 Chromosomal representation and fitness

GA's population is an array of POP individuals. An individual in the population is encoded as an A-element array of records that corresponds to a candidate exam schedule. Each record includes: a color $c(i)$, which represents the period to which an exam i (graph vertex) is assigned, and a classroom. The initial population of individuals is randomly generated. That is, the colors, c_k , assigned to $c(i)$, $i=1, 2, \dots, A$, in each individual are randomly selected from the set ξ and a room is randomly selected from the set of available classrooms.

We use $1/OF1$ as the fitness of an individual that is required to be maximized.

4.2 Reproduction scheme and convergence

The whole population is considered a single reproduction unit within which random selection is performed. Our reproduction scheme involves ranking, followed by random selection of mates from the list of reproduction trials (or copies) assigned to the ranked individuals. In the ranking scheme, the individuals are sorted by fitness values. After sorting, each individual is assigned a rank based on a scale of equidistant values for the population. The ranks assigned to fittest and least-fit individuals are 1.2 and 0.8, respectively. Individuals with ranks greater than 1 are first assigned single copies. Then, the fractional part of their ranks and the ranks of the lower half of individuals are treated as probabilities for random assignment of copies.

Elitism is used to exploit good building blocks and to ensure that good candidate solutions are preserved. This is done by replacing the least-fit individual with the best-so-far individual if the latter is better than the current-fittest. Convergence is detected when the best-so-far candidate solution does not change its OF1 value for 20 generations.

4.3 Genetic operators

The genetic operators employed in GA are 2-point crossover and mutation at the rates 0.75 and 0.01, respectively. The

application of the operators starts with randomly selecting pairs of individuals from the mating pool. Each pair of these chromosomes undergoes crossover, where positions k and l along the chromosome are selected at random between 1 and N , and all genes between k and l are swapped to create two new chromosomes. Then, mutation is applied to randomly selected genes: the value of $c(i)$, is randomly changed from c_j to c_k ($\in \zeta$), and the room is randomly changed. Eventually, the new offspring population replaces the parent population.

We note that the genetic operators, especially crossover, may produce chromosomes that represent infeasible solutions. Such solutions require fitting a larger number of students in a room than the rooms' capacity. We tolerate infeasible chromosomes in order to allow searching infeasible regions in the fitness landscape since such regions might contain useful paths to high-fitness feasible regions. But, the formulation of OF1 includes a penalty term (the fourth term) for infeasibility, which reduces the likelihood of survival of infeasible chromosomes.

4.4 Hybridization

For many applications, hybridizing the GA is useful for improving the quality of the final solution and to reduce evolution time (Davies, 1991). Our exam scheduling GA is hybridized with a local hill-climbing procedure which is applied to all offspring in each generation. In this procedure, we consider every element in a chromosome and randomly reassign it. That is, we reschedule an exam to another period and another room. This change is accepted only if it increases the fitness of the chromosomes and that it does not violate the room capacity constraint. This procedure can be carried out efficiently since the computation of the change in fitness is done based on local information related to the reassigned exam only.

5. FESP-SA Algorithm

5.1 Background

Lotfi and Cerveny (1991) proposed a heuristic algorithm for exam scheduling and called it Final Exam Scheduling Package FESP. FESP consists of four phases, where each phase solves a part of the problem until a final solution is obtained. An outline of FESP is given in Figure 3 .

Phase I: Assign exams to I blocks aiming to minimize simultaneous exams.
 Phase II: Assign the I blocks to exam days aiming to minimize multiple exams per day.
 Phase III_1: Arrange blocks within the same day to minimize consecutive exams.
 Phase III_2: Arrange exam days to minimize consecutive exams resulting from exams that belong to the last period in one day and the first period in the next day.
 Phase IV: Assign exams to rooms.

In Phase I, first, any I exams are assigned to I blocks. Then, an exam is selected and assigned to one of the I blocks where it does not cause any simultaneous exam conflicts. If such conflicts exist in all I blocks, the exam is assigned to the block with the least number of simultaneous conflicts. This step is repeated until all exams are assigned. In Phase II, the problem of assigning I blocks to B exam days is formulated as a quadratic assignment problem as is solved by an iterative improvement heuristic. In Phase III, the E blocks within each day are arranged to minimize the number of consecutive exams. This is a trivial traveling salesperson problem (TSP), since E is typically close to 4. Then, the B exam days are arranged to minimize consecutive exams resulting from the last period of one day and the first period of the next day. This is also formulated as a TSP and is solved using a heuristic. In Phase IV, exams are assigned to rooms using a greedy algorithm.

5.2 FESP-SA

FESP consists of four-phase heuristics for solving four optimization sub-problems, one sub-problem in each phase. The successful application of SA to exam scheduling made us consider applying it to some of the optimization sub-problems. In this subsection, we describe how FESP is modified by replacing its heuristics for solving phases III and IV with SA. We refer to the resulting algorithm as FESP-SA.

In phase III, FESP attempts minimize the number of consecutive conflicts in two steps, first within days and then among consecutive days. We observe that one step might impede the other. Thus, we apply SA to the whole TSP sub-problem and use OF2 = S_{CE} as the energy function to be minimized. The configuration to be annealed is based on the output of phase II, which is a graph with I vertices. Each vertex represents a block (determined in phase I) and an edge weight represents the number of students participating in exams that lie in both adjacent blocks. Thus, the configuration considered by SA is an array of I elements, where each element refers to the period to which the whole block is assigned. Perturbations are done by swapping blocks between periods, provided that they respect the results of phase II that assigned groups of E blocks to exam days. This implies that we allow two types of perturbations with equal probabilities: fine-grain perturbation, in which blocks are swapped within the same day, and course-grain perturbation, in which two groups of E blocks each are swapped.

In phase IV, we also replace the greedy algorithm with an SA algorithm. The input for this phase is an exam schedule with A elements, each with a specified exam period and needs to be assigned to a classroom. We use a configuration similar to that described in Subsection 3.1. A perturbation consists of reassigning a room in an element (for an exam). At each temperature, thermal equilibrium is reached after $(A * R / I)$ perturbations. The energy function to

Figure 3 : Outline of the FESP Algorithm.

be minimized is $OF3 = \sum_{1 \leq k \leq \Pi} \sum_{1 \leq i \leq R} \rho_{ik}$. The SA algorithm

is applied (Π times) to the exams in each of the Π blocks that have been assigned to the exam periods.

6. Experimental Results

6.1 Situation

In this section, we present the results of SA, GA, and FESP-SA and compare them with those of FESP and manual scheduling. These algorithms are applied to real data of university exams obtained from five semesters. These data provide five subject problem instances, SP1-SP5, illustrated in Table 2. This table shows the number of exams (A), number of available classrooms (R), the number of students, and the total number of student enrolment for all exams-courses (a student usually enrolls in more than one course). We also experiment with two sets of values for the coefficients of the four terms of OF1 in equation 1. These values lead to two versions of SA and GA, V1 and V2. Table 3 gives the values used for the two versions.

Table 2 : Subject Problems

Subject Problem	A	R	# of Students	# of Student Enrolments
SP1	336	21	2456	9550
SP2	357	21	2489	9735
SP3	359	21	2512	10836
SP4	426	21	3063	12275
SP5	477	21	3115	12406

Table 3 : Different versions using different weights in OF1

Versions	α	φ	σ	γ
Version 1	100	5	0.2	200
Version 2	100	1	1	200

The exam schedules produced by the algorithms are evaluated in terms of the five metrics: number of students having simultaneous exams (S_{SE}), number of students having consecutive exams (S_{CE}), number of students having two or more exams per day (S_{ME}), total number of rooms assigned with more students than their capacities over all exam periods, and execution time. We first apply the algorithms to the five subjects problems with a typical number of exam periods (at the university where data is obtained), namely $\Pi = 32$ periods for $B = 8$ days and $E = 4$ periods per day. Then, we consider the results of the algorithms for tighter and more relaxed number of exam periods, specifically for $\Pi = 20, 24, 28, 36, 40$. The results recorded below for SA and GA are averages of 10 runs (rounded to the nearest integer).

6.2 Results and discussion

Tables 4–8 give the results of the four algorithms (SA, GA, FESP-SA, FESP) and of manual scheduling for the five subject problems SP1–SP5. The values produced by SA and

GA are associated with the two versions illustrated in Table 3. These results apply for a typical number of 32 exam periods (over 8 days). Tables 9–13 give the results of the four algorithms for different exam periods $20 \leq \Pi \leq 40$ (and different exam days). In these tables, we only show the results of version 2 of SA and GA. Table 14 gives the range of execution times of the implementation of the four algorithms over the five problems. From these results, we infer the following findings:

- SA and GA produce ‘good’ exam schedules. This finding is based on the low values obtained for S_{SE} and S_{ME} relative to the total number of students. SA and GA also manage to fit students in the available rooms except for very few cases where $\Pi = 20$. Further, SA and GA easily allow a reduction in the number of exam days while keeping the values of the first three metrics reasonable using the same number of rooms. In our examples, SA and GA allow a reduction of at least one day (i.e., from 32 to 28 periods). Furthermore, SA and GA can provide alternative solutions to a user simply by rerunning them with new seed values for the random number generator included in them.
- SA and GA produce better solutions than FESP, FESP-SA, and manual scheduling. This is true for all values recorded in Tables 4–8 and for almost all values recorded in Tables 9–13. The advantages of SA and GA are clearer for smaller number of exam periods (in Tables 9–13). A shortcoming of FESP stands out in Table 13, where it fails to produce feasible solutions that fit the available classrooms for all six values of exam periods.
- Table 14 gives an impression that SA, FESP, and FESP-SA are comparable in terms of execution times. GA is much slower. But, this performance of GA may be acceptable since exam schedules are usually produced off-line. However, since SA and GA are comparable for the first four metrics, SA is favored since it is faster as far as the performance metric is concerned.
- FESP-SA produces lower values for S_{CE} than those of FESP for all five problems and all six values of the exam periods. It also produces better room assignments as shown in Tables 9–13. Therefore, FESP-SA is certainly an improvement to FESP.
- As expected, all four algorithms produce better results as Π increases and worse results as Π decreases.
- The use of two sets of values of the coefficients of OF1, shown in Table 3, show the flexibility allowed in SA and GA for the user to place different emphasis on the weights of the three metrics: S_{SE} , S_{CE} , and S_{ME} . For example, we note that version 1 involves a higher φ to σ ratio than version 2. This is why SA-V1 and GA-V1 produce lower S_{CE} and higher S_{ME} values than those of SA-

V2 and GA-V2, respectively (refer to Tables 4–8).

Table 4 : Results for Subject Problem SP1, $II = 32$

	S_{SE}	S_{CE}	S_{ME}	# Rooms exceeding capacity
SA-V1	0	62	268	0
GA-V1	0	65	247	0
SA-V2	0	81	203	0
GA-V2	0	86	203	0
FESP-SA	0	133	286	0
FESP	0	221	286	0
Manual	8	267	329	0

Table 5 : Results for Subject Problem SP2, $II = 32$

	S_{SE}	S_{CE}	S_{ME}	# Rooms exceeding capacity
SA-V1	0	34	206	0
GA-V1	0	23	207	0
SA-V2	0	41	136	0
GA-V2	0	67	150	0
FESP-SA	0	119	264	0
FESP	0	183	264	0
Manual	5	251	310	0

Table 6 : Results for Subject Problem SP3, $II = 32$

	S_{SE}	S_{CE}	S_{ME}	# Rooms exceeding capacity
SA-V1	0	16	80	0
GA-V1	0	17	95	0
SA-V2	0	17	52	0
GA-V2	0	11	50	0
FESP-SA	0	32	98	0
FESP	0	67	98	0
Manual	4	115	146	0

Table 7 : Results for Subject Problem SP4, $II = 32$

	S_{SE}	S_{CE}	S_{ME}	# Rooms exceeding capacity
SA-V1	0	36	243	0
GA-V1	0	40	264	0
SA-V2	0	68	179	0
GA-V2	0	73	173	0
FESP-SA	0	179	368	0
FESP	0	242	368	0
Manual	12	283	396	0

Table 8 : Results for Subject Problem SP5, $II = 32$

	S_{SE}	S_{CE}	S_{ME}	# Rooms exceeding capacity
SA-V1	0	14	100	0
GA-V1	0	17	102	0
SA-V2	0	19	72	0
GA-V2	0	18	61	0
FESP-SA	0	62	157	0
FESP	0	100	157	2
Manual	9	154	231	0

Table 9 : Results for subject problem SP1, $20 \leq II \leq 40$

		20	24	28	32	36	40
S_{SE}	SA-V2	3	2	0	0	0	0
	GA-V2	4	1	0	0	0	0
	FESP-SA	41	5	2	0	0	0
	FESP	41	5	2	0	0	0
S_{CE}	SA V2	336	194	150	81	52	30
	GA V2	308	220	143	86	58	24
	FESP-SA	272	215	159	133	81	62
	FESP	368	300	241	221	138	133
S_{ME}	SA-V2	572	384	300	203	140	116
	GA-V2	537	420	265	203	162	106
	FESP-SA	577	466	374	286	221	181
	FESP	577	466	374	286	221	181
#Rooms exceeding capacity	SA-V2	1	0	0	0	0	0
	GA-V2	1	0	0	0	0	0
	FESP-SA	2	0	0	0	0	0
	FESP	3	2	0	0	0	0

Table 10 : Results for subject problem SP2, $20 \leq II \leq 40$

		20	24	28	32	36	40
S_{SE}	SA-V2	2	0	0	0	0	0
	GA-V2	2	0	0	0	0	0
	FESP-SA	22	1	1	0	0	0
	FESP	22	1	1	0	0	0
S_{CE}	SA-V2	252	200	77	41	19	15
	GA-V2	226	143	92	67	26	9
	FESP-SA	246	181	156	119	83	45
	FESP	332	264	212	183	165	97
S_{ME}	SA-V2	478	310	203	136	80	65
	GA-V2	412	293	190	150	90	62
	FESP-SA	526	422	344	264	216	168
	FESP	526	422	344	264	216	168
#Rooms exceeding capacity	SA-V2	0	0	0	0	0	0
	GA-V2	0	0	0	0	0	0
	FESP-SA	2	1	0	0	0	0
	FESP	4	2	1	0	0	0

Table 11 : Results for subject problem SP3, $20 \leq II \leq 40$

		20	24	28	32	36	40
S_{SE}	SA-V2	1	1	0	0	0	0
	GA-V2	1	0	0	0	0	0
	FESP-SA	5	1	0	0	0	0
	FESP	5	1	0	0	0	0
S_{CE}	SA-V2	104	52	26	17	7	4
	GA-V2	108	66	41	11	10	8
	FESP-SA	94	71	45	32	28	22
	FESP	136	106	77	67	54	49
S_{ME}	SA V2	187	112	74	52	36	27
	GA V2	190	134	94	50	44	25
	FESP-SA	216	165	117	98	71	69
	FESP	216	165	117	98	71	69
#Rooms exceeding capacity	SA-V2	0	0	0	0	0	0
	GA-V2	0	0	0	0	0	0
	FESP-SA	1	0	0	0	0	0
	FESP	2	0	0	0	0	0

Table 12 : Results for subject problem SP4, $20 \leq \Pi \leq 40$

		20	24	28	32	36	40
S _{SE}	SA-V2	2	1	0	0	0	0
	GA-V2	3	0	0	0	0	0
	FESP-SA	32	8	3	0	0	0
	FESP	32	8	3	0	0	0
S _{CE}	SA-V2	370	223	111	68	43	11
	GA-V2	293	186	112	73	38	25
	FESP-SA	313	251	182	179	98	72
	FESP	431	304	250	242	162	152
S _{ME}	SA-V2	606	386	269	179	121	79
	GA-V2	549	382	265	173	114	75
	FESP-SA	688	541	379	368	277	226
	FESP	688	541	379	368	277	226
#Rooms exceeding capacity	SA-V2	3	0	0	0	0	0
	GA-V2	0	0	0	0	0	0
	FESP-SA	3	1	0	0	0	0
	FESP	7	2	1	0	0	0

Table 13 : Results for subject problem SP5, $20 \leq \Pi \leq 40$

		20	24	28	32	36	40
S _{SE}	SA-V2	2	1	0	0	0	0
	GA-V2	2	0	0	0	0	0
	FESP-SA	9	1	0	0	0	0
	FESP	9	1	0	0	0	0
S _{CE}	SA-V2	130	68	35	19	14	6
	GA-V2	117	77	44	18	20	8
	FESP-SA	137	99	80	62	36	28
	FESP	195	130	118	100	64	51
S _{ME}	SA-V2	222	144	87	72	49	29
	GA-V2	255	158	117	61	50	36
	FESP-SA	316	227	175	157	116	89
	FESP	316	227	175	157	116	89
#Rooms exceeding capacity	SA-V2	0	0	0	0	0	0
	GA-V2	0	0	0	0	0	0
	FESP-SA	3	1	2	0	1	0
	FESP	5	2	3	2	2	1

Table 14 : Range of execution times, in minutes

	Execution time
SA	2.5 - 4.2
GA	48 - 89
FESP-SA	3.3 - 4.2
FESP	2.4 - 4.0

7. Conclusion

We have presented three algorithms for solving the exam scheduling problem: simulated annealing (SA), genetic algorithm (GA), and FESP-SA which is an SA based improvement to a clustering heuristic called FESP. We have also experimentally evaluated these algorithms using realistic university data for five semesters. The experimental results demonstrate that SA and GA produce good exam schedules and allow a reduction in the number of exam days/periods and yet the number of students with unfair exam schedules is still reasonable. The results also show that SA and GA produce comparable solutions. But, SA is

more favorable since it is faster than GA for exam scheduling.

Acknowledgement

We thank A. Tarhini, M. Awad, Z. Mikati, and Vatche Ishakian for helping with the coding.

References

- [1] Arani, T. Karwan, M. and Lotfi, V. (1988). A Lagrangian relaxation approach to solve the second phase of the exam scheduling problem, *Euro. J. Operational Research*, 34 (3), pp. 372-383.
- [2] Balakrishnan, N. (1991). Examination Scheduling: a Computerized Application *Omega*, 19 (1), pp. 37-41.
- [3] Beynon, M. Ward, A. and Maad, S (2000). The Temposcope: A Computer Instrument for the Idealist Timetabler. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Germany, pp. 153-175.
- [4] Burke, E.K. Bykov, Y. and Petrovic, S. (2000). Multicriteria Approach To Timetabling Problems. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Germany, pp.180.
- [5] Burke, E., Bykov, Y., Newall, J. and Petrovic, S. (2004). A Time-Predefined Local Search Approach to Exam Timetabling Problems. *IIE Transactions on Operations Engineering*, 36 (6), pp. 509-528.
- [6] Carter, M. and Johnson, D. G. (2001). Extended Clique Initialization in Examination Timetabling. *Journal of the Operational Research Society*, 52 (5), pp. 538-544.
- [7] Carter, M. Laporte, G. and Chinneck, J. (1994). A general examination scheduling system, *Interfaces*, 24 (3), pp. 109-120.
- [8] Corne, D. Fang, H. L. and Mellish, C. (1993). Solving the Modular Exam Scheduling Problem with Genetic Algorithms. *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh, pp. 370-373.
- [9] Davies, L (Ed.), (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.
- [10] Deris, S. Omatu, S. and Ohta H. (2000). Timetable Planning Using the Constraint Based Reasoning. *Computer and Operations Research*, 27 (9), pp. 819-839.
- [11] Descroches, S. Laporte, G. and Rousseau, J. (1978). HOREX: a computer program for the construction of examination schedules, *INFOR*, 16 (3), pp. 294-298.
- [12] Di Gaspero, L. and Schaerf, A. (2000). Tabu

- Search Techniques for Examination Timetabling. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Germany, pp.176-179.
- [13] Erben, W. (2000). A Grouping Genetic Algorithm for Graph Coloring and Exam Timetabling. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Germany, pp.397-421.
- [14] Ergul, A. (1996). GA-based examination scheduling experience at Middle East Technical University, in E. Burke and P. Ross (Ed.), *The Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, Springer-Verlag, pp. 212-226.
- [15] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA, Addison-Wesley.
- [16] Kendall G., and Mohd Hussin N., (2005). A Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology. In E. Burke. And M. Trick (eds.) *Proceedings of the fifth International Conference on the Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science 3616, pp. 270-293.
- [17] Kirkpatrick, S. Gelatt, C. and Vecchi, M. (1983). Optimization by simulated annealing, *Science*, 220, pp. 671-680.
- [18] Leong, T. and Yeong, W. (1987). Examination scheduling: a quadratic assignment perspective, *Proc. Int. Conf. On Optimization: Techniques and Applications*, Singapore, pp. 550-558.
- [19] Lotfi, V. and Cerveny, R. (1991). A final-exam-scheduling package, *J. of the Operational Research Society*, 42, pp. 205-216.
- [20] McCollum, B. and Newall, J. (2000). Introducing Optime: Examination Timetabling Software. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Germany, pp.485-490.
- [21] Mehta, N. (1981). The Application of a Graph Coloring Method to an Examination Scheduling Problem. *Interface*, 11 (5), pp.57-64.
- [22] Merlot L.T.G., Boland N, Hughes B.D. and Stuckey P.J. (2003). A Hybrid Algorithm for the Examination Timetabling Problem, In E.K. Burke and P. D. Causmaecker (eds.): *Proceedings of the fourth International Conference on the Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science 2740, Springer-Verlag, pp. 207-231.
- [23] Tarhini, A. and Mansour N., (1998). Natural Optimization Algorithms for Exam Scheduling, *Proceedings of International Conference on Computer Systems*, Irbid, IASTED, ACTA Press, pp. 124-127.
- [24] Thompson, J. and Dowsland, K. (1996). General cooling schedules for a simulated annealing based timetabling system, in E. Burke and P. Ross (Ed.), *The Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, Springer-Verlag, pp. 345-363.
- [25] Thompson, J. and Dowsland, K. (2005). Ant Colony Optimization for the examination scheduling problem, *Journal of the Operational Research Society*, 56, 426-438.
- [26] White, G and Xie B. S. (2000). Examination Timetables and Tabu Search With Longer Term Memory. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Germany, pp.184-201.
- [27] Wood, D. (1968). A system for computing university examination timetables, *Computer Journal*, 11, pp. 41-47.