Multi-objective Optimization for Exam Scheduling to Enhance the Educational Service Performance

Khaled S Abdallah [1]
[1] *Assistant Professor*
College of International Transport and Logistics
Arab Academy for Science, Technology, and Maritime Transport (AAST)
khaled.seif@aast.edu

Abstract

The exam scheduling problem has been a complex problem. In this paper, we consider planning for the scheduling of examinations of an Egyptian university. We provide and solve different formulations to the problem of scheduling examinations to minimize the total number of students who may have more than one exam at the same time taking into consideration capacity and other operational constraints. We further extend the problem and consider the case where we need to also minimize the number of students that have more than one exam per day, this results in a non-linear multi objective model where the school wants to minimize the number of conflicts (students with concurrent examination) and the number of students with more than one exam per day. We solve the problem using a genetic algorithm and applied it to schedule the final examination in one of the largest universities in Egypt leading to a tremendous reduction of the required man power and a great reduction in the planning errors.

1. Introduction

In recent years, the examination scheduling problem has been getting increasingly difficult as universities are enrolling more students into a wider variety of courses including an increasing number of combined degree courses and hybrid classrooms [1].

In this paper, we consider a large Egyptian university with more than nine campuses nationwide and still expanding, beside other international campuses. More specifically, we consider a school that is operating at near maximum capacity. As the school is operating at near maximum capacity, there is a large number of students registered per course, and most of the times, classes are filled up fast.

The school recommends a set of courses to be registered by students for each term. However, the school does not enforce this recommendation as it adopts a credit hour system. In addition to the already crowded classes, there aren't too many professors available to open extra classes when needed. All of the above, combined with the normal success/failure rates, result in that the students may have to register for other courses that have room available instead of the recommended courses by the school.

Planning for examinations under such dense student enrollment is a hard task as the exam schedule will likely result in a lot of students having 2 or more exams in the same time, not to mention the number of students having 2 or more exams in the same day.

The school under consideration doesn't have an appropriate planning software and relies on experience in developing a tentative schedule that goes through several feedback iterations. Realizing the problem, the author received an award to

build a comprehensive planning algorithm to plan the education process including exam scheduling, examination hall assignment, etc. Phase 1 of the project developed schedules with a minimum number of students with more than one exam at the same time. However, the author wants to go the further step of minimizing the number of students with more than one exam in the same day besides minimizing the number of students with several exams in the same day.

The rest of the paper is divided as follows. Section 2 introduces a literature review for the problem. Section 3 covers the problem description and formulation. Section 4 discusses the solution approach. Section 5 provides numerical results, section 6 provides final conclusion, and section 7 provides future work.

2. Literature review

The exam scheduling problem is defined in the literature as "the allocation of a set of exams to a number of periods (or time slots) so as to satisfy a set of constraints". It follows that different universities have differing views on what constitutes a good exam timetable. This has led to many different formulations of the problem considering different sets of constraints ([2]; [3]; [4]; [5]).

There are generally two groups of constraints: soft and hard constraints. Hard constraints have to be satisfied by any solution, while the soft constraints can be violated to an allowable extent whenever there is an absolute need. Burke et al. [2] discussed some constraints such as: no student should have to take more than one exam in consecutive periods, no student should have to take more than one exam on the same day, large exams should be held earlier in the exam period to allow enough time for marking of the scripts, some exams can only be held in a limited number of periods, and all exams should be scheduled in less than a particular number of periods.

Among the constraints that are mostly case specific, there are two constraints that are universal to all timetabling problems ([6]; [7]). The first is that no student is to be scheduled to take more than one exam at any given time. Violation of this constraint is referred to as a conflict. The second one is that for each period, there must be sufficient seats for all the exams that are scheduled for that period.

Several solution methods are proposed in the literature, however multi-objective models are limited. According to Cheong et al. [9], most of the existing literature in solving the examination scheduling problem use single-objective-based models ([6]; [10]; [11]; [1]; [12], [13]). Wong et al. [14] attempted a multi-objective approach based on a hybrid multi-objective evolutionary algorithm to maximize free time between exams and minimize the conflict. Cheong et al. [9] introduced a multi-objective evolutionary algorithm that minimizes the timetable length as well as the number of occurrences of students having to take exams in consecutive periods within the same day. This paper attempts to solve for minimizing the number of conflicts besides minimizing the number of students with more than one exam in the same day given a fixed length for the exam duration and a fixed number of slots per day.

3. Problem description and formulation

Consider a set of courses V whose exams are to be scheduled in a set of days I each with a set of slots J. The courses are covered in $1,2,\ldots T$ terms with N^c courses per term. $y_{a,t}$ is a binary constant that is equal to 1 when course a is recommended to be taught in term t. There are some courses whose exams must be conducted in the same day as they are taught by another school whose faculty visit us once a week. These courses are represented by a set S of sets of paired courses that should be conducted in the same day. n_a is the number of students registered in course a. Examination rooms have a total capacity of N and individual capacity of N_f for F rooms available. Let $\psi_{a,b}$ be a constant indicating the number of students registered in courses a and b (i.e., conflict if scheduled in the same time). Let $x_{a,i,j}$ be a binary variable with the value of 1 when course a's exam is scheduled in day i and slot j. Now the examination scheduling program for can be formulated as follows:

Minimize

$$\sum_{a,b\epsilon V, a\neq b, i\in I, j\in J} \left(\psi_{a,b} x_{a,i,j} x_{b,i,j}\right) \quad (1)$$

Subject to

$$\sum_{a\epsilon V} \left(x_{a,i,j} n_a\right) \leq N \quad \forall i \in I \text{ and } j \in J \quad (2)$$

$$\sum_{a\epsilon V, j\in J} \left(x_{a,i,j} y_{a,t}\right) \leq 1 \quad \forall i \in I \text{ and } t \in \{0,1,\dots T\} \quad (3)$$

$$\sum_{i\epsilon I, j\in J} \left(x_{a,i,j}\right) = 1 \quad \forall a \in V \quad (4)$$

$$\sum_{j\in J} \left(x_{a,i,j} - x_{b,i,j}\right) = 0 \quad \forall i \in I \text{ and } (a,b)\epsilon S \quad (5)$$

The objective is to minimize the number of students with conflicts. Constraints (2) ensure that the number of students in a certain slot in a certain day does not exceed the total capacity of the examination rooms. Constraints (3) ensure that no more than one course from a specific term is allowed in any examination day. Constraints (4) ensures that any course must be assigned only one slot in any day. Constraints (5) ensures the courses that are supposed to be assigned the same slot are indeed assigned the same day and slot.

The above formulation does not cover all the preferences of the colleges. Other preferences include:

1. Course precedes another course

In this case, we define S^1 as a set of paired courses *(a,b)* where course *a* needs to be scheduled before course *b*, formulated as follows

$$\left(x_{a,i,j} - \sum_{i'=i+1}^{I} x_{b,i',j}\right) = 0 \quad \forall i \in I \text{ and } (a,b) \in S^1 \quad (6)$$

2. Two Courses in the same day

In this case, we define S^2 as set of paired courses *(a,b)* where course *a* needs to be scheduled in the same day as course *b*, but not necessarily the same slot, formulated as follows

$$\sum_{j\in J} (x_{a,i,j} - x_{b,i,j}) = 0 \quad \forall i \in I \text{ and } (a,b) \in S^2 \quad (7)$$

3. In the same slot of another course

In this case, we define S^3 as set of paired courses *(a,b)* where course *a* needs to be scheduled in the same slot as course *b*, but not necessarily the same day, formulated as follows:

$$\sum_{i\in I} (x_{a,i,j} - x_{b,i,j}) = 0 \quad \forall j \in J \text{ and } (a,b) \in S^3 \quad (8)$$

4. In the same time of another course

In this case, we define S^4 as a set of paired courses *(a,b)* where course *a* needs to be scheduled in the same time as course *b*. This means that course *a* will be scheduled in the same day and slot of course *b*, formulated as follows:

$$(x_{a,i,j} - x_{b,i,j}) = 0 \quad \forall j \in J, i \in I, \text{and } (a,b) \in S^4 \quad (9)$$

5. Not in the same day of another course

In this case, we define S^5 as a set of paired courses *(a,b)* where course *a* should not be scheduled in the same day as course *b*, but can be scheduled in the same slot.

This can be performed by adding the following constraints

$$0 \leq \sum_{j\in J} (x_{a,i,j} + x_{b,i,j}) \leq 1 \quad \forall i \in I \text{ and } (a,b) \in S^5 \quad (10)$$

6. Not in the same slot of another course

In this case, we define S^6 as a set of paired courses *(a,b)* where course *a* should not be scheduled in the same slot as course *b*, but not necessarily in the same day; formulated as follows:

$$\sum_{i\in I} (x_{a,i,j} + x_{b,i,j}) \leq 1 \quad \forall j \in J \text{ and } (a,b) \in S^6 \quad (11)$$

7. Not in the same time of another course

In this case, we define S^7 as a set of paired courses *(a,b)* where course *a* should not be scheduled in the same time as course *b*; formulated as follows:

$$(x_{a,i,j} + x_{b,i,j}) \leq 1 \quad \forall j \in J, i \in I, \text{and } (a,b) \in S^7 \quad (12)$$

8. One course per term per day

Constraints 3 guarantees that no more than one course from the recommended courses for a specific term is to be selected. These constraints can be relaxed when the exam duration is less than the number of terms.

9. A course is not assigned a room

When an examination does not need to be assigned an examination hall, then the capacity requirement for that course is not added to the total requirement. Define R_a as a binary constant that indicates whether the capacity requirement for course a should be included, or not, formulated as follows:

$$\sum_{a\epsilon V}\left(x_{a,i,j}n_aR_a\right) \le N \;\; \forall i \in I \; and \; j \in J \; (13)$$

10. Scheduled on a specific Day

When a course needs to be scheduled on a specific day, the following constraints can be added:

$$\sum_{j\in J}(x_{a,i,j}) = 1 \qquad for \;\; given \; a,i \;\; (14)$$

11. Scheduled on a specific slot

When a course needs to be scheduled on a specific slot, it is formulated as follows:

$$\sum_{i\in I}(x_{a,i,j}) = 1 \qquad for \;\; given \; j,a \;\; (15)$$

12. Scheduled in Day and slot

When a course needs to be scheduled on a specific time, it is formulated as follows:

$$x_{a,i,j} = 1 \qquad \text{for given } j \text{ and } i \;\; (16)$$

13. Course/s that need specific Room/s

In this case, the capacity of all the courses that need the same rooms are checked against the available capacity when the courses are assigned the same time. Define S^8={(S_1^C, S_1^R), (S_2^C, S_2^R), ……., (S_k^C, S_k^R)} as a set of pairs (S_k^C, S_k^R) representing some courses and the rooms requested for them where $S_k^R = \{S_{k,1}^R, S_{k,2}^R, \ldots\ldots, S_{k,P}^R\}$ is the set of rooms that are to be assigned to courses in set S_k^C.

Then we add constraints that check the number of students enrolled in the courses in S_k^C provided that they are scheduled in the same time against the capacity of their associated rooms in S_k^R. If we define S_K^C as the capacity of room, then this can be formulated as follows

$$\sum_{S_{K,q}^C \epsilon S_K^C}\left(x_{S_{K,q},i,j}n_a\right) \le \sum_{S_{K,p}^R \epsilon S_K^R}\left(N_{S_{K,p}^R}\right)$$

$$\forall k \in K, p \in P, q \in Q \qquad (17)$$

It should be noted that S^8 needs also to be extended to include rooms that are included in several course/room combination. Let's for example assume that a certain room is common in (S_1^C, S_1^R), (S_2^C, S_2^R), and (S_3^C, S_3^R), then all of the combinations of these sets need to be added to S^8. So the following combinations are added ($S_1^C \cup S_2^C$, $S_1^R \cup S_2^R$),($S_2^C \cup S_3^C$, $S_2^R \cup S_3^R$),($S_1^C \cup S_3^C$, $S_1^R \cup S_3^R$), and ($S_1^C \cup S_2^C \cup S_3^C$, $S_1^R \cup S_2^R \cup S_3^R$).

14. Same schedule for different languages tracks

This can be achieved by two ways: the first method is by adding constraints that relate to other languages' courses to the English track. Let the number of tracks be n_L. In this case, one language's courses are indexed from 0 to |V|/n_L-1, while the second language's courses are indexed from |V|/ n_L to 2|V|/ n_{L-1}, etc. Then, the following constraints are added

$$(x_{a,i,j} - x_{a+l*\frac{|V|}{n_L},i,j}) = 0$$

$$\forall \, l \in n_L, j \in J, i \in I, \qquad a = 0,1,.\frac{|V|}{n_L} - 1 \; (18)$$

A better alternative is to use the formulation for the English track and incorporate the other language requirements into the constraints (e.g., the capacity check will include the capacity of classes taught in all languages.)

15. Excluded in the schedule

Some courses may not be scheduled for different reasons such as when there is no final exam for them. In this case, we define S^9 as a set of the courses that are excluded from the schedule. Hence constraints (4) are replaced by:

$$\sum_{i\epsilon I, j\in J} x_{a,i,j} = 1 \qquad \forall a \in V/S^9 \;\; (19)$$

16. Excluded from term check

Some courses are not included in the recommended plan but are known to be taught in certain terms under specific conditions such as non-credit language courses. The college may want to exclude this course from the one course per term per day. Define S^{10} as a set of courses that need to excluded from the term check.

Constraints (3) can then be adjusted as follows:

$$\sum_{a \epsilon V/S^{10}, j \in J} \left(x_{a,i,j} y_{a,t}\right) \leq 1 \quad \forall i \in I \text{ and } t \in \{0,1,\dots T\} \quad (20)$$

17. Minimizing number of students with more than one exam in different slots per day

In this case, the objective function becomes much more complex by adding several terms to compute the number of students that have 2,3….and up to *J* exams per day as follows

$$\sum_{u=2}^{J} \sum_{\substack{a_1<a_2<,\dots\dots<a_u \epsilon V \\ and \\ j_1,j_2,\dots\dots j_u \epsilon J}} \left(\psi_{a_1,a_2,\dots\dots a_u} x_{a_1,i,J_1} x_{a_2,i,J_2} \dots \dots . x_{a_u,i,J_u}\right) (21)$$

The inequality $a_1 < a_2 <, \dots \dots < a_u \epsilon V$ ensures that the different combination of courses are not computed at different times by using the exact same courses but in different order.

This term when added to the objective, it becomes non-linear. Moreover, there are multiple objectives in this formulation. The first objective is to minimize the number of students with conflicts, and the second group of objectives minimize the students with 2 or more exams in the same day but in different times, called hereafter as clash. The objective is formed by a linear combination of both.

All of the above formulations except for the objective of minimizing students with clash are actually quadratic integer models that can be solved efficiently using IBM ILOG CPLEX given enough time.

4. Solution approach

In order to solve the problem, a genetic algorithm is developed. The algorithm starts with constructing chromosomes formed of two columns. The first column represents assigned examination day ranging from 1 up to the examination duration, and the other column represents the assigned slot. Each row represents a course's assigned examination day and slot. Each group of courses that belong to the same term are grouped together in blocks. A block has a length equal to the exam duration, with any extra row not assigned a course represents a dummy course. As a result of the equal size blocks, some of the constraints will always be maintained during the search as discussed later.

All courses that are not assigned specific terms in the recommended course plan are grouped together in another block. The total number of blocks is equal to the number of terms plus 1. A complete chromosome is shown in figure 1.

4.1. Initial solution

An initial solution of 100 chromosomes is formed using the following approach. For each block representing a term, random generated combinatorial integers from 1 to exam duration are assigned to days. Note that being combinatorial integers, no integer is repeated. Hence, there are not any courses from the same term that are assigned the same day. Then a randomly generated integer from 1 to the allowed number of slots per days is assigned to each row. Note that a slot may be repeated several times within a block, but does not affect feasibility.

It should be noted that the generated initial solutions don't necessarily have to be feasible. An example is shown in figure 2.

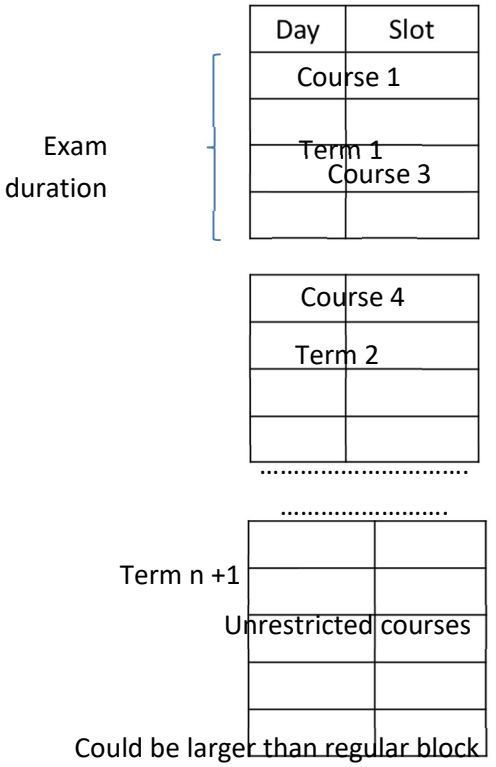

Figure 1. A complete chromosome setup for a possible solution

Day	Slot
2	1
1	1
3	2
1	1
3	2
2	1
…..	…..
4	1
4	2
2	2
1	1

Figure 2. A chromosome with initial solution for three days examination duration

4.2. Fitting function

In order for the algorithm to evaluate how well a solution is, it first loops through each pairwise courses within a slot. Two values are computed. The first value is the share of conflict per course which means the conflict resulting from a course being scheduled in the same time with other courses in the current chromosome. The second value is the total conflict resulting from all the courses being scheduled in the same time in the current chromosome. The next step is to loop through all the courses scheduled in a given day to compute the share of the same day clash contribution of one course, and total clash for each day.

The total conflict and total clash of a schedule are computed concurrently while computing the above values along with a check of the capacity requirement and other precedence requirements.

In the first ten iterations, we allow infeasible solutions; however, later when there is infeasibility, the total conflict for an infeasible solution is considered a huge value.

Finally, the fitting function computes which is a weighted sum of the total same slot conflict and the total same day conflict.

4.3. Search operations

As the current chromosome setting allows the exchange of the days and slots of one block with another block without violating the term requirements, we allow cross over as follows.

In the first half of allowed iterations, we select candidate parents randomly to allow appropriate exploring of the solution space. In the second half, we select parents using Roulette wheel selection with

$$P_i = \frac{\exp(-2 * \frac{conflict}{worst\ conflict\ in\ the\ pool})}{\sum_{j=1}^{N} \exp(-2 * \frac{conflict}{worst\ conflict\ in\ the\ pool})}$$

Two children chromosomes are generated by exchanging the day and slot assignments of the two blocks that have the maximum number of students with conflict as shown in figure 3.

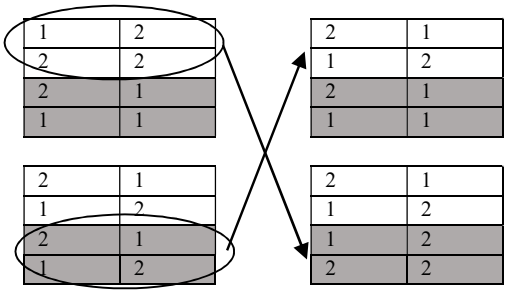

Figure 3. Crossover of regular courses.

The cross over for the unrestricted is different. We use the same parent selection as before. We select the parent with the maximum number of conflicts between the two parents. A start point and end point are selected based on a cumulative relative frequency of the number of conflicts resulting from a course relative to the total conflict of the unrestricted courses block as shown in figure 4.

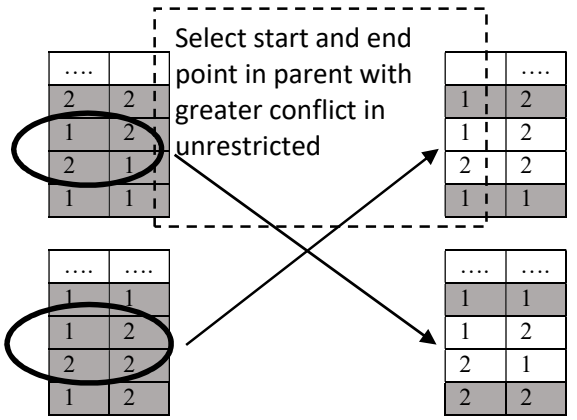

Figure 4. Crossover of unrestricted courses.

The algorithm also does a slot cross over. A chromosome is randomly selected. Then the day with the most conflict is selected. Within the selected day, the course with the maximum and minimum share of conflicts are selected and their slots are swapped as shown in figure 5.

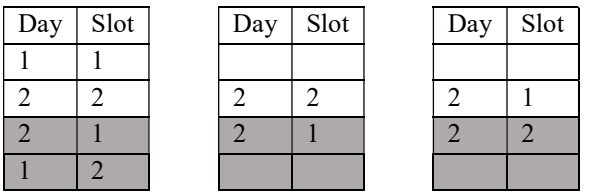

Figure 5. Slot crossover

As the number of conflicts decrease with the progression of the algorithm, there is more than one course that does not contribute in any conflict. In this case, selection of the two courses with the maximum and minimum conflict contribution wouldn't be the best action. It will be better to choose a course among the top 3 courses according to their conflict contribution, and another course among the top 3 with the least conflict contribution.

In order not to get stuck in a local sub optimal solution, the algorithm performs mutations in two ways: day and slot mutations. For both cases the algorithm selects a chromosome randomly. Based on the cumulative frequency of each course's conflict contribution relative to the total conflict of the selected chromosome, the algorithm generates a probability that is transformed into an equivalent course selection for mutation.

With a course selected for mutation, the algorithm can mutate the slot directly by choosing a value from 1 to the maximum slots available per day other than the current value.

For day mutation, when the day of the selected course is mutated, it is then assigned a day that is already assigned to another course in the same block. The other course should then be assigned the day originally assigned to the selected course.

Instead of randomly assigning a new day to the selected course, a third approach is adopted. We randomly select one course among the top three courses with the most conflict contributions and another course among the top three courses with the least conflict contribution. After swapping the days of the two selected courses, there would be two courses with the same assigned day within their respective blocks, so we switch the assigned day of the other course within each block with the assigned day originally assigned to the respective selected course as shown in figure 6.

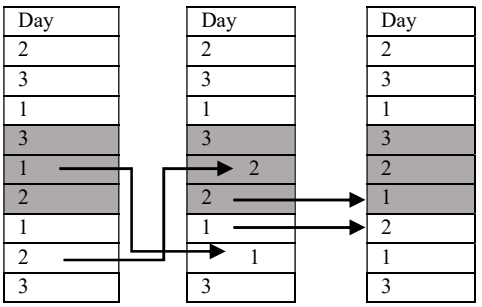

Figure 6. Mutation of Days

4.4. Obtaining Multi-objective solution

While progressing through the iterations, infeasible solutions are allowed half way through the iterations. Then infeasible solutions are assigned a very high value in order not to be selected in later iterations.

Due to the combinatorial nature of the

problem, many chromosomes could be generated despite having the same solution, but arranged in different orders. So, only unique solutions are allowed after half way through the iterations.

For a single objective, the new pool of solutions is arranged according to the conflict after an iteration ends. While for the multi-objective of minimizing the clash in addition to the conflict, the algorithm selects new generations in two techniques:

Case 1: Arranging the new generation according to conflict then clash. This is equivalent to the Pareto importance, as the former objective is far more important for the school than the latter.

Case 2: Arranging according to conflict then according to a weighted average of conflict and clash.

In later iterations and as the algorithm converges to the optimal solution, we allow non-unique solutions to improve the quality of the second objective, i.e., by minimizing clash after the algorithm had already minimized the number of conflicts. This setting allows both approaches to favor the objective of minimizing conflict as it is more important.

The algorithm steps are as follows:

1. Set GA parameters.
2. Initial population.
3. Run fit function for each chromosome.
4. Repeat for the max number of iterations.
5. Term crossover via random selection for the first half iterations then via Roulette wheel selection.
6. Apply unrestricted term crossover.
7. Apply regular slot mutation.
8. Apply slot mutation.
9. Apply day mutation.
10. Apply mutation for unrestricted courses.
11. Append new solutions to the population.
12. Arrange ascendingly according to conflict, then according to clash or weighted average of conflict and clash.
13. Halfway through iterations, the best chromosomes are kept, then later, the best unique solutions are kept. And loop.

5. Numerical Results

The algorithm is tested on a large school. It offers two languages tracks, each with three majors. Most of the courses are taught in the three majors. The school has eight terms with six courses per semester. There are about 1,285 students with more than 6,500 course student enrolments. There are 21 examination halls available, each with a capacity of 30 students.

The quadratic model developed in this paper was solved using two hours run of IBM ILOG Cplex 32 bit on a laptop with Intel i3-4030U CPU clocked at 1.9GHz and 4GB ram. The results are shown in table 1 and 2 for two slots and three slots per day, respectively.

Table 1. Results of running the exact Quadratic model for 2 slots per day.

Days	6	7	8	9	10
Conflict	86	40	49	21	2
Clash	579	512	414	325	377

Table 2. Results of running the exact Quadratic model for 3 slots per day.

Days	6	7	8	9	10
Conflict	10	9	1	1	0
Clash	694	685	486	595	474

Although the exact model was successful in finding good solutions, the values of the clash are, however, very high.

The Genetic Algorithm is then applied to the same set of data. Two cases are analyzed: Case 1, when the GA arranges the pool according to conflict then clash, and case 2 when GA arranges the pool according to conflict then weighted average conflict and clash. The results are shown in figures 7 and 8 for conflict and clash, respectively, for three slots per day. Figures 9 and 10 show the same for two slots per day.

It is observed that for 3 slots per day, case 2 consistently results in lower conflict compared to case 1, but case 1 results in lower clash when the duration is not tight (i.e., greater than the number of terms).

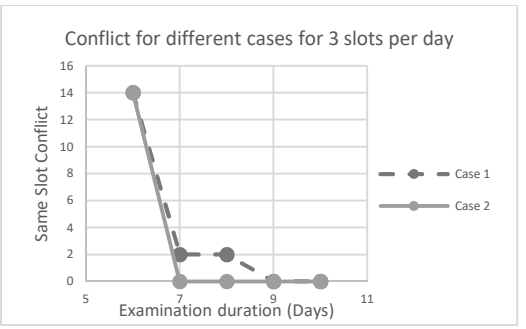

Figure 7. Conflict for 3 slots per day.

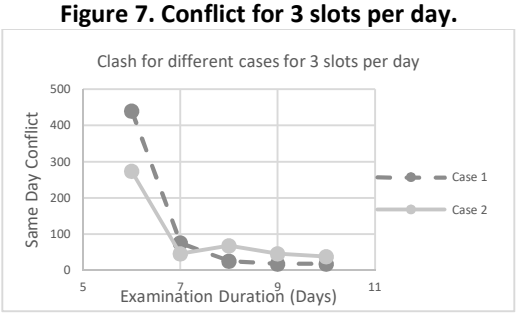

Figure 8. Clash for 3 slots per day.

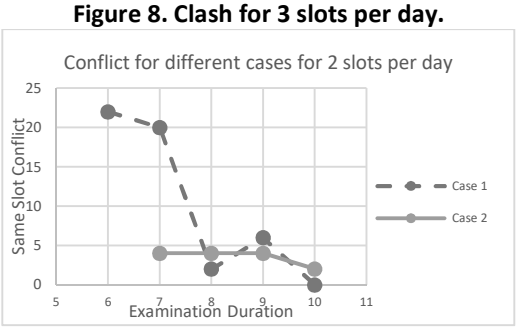

Figure 9. Conflict for 2 slots per day.

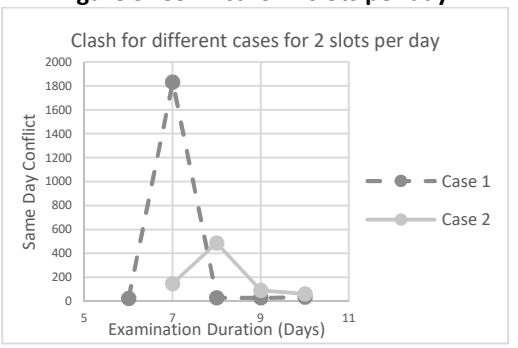

Figure 10. Clash for 2 slots per day.

Once again, it is observed that case 2 generally results in lower conflict compared to case 1, but case 1 results in lower conflict when the schedule is not tight. Moreover, case 2 was unable to obtain a solution when the schedule was very tight with number of days equal to the number of terms and two slots per day.

6. Conclusion

In this paper, two types of models are introduced for exam scheduling: an exact quadratic model to minimize number of conflicts, and a genetic algorithm that adds another objective to minimize the number of clashes in addition to the number of conflicts.

The models take into consideration a large number of operational constraints that are general in nature and not only tied to the Egyptian university school under consideration, but they can also be applied at other universities as well.

Both models performed well and consistently provided a good solution for the conflict. However, the quadratic model doesn't always result in a small number of clashes.

The proposed GA consistently resulted in a good number of clashes. Within the proposed GA model, two cases are suggested to select the pool of chromosomes between iterations. From the numerical results, it is recommended to use case 2 for tight schedules and case 1 for loose schedules.

7. Future Work

A possible extension to the proposed model is to develop a model to add another objective to the objectives considered in this paper to minimize room sharing between colleges when the exam duration is very tight and the capacity would not be enough to build a schedule.

8. References

[1] Merlot, L. T. G., Boland N., Hughes, B. D., & Stuckey, P. J. (2003). A hybrid algorithm for the examination timetabling problem. In E. K. Burke & P. De Causmaecker (Eds.), Lecture notes in computer science: Vol. 2740. Proceedings of the 4th international conference on the practice and theory of automated timetabling, PATAT 2002, Gent, Belgium (pp. 207–231). Berlin: Springer.

[2] Burke, E. K., Elliman, D. G., Ford, P. H., & Weare, R. F. (1996b). Examination timetabling in British universities—a survey. In E. K. Burke & P. Ross (Eds.), Lecture notes in computer science: Vol. 1153. Proceedings

of the 1st international conference on the practice and theory of automated timetabling, PATAT 1995, Edinburgh, Scotland (pp. 76–90). Berlin: Springer.

[3] Carter, M. W., & Laporte, G. (1996). Recent developments in practical examination timetabling. In E. K. Burke & P. Ross (Eds.), Lecture notes in computer science: Vol. 1153. Proceedings of the 1st international conference on the practice and theory of automated timetabling, PATAT 1995, Edinburgh, Scotland (pp. 3–21). Berlin: Springer

[4] Schaerf, A. (1999). A survey of automated timetabling. Artificial Intelligence Review, 13(2), 87–127).

[5] Qu R., Burke EK (2009) Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. J Oper Res Soc 60:1273–1285

[6] Burke, E. K., Newall, J. P., & Weare, R. F. (1996a). A memetic algorithm for university exam timetabling. In E. K. Burke & P. Ross (Eds.), Lecture notes in computer science: Vol. 1153. Proceedings of the 1st international conference on the practice and theory of automated timetabling, PATAT 1995, Edinburgh, Scotland (pp. 241–250). Berlin: Springer

[7] Chan, C. K., Gooi, H. B., & Lim, M. H. (2002). Co-evolutionary algorithm approach to a university timetable system. In Proceedings of the 2002 congress on evolutionary computation, CEC 2002, Honolulu, HI, USA (Vol. 2, pp. 1946–1951)

[8] Cheong CY, Tan KC, Veeravalli B (2009) A multi-objective evolutionary algorithm for examination timetabling. J Sched 12:121–146

[9] Caramia, M., Dell'Olmo, P., & Italiano, G. F. (2001). New algorithms for examination timetabling. In S. Näher & D. Wagner (Eds.), Lecture notes in computer science: Vol. 1982. Algorithm Engineering 4th International Workshop, WAE 2000, Saarbrücken, Germany (pp. 230–241). Berlin: Springer.

[10] Di Gaspero, L., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. In E. K. Burke & W. Erben (Eds.), Lecture notes in computer science: Vol. 2079. Proceedings of the 3rd international conference on the practice and theory of automated timetabling, PATAT 2000, Konstanz, Germany (pp. 104– 117). Berlin: Springer.

[11] Abdullah, S., Ahmadi, S., Burke, E. K., & Dror, M. (2007a). Investigating Ahuja–Orlin's large neighbourhood search approach for examination timetabling. OR Spectrum, 29(2), 351–372.

[12] Abdullah, S., Ahmadi, S., Burke, E. K., Dror, M., & McCollum, B. (2007b). A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem. Journal of the Operational Research Society, 58, 1494–1502.

[13] Wong, T., Côté, P., & Sabourin, R. (2004). A hybrid MOEA for the capacitated exam proximity problem. In Proceedings of the 2004 congress on evolutionary computation, CEC 2004, Portland, OR, USA (Vol. 2, pp. 1495–1501).