University Course Scheduling Using Evolutionary Algorithms

Conference Paper · September 2009

DOI: 10.1109/ICCGI.2009.15 · Source: IEEE Xplore

4 authors, including:

Mohammed Aldasht
Palestine Polytechnic University
14 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Safa Adi
Palestine Polytechnic University
2 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Mohammad Abu Qbeitah
Zayed University
2 PUBLICATIONS **10** CITATIONS
SEE PROFILE

University Course Scheduling Using Evolutionary Algorithms

Mohammed Aldasht
Palestine Polytechnic University
Hebron, Palestine
mohammed@ppu.edu

Mahmoud Alsaheb
Palestine Polytechnic University
Hebron, Palestine
alsaheb@ppu.edu

Safa Adi
Palestine Polytechnic University
Hebron, Palestine
safa_adi@ppu.edu

Mohammad Abu Qopita
Palestine Polytechnic University
Hebron, Palestine
mohammadqb@exceed.ae

***Abstract*—This paper presents a new heuristic based on evolutionary algorithms and applied to the university course scheduling problem, where a feasible and comfort time tables are required. Here, the idea is to use an evolution program which is a stochastic optimization strategy similar to genetic algorithms. The main difference is that evolutionary programming insists on the behavioral linkage between parents and their offspring rather than seeking to emulate specific genetic operators as observed in nature. The paper starts by defining the problem and determining the constraints under which the solution should be found. Then, the problem model is described with a set of courses, rooms, instructors, and student groups. Finally, the proposed methodology is applied on a real data set from one of the four colleges of our university. Results show that our methodology permits more robust exploration for the search space of the designated problem which gives more optimized time schedules than those performed manually. The obtained results also show that the proposed solutions can solve many registration difficulties.**

Keywords-university course scheduling; timetabling; evolutionary algorithms

I. INTRODUCTION

University course scheduling can be considered as an instance of what so called timetabling problem, which appears to be a tedious job in every academic institute once or twice a year. The problem involves the scheduling of classes, students, teachers and rooms at a fixed number of timeslots, in a way that satisfies a set of hard constraints and minimize the cost of a set of soft constraints [1]. This problem is considered to be NP-complete or even NP-hard problem [15]; this implies that there is no known polynomial time algorithm that can guarantee finding the best solution [2]. Traditionally, the problem is solved manually by trial and hit method, where a valid solution is not guaranteed. Even if a valid solution is found, it is likely to miss far better solutions. These uncertainties have motivated scientific studies of the problem to develop automated solution techniques for it [1][3][4][6-10][13].

In this work, the university course scheduling problem will be handled; this includes the process of assigning a set of classrooms and a set of instructors to a given set of courses taking into account a set of hard and soft constraints. An example of a hard constraints: no person can be in more than one place at a time. Another example is that the total resources allocated to a timeslot must be less than or equal to the resources that are available in that timeslot. Soft constraints, on the other hand, is like a teacher or a group of students should not have consecutive classes in different and far distant places and other Individual preferences [4][5].

The remainder of this paper is structured as follows: Section II describes a number of relevant works. Section III describes the problem modelling which is important for the description of how the solution is implemented. Then, Section IV describes the evolutionary algorithm used to explore the search space defined by the model in Section III. Section V gives the experimental results obtained and the corresponding comments and finally, Section VI provides the conclusion of the paper and some future work.

II. RELATED WORK

There are many solutions proposed in the literature to solve such a problem. In [1], two approaches are proposed to solve the problem: (1) MGA (Modified Genetic Algorithm) and (2) CGA (Cooperative Genetic Algorithm). The authors were aiming at enhancing the algorithm performance using a modified genetic operators or cooperative genetic method, respectively. In [6], the authors report on the development of a general tool called NExshed; this tool is implemented as a plug-in for Microsoft's Excel. In [7], the authors used a hybrid algorithm for solving the timetabling problem. This algorithm combines an integer programming approach, a greedy heuristic and a modified simulated annealing algorithm. In [8], a method is presented to translate the course requirements into a relational formula. Thus, allowing the scheduling problem to be solved with existing tools such as the Allay Analyzer. In [9], the authors address academic course scheduling in a networked environment using intelligent agents within a decision support framework. In [10], the authors have investigated a variety of approaches based on simulated annealing. This includes mean-field annealing, simulated annealing with three different cooling schedules, and the use of a rule-based preprocessor to provide a good initial solution for annealing. In [3], a multi-agent system for university course scheduling is proposed. In [4], the authors improved a complete approach using ILP (Integer Linear

Programming) to solve the problem. The ILP model is developed and solved using the three advanced ILP solvers based on GA (Genetic Algorithm) and SAT (Boolean Satisfiability Technique).

Evidently, the performance of all different evolutionary algorithms, most notably GAs, in solving the university course timetabling problem has been widely studied. Those algorithms are highly dependent on their special operators, i.e., mutation and crossover. Because the search space of the problem is very complex, a more general approach is needed to guarantee a robust exploration and to avoid the local minimum problem which is frequent in such search spaces. In this paper, our approach permits a simple representation for the solution and a robust exploration for the search space.

On the other hand, most of the above mentioned approaches model the problem according to a set of different parameters defined to be suitable for a given institution. Therefore, institutions' private rules and conditions related to their timetables should be considered.

III. PROBLEM DESCRIPTION AND MODELLING

The college of Administrative Sciences and Informatics in Palestine Polytechnic University is considered for prototyping. In this college, there are four academic programs. Courses are offered at the beginning of each semester for more than twenty student groups. There exists 5 work days a week (Sunday to Thursday). On Sundays, Tuesdays and Thursdays there are nine 60-minutes timeslots a day. On Mondays and Wednesdays there are six 90-minutes timeslots a day.

To handle the problem, six different sets have been defined: students, instructors, courses, classrooms, timeslots, and the set of constraints. Then, the problem is formulated as: P = {S, T, C, R, L, O}. Where: S = {s_1, s_2, …, s_i} is the set of student groups. T = {t_1, t_2, …, t_j} is the set of instructors. C = {c_1, c_2, ……, c_n} is the set of offered courses. Each entry in C is a vector used to describe the following: group of students which a given course is offered for, course number, section, maximum capacity of the section, instructor of the section, number of theoretical hours of the course, number of practical hours of the course, type of the practical hours and finally the period that the instructor should be present in the lab. R = {r_1, r_2, ….., r_m} is the set of classrooms. Each entry in R is a vector used to describe maximum capacity of a given room, a flag to inform weather the room has a data show and classroom type. L = {l_1, l_2, ….., l_k} is the set of timeslots of the week. O = {o1, o_2, ……, o_q} is the set of constraints. Where o_i: is the penalty weight (cost) of the constraint. Hard constraints are assigned a big cost in case of violation, while each soft constraint is assigned a small cost in case of violation.

The set of solutions for the problem is a large set, where each solution will be evaluated using an indicator to determine the fitness of the solution.

A. Solution definition

The solution is represented as: G = {g_1, g_2,…., g_w}. Where G is the set of genes which constitute the Individual or *chromosome.* As shown in Figure 1, each gene is a vector containing the following: course number, room number, practical hour timeslot, lab number and theoretical hour timeslot.

G_1	G_2	…..	G_i	….	G_w
course$_1$	*course$_2$*	……	*Course$_i$*	…….	*Course$_w$*
room_no	*room_no*		*room_no*		*room_no*
ph_tslot	*ph_tslot*		*ph_tslot*		*ph_tslot*
lab_no	*lab_no*		*lab_no*		*lab_no*
th_tslot	*th_tslot*		*th_tslot*		*th_tslot*

Figure 1. The (Individual).

As mentioned in the introduction, the university course timetabling problem is considered to be NP-hard. This means that the search domain of such a problem is very complex. Thus, each course section will need a suitable classroom and/or lab and time slots to be assigned so that they are suitable for the student groups and the lecturer. In order to reduce the search time and complexity, the algorithm randomly chooses the suitable place for the course section. Then, the search is dedicated for optimizing the solution taking into account the hard and soft constraints for the students and lecturers.

The Individual is constructed in a way that every gene has different range of options when choosing timeslot and classroom; Thus, course section represented in gene G_1 has the maximum number of options available when randomly selecting classrooms and timeslots. Course section represented in gene G_2 has 1 option less than those available for gene G_1 and the last gene G_w has the least number of options.

The instance chosen for application has a maximum of 14 classrooms and 7 labs. Also, through the week, there exists 41 theoretical classes "1 hour each" and 13 practical classes or labs "3 hours each". That means the maximum room capacity is 41*14 = 574 theoretical classes, and the maximum lab capacity is 13*7 = 91 practical classes. In Figure 2 the value 1 means timeslot is available and 0 means timeslot is unavailable.

Days	Time slots									
	Sunday	1	1	1	1	1	0	1	1	1
	Monday	1	1	1	1	1	1	1	1	1
	Tuesday	1	1	1	1	0	0	1	1	1
	Wednesday	1	1	1	1	1	1	1	1	1
	Thursday	1	1	1	1	1	0	1	1	1

Figure 2. The timeslots.

B. Constraints

The final solution is required to satisfy a set of constraints. These constraints are divided into hard constraints which must be satisfied and soft constraints

which should be satisfied. The set of constraints and their weights are as follows:

Hard constraints are:

1. An instructor must not have more than one class at any given timeslot.
2. An instructor should be assigned classes only in his available time, like part timers.
3. A classroom must not be assigned more than one class at any given timeslot.
4. Number of students in any lecture should be less than or equal to the maximum capacity of the classroom. (The cost increases as the number of students above the capacity of classroom increases. Each student above the capacity increases the cost by a constant until the number of students above the capacity is greater than a given threshold.
5. At any given timeslot no student group can have more than one class.

Soft constraints have a violation cost of a small constant which is relative to the constraint importance.

Soft constraints are:

1. Students should not have more than three classes consecutively.
2. Instructors should not have long free time between lectures.
3. Instructors should not have very late classes daily.
4. A group of students should not have consecutive classes in different and far distant places.
5. Students should not have long free time between lectures.

IV. EVOLUTIONARY ALGORITHM

A wide range of algorithms can be classified as evolutionary. Because most of the proposed algorithms are highly dependent on their special operators of mutation and crossover, we have decided to design an evolutionary algorithm that doesn't contain the crossover operator in order to permit the Individuals evolving independently and to get more robust exploration. In this work, EP (evolutionary programming) is used which is a stochastic optimization strategy [16]. The main difference between EP and GAs is that the typical GA involves encoding the problem solutions as a string of representative signs, i.e., binary representation [11][12]. In EP, the representation comes from the nature of the problem which permits a more simple representation of the solution. Furthermore, EP generally depends on mutation process and not on the recombination like crossover to produce offspring.

Evolutionary programming process begins with selecting parents from population to reproduce. Then, parents' properties are mutated to produce offspring who are added to the population which doubles the population size. Then, the algorithm discards unsuitable Individuals and selects the best Individuals from the population to bring it back to the first size in order to use it in the next generation.

EP technique consists of the following steps [16]:

1. *Initialize the population.*
2. *Expose the population to the environment.*
3. *Calculate fitness for each member.*
4. *Randomly mutate each "parent" population member.*
5. *Evaluate parents and offspring.*
6. *Select members of new population.*
7. *Go to step 2 until some condition is met.*

In this work, the mentioned steps can be described as follows:

Step 1: load the needed data from different files which are: student groups' timeslots, teachers' timeslots and rooms' timeslots. The algorithm will take knowledge from the mentioned files about the time availability for students, teachers and rooms respectively. Initially, it is supposed that students are available all the time along the five working days of the week, while teachers and rooms may have some time availability constraints.

After data initialization, an initial population is constructed according to the loaded data and constraints producing the first population.

Step 2: population cost is measured as the sum of the cost for each gene in the Individual. The cost is determined by scanning the genes of the Individual orderly for any constraints violation.

Step 3: the fitness is calculated by the simple function $f(x)=1/(\text{cost}(x)+1)$. Where cost(x) is the sum of cost for the Individual x.

Step 4: a random modification is done on the non-zero cost genes in the Individual. In our algorithm every Individual is reordered according to the genes' costs. Where genes with a zero cost are placed at the beginning of the Individual. Then, the mutation starts from the first gene whose cost is greater than zero. This mutation is called RM (Reordered Mutation).

Step 5: the next step is the selection of the Individuals of the next population depending on the calculated fitness values. In our algorithm the roulette-wheel selection with elitism is used.

V. EXPERIMENTAL RESULTS

Our application is implemented using C programming on a workstation with dual processor Intel (R) Xeon™ at 3 GH, 512 KB cache, 2 GB memory and runs under Linux. In our experiments real data available on the database of Palestine Polytechnic University is used. The data is coded into numbers and stored in text files which are used as inputs to the program

A. Results

As the evolution time of our algorithm is greatly dependent on the random function in use, Figure 3 shows the nature of the random function used in our experiments. It is clear that the function has a uniform distribution with mean 49.96362 and variance 832.741.

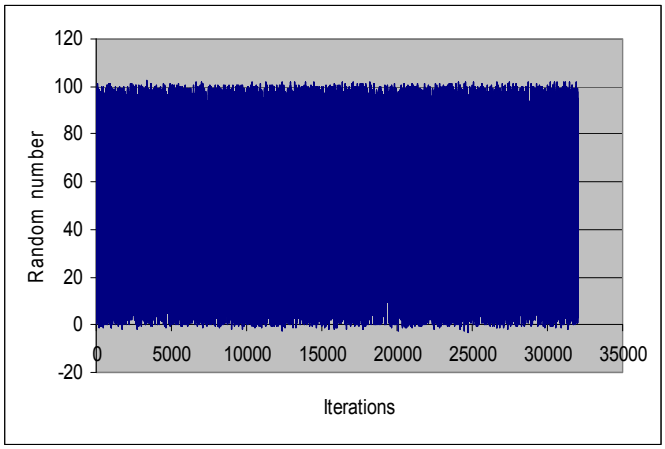

Figure 3. Random function distribution.

The following figures, i.e., Figure 4, 5 and 6, correspond to an experiment which is done with population size of 30 Individuals and number of generations 500. The experiment is repeated 3 times. Then, the average is taken to draw Figures 4, 5 and 6. After these iterations, the best Individual has been found in generation 228. The overall execution time was about 9 minutes.

Figure 4 shows the relation between number of generations and the average fitness per generation.

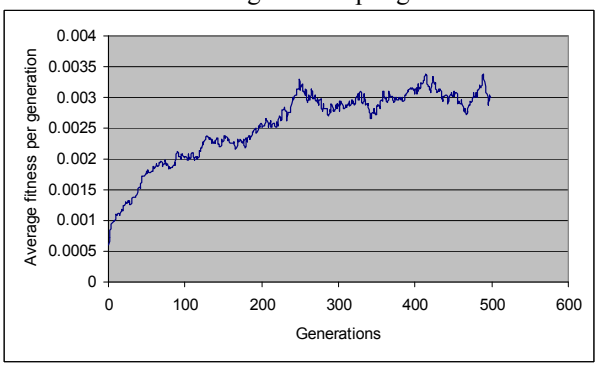

Figure 4. The average fitness evolution.

Figure 5 shows the evolution of the average best fitness per generation in front of the number of generations.

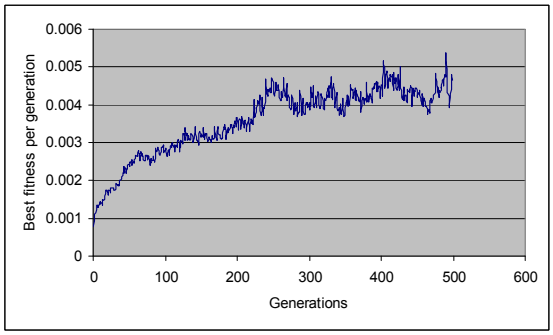

Figure 5. The best fitness evolution.

In the same way, Figure 6 shows the relation between the number of generations and the average cost for best fitness per generation.

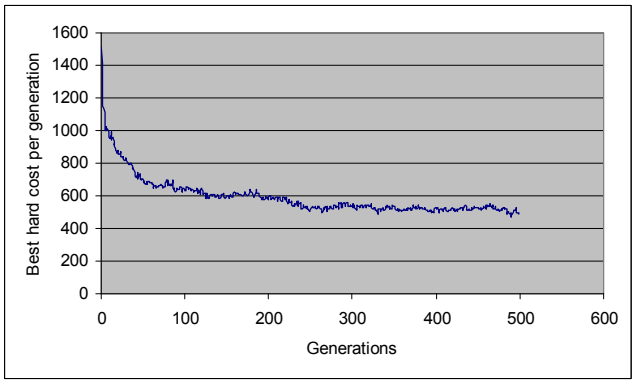

Figure 6. The cost evolution.

B. *Acceptance testing*

To perform the acceptance testing, a random sample of 20 students distributed among the different majors were asked to answer 5 questions as follows:

1. Credit hours wanted to register
2. Credit hours registered using the manual schedule
3. Credit hours registered using the proposed schedule
4. Degree of satisfaction from the manual schedule
5. Degree of satisfaction from the proposed schedule

The results of the questionnaire are shown in table 1.

TABLE 1. SUMMARY OF RESULTS OF THE QUESTIONNAIRE

		Sample	Students who had registration problems	Students who hadn't registration problems
Percentage		100%	35%	65%
Average credit hours wanted to be registered by the student		16.6	17.14	16.3
Average credit hours registered using the manual schedule		15.7	14.4	16.4
Average credit hours registered using the proposed schedule		16.5	16.7	16.4
Satisfaction from the manual schedule	**Satisfied or very satisfied**	60%	57%	62%
	Not satisfied	40%	43%	38%
Satisfaction from the proposed schedule	**Satisfied or very satisfied**	80%	100%	70%
	Not satisfied	20%	0%	30%

VI. CONCLUSION AND FUTURE WORK

In this paper, an EP algorithm is proposed to solve the university course scheduling problem. The algorithm is capable to achieve feasible and comfort solutions for the problem.

Conclusion can be summarized in the following points:

1. A problem model is introduced suitable for a wide range of academic institutions.
2. An evolutionary algorithm is proposed with special mutation operator called RM which is proved to converge to optimal solutions.
3. A framework is implemented to construct a search space in order to apply the search methodology.
4. The proposed methodology is applied on a real data set and can be extended to consider other institutions.
5. Results show that our methodology can give time tables with up to 97% fulfillment of soft constrains and up to 100% fulfillment of hard constrains.
6. Acceptance testing shows that our work has solved registration problems for more than a third of the students.
7. Also, Table 1 shows that our proposed schedule has satisfied the student more than the one placed manually.

It's worthwhile to mention a number of future works that is desired to enhance the evolutionary algorithm itself and to achieve better solutions for the problem.

Future work can be summarized in the following points:

- Using parallel implementation of the methodology to reduce the time and get faster convergence.
- Modify the evolutionary algorithm to work as a multi-objective optimizer in order to reduce the soft cost.
- Trying other methods and evolutionary techniques such as particle swarm optimization PSO algorithms [14].

ACKNOWLEDGMENT

This paper has been supported by the Deanship of Scientific Research and Higher Education at Palestine Polytechnic University, under the grant for the research project "intelligent algorithms for prediction and optimization in heterogeneous clusters"

REFERENCES

[1] M. Ghaemi, M. Vakili and A. Aghagolzadeh, "Using a genetic algorithm optimizer tool to solve University timetable scheduling problem", 9th International Symposium on Signal Processing and Its Applications, pp. 1-4, 2007.

[2] P. Pongcharoen, W. Promtet and C. Hicks, "Stochastic Optimisation Timetabling Tool for University Course Scheduling", International Journal of Production Economics, 2007.

[3] M. Opera, "Multi-Agent System for University Course Timetable Scheduling", The 1st International Conference on Virtual Learning, pp. 231-237, 2006.

[4] A. Wasfy and F. Aloul, "Solving University Class Scheduling Problem Using Advances ILP Techniques", American University of Sharjah publications, 2006.

[5] S. Petrovic, "Towards the Benchmarks for Scheduling Problems", Proceedings of the Workshop "Scheduling a Scheduling Competition" held in conjunction with the 17th International Conference on Automated Planning & Scheduling (ICAPS '07), Providence, Rhode Island, USA, September 22-26, 2007.

[6] S. Chitnis, M. Yennamani and G. Gupta, "NExSched: Solving Constraints Satisfaction problems with the Spreadsheet Paradigm", Unpublished technical report, 2006.

[7] A. Gunawan, K. Ming Ng and K. Poh "Solving the Teacher Assignment-Course Scheduling Problem by A hybrid Algorithm", International Journal of Computer, Information and System Scienceand Engineering, Volume1 number 2 ISSN 1307-2331, pp. 136-141, 2007.

[8] V. Yeung, "Declarative Configuration Applied to Course Scheduling", MSc. thesis MIT, 2000.

[9] P. Dasgupta and D. Khazanchi, "Adaptive decision Support for Academic Course Scheduling Using Intelligent Software Agents", International Journal of Technology in Teaching and Learning, pp. 63-78, 2005.

[10] S. Elmohamed, P. Coddington and G. Fox, "A comparison of Annealing Techniques for Academic Course Scheduling", Selected Papers from the 2nd International Conference, PATAT'97, 1998.

[11] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, MA, 1989.

[12] M. Mitchell, J. Holland and S. Forrest, "When will a genetic algorithm outperform hill climbing", In J. Cowan, G. Tesauro and J. Alspector (Eds.), Advances in Neural Information Processing Systems. Morgan Kaufman, 1994.

[13] K. Murray, M. Tomas and H. Rudova, "Modeling and Solution of a Complex University Course Timetabling Problem", lecture notes in computer science, Springer-Verlag GmbH, 2007.

[14] J. Kennedy, R. Eberhart and Y. Shi, "Swarm intelligence", San Francisco, Morgan Kaufmann Publishers, 2001.

[15] <en.wikipedia.org/wiki/np-hard> April, 28, 2009.

[16] A. Eiben and J. Smith, "Introduction to Evolutionary Computing", Springer, Natural Computing Series, 1st edition, 2003.