Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 3, No. 1, April 2017.

DEVELOPING A COURSE TIMETABLE SYSTEM FOR ACADEMIC DEPARTMENTS USING GENETIC ALGORITHM

Mohammad A. Al-Jarrah[1], Ahmad A. Al-Sawalqah[2] and Sami F. Al-Hamdan[3]

(Received: 13-Jun.-2016, Revised: 10-Nov.-2016, Accepted: 15-Jan.-2017)

ABSTRACT

Preparing course timetables for universities is a search problem with many constraints. Exhaustive search techniques in theory can be used to develop course timetables for academic departments, but unfortunately these techniques are computation intensive, since the search space is very large and therefore are impractical. In this paper, Genetic Algorithms (GA's) are utilized to build an automated course timetable system. The system is designed for any academic department. The proposed timetabling system requires minimal effort from the administration staff to prepare the course timetable. Moreover, the prepared course timetable considers faculties' desires, students' needs and available resources, such as classrooms and laboratories with optimal utilization.

The proposed timetabling process was divided into three stages. The first stage is the data collection stage. In this stage, the administrative staff; usually the head of the department, is responsible for preparing the required data, such as the names of the faculty personnel and their desires of courses and laboratories ordered with some priority scheme. Number and type of theoretical and practical courses are also fed to the system based on some statistics about student numbers and previous course timetable history. The system is also fed with number of lecture rooms allocated for the department and number of labs with information about theoretical courses they are able to serve. In the second stage, the program generates an initial set of suggested schedules (chromosomes). Each chromosome represents a solution to the problem, but usually is not satisfactory. Finally, the proposed timetabling system starts the search for a good solution that satisfies best interests of the department according to a cost function. GA is applied in search for a satisfactory course timetable based on a pre-defined criterion. The system has been developed and tested utilizing benchmarked datasets developed by an international timetabling competition (ITC2007) and for the Computer Engineering Department at Yarmouk University. In both cases, the algorithm showed very satisfactory results.

KEYWORDS

Courses timetable problem, Courses timetable generation, Genetic algorithm, Chromosome generation, Parents selection algorithm, Crossover, Mutation.

1. INTRODUCTION

The head of an academic department in any university is usually responsible for preparing a course timetable every semester. Preparing course timetables is a time-consuming task which academic colleges face. Course timetabling is not only formulating a timetable for courses, but also has to be performed based on many constraints, such as classroom availability and capacity, interference between rooms and courses and conflicts between courses and instructors. Very early in 1999, class timetabling at Sirindhon International Institute of Technology has been

1. M. Al-Jarrah is with the Computer Engineering Department, Yarmouk University, Irbid, Jordan. E-mail: jarrah@yu.edu.jo.
2. A. Al-Sawalqah is with the Communications & Electronics & Computer Engineering Department, Tafila Technical University, Tafila, Jordan. E-mail: ahmadsw@ttu.edu.jo.
3. S. Al-Hamdan is with the Computer Engineering Department, Yarmouk University, Irbid, Jordan. E-mail: shamdan@yu.edu.jo.

tackled based on the mentioned constraints. A cost function was defined for each one to utilize the existing facilities and resources effectively [1].

Generally, course timetabling in many universities is prepared manually based on administration experience. The administrator should consider all available facilities and resources, such as courses, instructors, rooms and laboratories. Moreover, the instructors' time and time of course sections are important constraints to handle. Therefore, based on all the mentioned constraints, course timetabling is a very exhaustive and time-consuming task.

Course timetabling is one of the Nondeterministic Polynomial-time (NP) hard problems [1], [12], [23]-[25]. Heuristic search algorithms are usually used in most of these problems to find a near-optimal solution. Nevertheless, this only works for simple cases. For more complex inputs and requirements, obtaining a considerably good solution can take long time and effort [2]-[3], [26]-[27].

Genetic algorithms (GA's) are a class of non-traditional techniques which can handle complex and large search spaces to a certain extent in problem solving [4]-[6], [14]-[16]. In some GA approaches, the search space is divided into subspaces to increase the probability of having good chromosomes in the initial population. This technique increases the chances of finding a near-optimal solution resulting usually in a reduction of computation time [17]-[20]. In the following sections, we propose an automated course timetabling system using genetic algorithms.

Preparing course timetables for academic departments usually utilizes different approaches, such as tabu search, simulated annealing and fuzzy logic [7]-[8], [21]. A combined approach including GA and Hill Climbing has been also introduced to prepare course timetables for academic departments [9]. Another combined approach has also been presented for solving timetabling problem using honey-bee mating optimization algorithm [10]. In [11], an evolutionary approach to solve university course timetabling problems has been introduced.

Wren [22] defines timetabling as follows: "Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives''. Lü *et al.* [28] developed a timetabling algorithm based on tabu search. Bonutti *et al.* developed a curriculum-based course timetabling for universities. The best known solutions have been discussed and compared [29]. Muller, the winner of the International Timetabling Competition 2007, introduced his algorithm based on hill climbing and great deluge technique.

In the course timetabling problem, an optimal solution is defined as the best utilization of available expertise, resources and time for best delivered services. The best utilization is defined under a set of constraints. The constraints are divided into hard and soft constraints. In this paper, the soft and hard constraints used are those related to time, room, instructors and courses. Also, some of the constraints are defined based on type of room (availability of a data show for example), instructor rank and type of course (theoretical or practical). All these constraints have been considered in the proposed algorithm to develop a satisfactory course timetable with minimal effort. Moreover, the proposed algorithm generates a course timetable that satisfies all hard constraints.

The rest of this paper is organized as follows: In section 2, course timetabling using genetic algorithms is discussed. In section 3, automated timetabling using GA's is described. In Section 4, the implementation and experimental results are illustrated for the proposed course timetabling using GA's. Finally, conclusions are presented in section 5.

2. COURSE TIMETABLING USING GENETIC ALGORITHMS

Course timetable is a table which contains information about courses offered for students to register in. Course timetabling is one of the major time-consuming tasks performed frequently

by heads of academic departments. The inputs for the process for preparing a course timetable can be formulated as follows:

1- Set of courses (*Co*) and number of offered sessions for each course, such that $Co = \{(c_1, s_1, ct1, sz_1), (c_2, s_2, ct2, sz_2) \ldots, (c_n, s_n, ctn, sz_n)\}$, where c_i is the i^{th} course, s_i is the number of sessions for course c_i, ct_i is the course type and sz_i is the number of students. Course type has two values; representing theoretical or practical.

2- Set of instructors (Ins), such that Ins= {(I1, L1), (I2, L2),…..,(Im, Lm)}, where Ii is instructor i and Li is the load of instructor i.

3- Set of lecture rooms and laboratories (R), such that R = {(r1, t1), …. , (rs, ts)}, where ri is room number I and ti is the room type. Room type has two values; representing lecture room or laboratory room.

4- Set of time slots (TS), such that TS= {(d1, s1), ….. (dn,Sn)}, where di is the day group selection and Si is the i^{th} slot of time from the selected day group. In universities, the week days are divided into two groups. For example, the first group includes Sunday, Tuesday and Thursday, whereas the second group includes Monday and Wednesday. Moreover, the first group is divided into time slots, such that each slot is 60 minutes, and the second group is divided into time slots, where the slot is 90 minutes. University departments offer their courses on a daily basis. For example, the course may be offered on Sunday, Tuesday and Thursday from 8:00AM to 9:00AM or on Monday and Wednesday from 8:00AM to 9:30AM for the entire semester.

Based on the inputs, the task is to prepare course timetabling that satisfies all hard constraints and most possible soft constraints. The hard constraints are:

H1- Scheduling all courses listed in *Co* list.
H2- Assigning an instructor to each course and satisfying instructors load according to *Ins* set.
H3- Assigning a lecture room to each course and satisfying room type according to *Co* list, where practical courses must be in a laboratory room.
H4- Sessions of the same course should not be in the same time slot.
H5- The lecture room size must be greater than the number of students in the *Co* set.
H6- An instructor must not have more than one lecture in the same time slot.

The soft constraints are:

S1- Room capacity to expected attending students. To satisfy this constraint, the ratio of the number of students in *Co* list to room size must be more than 0.85.
S2- Room utilization. It is expected to utilize lecture room all the time. To satisfy this constraint, the ratio of the utilization time to the total room time must be more than 0.90.
S3- Instructor preferable courses. This constraint is satisfied if the assigned course to an instructor is one of the preferred courses of this instructor.
S4- Instructor preferable lecturing time slots and not preferable time slots. This constraint is satisfied if the assigned course time to an instructor is one of the preferred time slots of this instructor.
S5- Instructor teaching experience. It is preferred to assign a course to an instructor who taught it. This constraint is satisfied if the course assigned to an instructor has been previously

taught by this instructor.
S6- Instructor timetable compactness. This constraint measures the distribution of instructor timetable. To satisfy this constraint, first, instructor timetable should not include more than two adjacent lectures and the waiting time between lectures should not be more than one time slot.

The process of preparing a course timetable considers the inputs, such that the resulting course timetable must satisfy all hard constraints and all possible soft constraints. Therefore, the process of preparing a course timetable is not a straight forward process and the solution is not unique. Moreover, preparing a course timetable is very costly in terms of time and effort if it's prepared manually. Preparing a course timetable automatically using intelligent algorithms has been investigated as mentioned earlier in the introduction section.

Genetic algorithms are one of the methods that reduce computation cost in comparison with other artificial intelligence approaches, such as heuristic search tree, particularly for hard or complex problems like preparing courses timetables. GA's clone biological evolution and consist of simple steps that go through several iterations (generations) until a satisfactory solution is reached. For course timetabling system, a satisfactory solution is one that satisfies all hard constraints and as much as possible of soft constraints. For example, the load for an instructor of the rank assistant or associated professor should not exceed 12 credit hours, while it is 9 credit hours for a full professor. This is a hard constraint; however, the desire of a certain instructor of having his courses after 10 O'clock is a soft constraint that is good to satisfy, but is not necessary. Assuming that the problem has an optimal solution, achieving 100% satisfactions for all constraints (optimal) is computationally very costly.

In GA's, a solution is represented as a chromosome. Population is a set of chromosomes that represent the solution subspace. New generations of chromosomes are generated through two basic genetic operations: crossover and mutation. In crossover, two genes, where a gene is a part of a chromosome, containing a part of the solution, are swapped between the mated pair of chromosomes. The new chromosome would be considered as an accepted, not accepted or satisfactory solution [9]. In mutation, a gene is completely changed randomly for the selected chromosomes. Crossover occurs much more frequently than mutation. GA's assume that the new generations of chromosomes will have better qualities than their ancestors. The process of crossover and mutation is repeated until a near-optimal solution for the problem in hand is reached [10].

GA's have many features that make them in some occasions the best problem solving method. GA's can be used in problems where the search space is usually large and complex when traditional methods need long time and high computation systems [13]. Furthermore, GA's are considered as intelligent systems, where the knowledge is built based on last best knowledge until a satisfactory solution is reached. This means that GA's do not require generating the full search space.

GA's have a parallel nature in which parallelism can be explored and used making it possible to speed up the calculations. To identify a near-optimal solution, each chromosome in the population has a fitness value. The fitness value for a chromosome defines how much it is close to the optimal solution. Genetic processing is applied to improve the fitness value of new generations of chromosomes. If near-optimal solution is not found after a reasonable number of search iterations, then the process is started all over again with a new set of chromosomes. Hopefully, the new running chromosomes will lead to the required near-optimal solution.

3. AUTOMATED COURSE TIMETABLING SYSTEM USING GENETIC ALGORITHM

The proposed algorithm for course timetabling utilizes genetic algorithms to generate a near-optimal course timetabling in a relatively short period of time and with a little effort from the administrator side. Figure 1 illustrates the proposed algorithm for course timetabling system utilizing GA's. The final schedule is achieved if the solution satisfies all predefined hard constraints. The main loop of search includes genetic operations, which include mutation and crossover.

According to Figure 1, the timetabling is divided into two stages. The first stage includes chromosome generation and population initialization. This stage starts after collecting data and constraints from administrators who are responsible to prepare the course timetable. The second stage includes the process of applying genetic operations, which are crossover and mutation of selected chromosomes. Each chromosome represents a solution to the problem (course timetable), but usually is not optimal. In this stage, we select chromosomes called parents to apply genetic operations. Operations will generate new chromosomes that have properties better than their parents. The generated chromosomes will be added to the population. The new population is evaluated to check if one of the chromosomes achieved the targeted goal.

3.1 Chromosome Generation and Population Initialization

In this subsection, we discuss the first stage that handles the generation of chromosomes and adding them to the population. The generated chromosomes are generated randomly. Each gene is selected randomly from the gene pool. The following steps explain how to generate a chromosome.

a. The course sessions that must be offered by the department are determined by the department administration and considered as an input list for the population initialization process. The set of courses is included in the Co set.
b. The chromosome genes are course genes, instructors' genes, days and time of lectures and lecture room or laboratory room.
c. Select session gene from the input list randomly and assign this value to a chromosome; if the input list is empty, go to step h.
d. Complete the remaining genes for this chromosome part. For example, select randomly an instructor from the instructor pool, a time from the time pool and a lecture room from the halls' pool. The process is accomplished for every gene in the chromosome.
e. For the generated chromosome part, check if the course is a laboratory one, then reselect the hall from the laboratory halls' pool.
f. After considering all items of the input list, the obtained chromosome represents an initial solution. Add this chromosome to the initial population and reload the input list.
g. Repeat steps c to f *N* times, where *N* is the number of chromosome in the population.
h. End of population initialization.

The result of population initialization is a set of solutions. The near-optimal solution which we are searching for is most probably not one of the initial generations. Therefore, the fitness value is computed using the proposed fitness function for each chromosome generated during the population initialization process.

3.2 Fitness Functions

The fitness function is computed and evaluated for every chromosome in the population. The fitness is measured through quantifying constraints. The constraints are divided into two groups; hard constraints and soft constraints. A hard constraint is a constraint that makes the course timetable invalid if not satisfied by the solution. For example, an instructor is assigned to teach two courses at the same time. Thus, for a valid course timetable, all hard constraints must be achieved. A soft constraint is one that is strongly recommended to satisfy, but does not make the solution invalid if not satisfied. For example, some instructors prefer to have their lectures early, while others prefer to have them at late times. A satisfied constraint is a constraint that will have a positive value (+2) on the fitness function and an unsatisfied one will have a negative value (-2). However, a satisfied soft constraint will have also a positive contribution (+1) on the fitness, while an unsatisfied one will have a zero effect on the fitness function (see Equations 1 and 2).

For hard constraints:

$$HC_i = \begin{cases} 0. \ if\ hard\ constraint\ i\ is\ accheived \\ 2. \ if\ hard\ onstraint\ i\ is\ not\ accheived \end{cases} \quad (1)$$

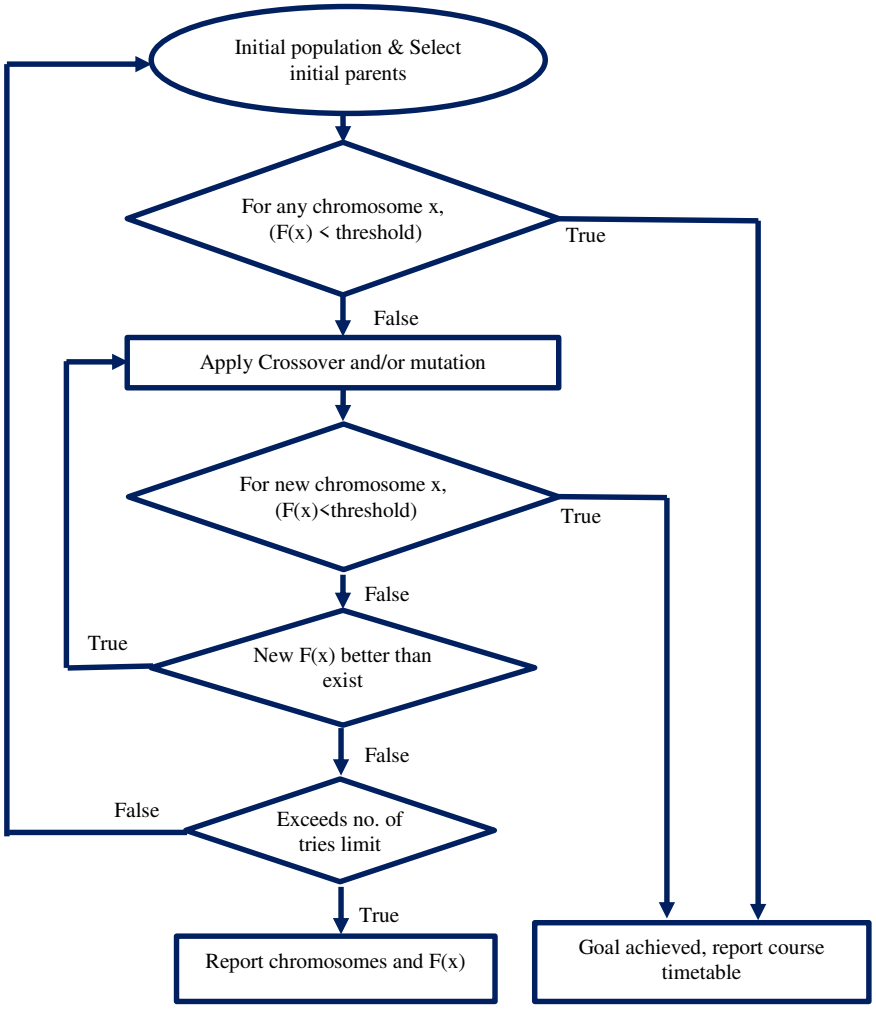

Figure 1. The proposed system for preparing a course timetable using GA's.

For soft constraints, (SCi) is calculated as follows:

$$SC_i = \begin{cases} 0. \ if\ soft\ constraint\ \ i\ is\ accheived \\ 1. \ if\ soft\ constraint\ \ i\ is\ not\ accheived \end{cases} \quad (2)$$

The fitness function will include all constraints. Thus, the fitness function is computed for a certain chromosome *x* as follows:

$$F(x) = \sum_{i=1}^{L} HC_i + \sum_{i=1}^{M} SC_i \quad (3)$$

where *L* is the number of hard constraints and *M* is the number of soft constraints. This fitness function ranges from 0 to (2N+M). According to Equation (3), the chromosome with a fitness value equal to zero is a chromosome that stratified all constraints. Thus, the ultimate goal for our algorithm is to obtain a chromosome with a fitness value equal to zero. However, finding

such a chromosome is not guaranteed by GA's and if possible will usually be computationally very costly. Thus, we defined the following condition, which we named *GOAL* in the proposed algorithm, to stop the search loop as follows:

$$GOAL = \begin{cases} True. \quad for\ any\ x. if\ F(x) < threshold. and\ HC(x) = 2L \\ False. \quad otherwize \end{cases} \tag{4}$$

The value of $HC(x) = 2L$ if all constraints are achieved.

3.3 Chromosome Selection

After *N* chromosomes are added to the initial population, the fitness function is calculated for each candidate in the population. Most likely, there is no chromosome in the initial population which satisfies the *GOAL*; therefore, a process of genetic operations will be applied on the set of solutions called parents. Two chromosomes will be mated together to produce new chromosomes. The selection of these two chromosomes is dynamic and depends on the fitness value for each chromosome. The probability of selecting a chromosome to be a parent depends on its fitness value. The probability value for selecting a chromosome (x) is defined as:

$$p(x) = 1 - \frac{F(x)}{\sum_{i=1}^{N} F(i)} \tag{5}$$

where *N* is the number of chromosomes in the population and $F(x)$ is the fitness of the chromosome. According to Equation 5, the chromosome with a higher fitness value has a better chance to be chosen for mating operations.

After selecting parents, the genetic operators are applied. From each operation, new chromosomes are generated and the fitness values for these chromosomes are computed for the next iteration.

3.4 Applying Genetic Algorithms Operation (Mutation and Crossover)

Genetic algorithms have two types of operation: crossover (mating) and mutation. Chromosome mating operation is usually accomplished utilizing a pair of chromosomes that are selected as parents by chromosome selection process. The crossover operation includes the exchanges of instructor, classroom, course or time genes in a random manner. The result of the mating operation is two new chromosomes that would have the good properties of their parents. The fitness function is computed for the new chromosomes. If the fitness values for the new chromosomes are greater than those of their parents, these new chromosomes are added to the population. Otherwise, they are discarded and the process is repeated.

The mutation is the process of changing a gene in parent chromosome with a new gene. In the proposed algorithm, the mutation is accomplished as follows:

1- Select a chromosome from the population as a parent one.
2- Randomly, select which gene is to be replaced. The targeted genes are course, instructor, classroom or time slot.
3- Select randomly a new gene from the gene pool. The selected gene type must be similar to that of the one selected in step 2. For example, if we decided to replace an instructor gene, then the pool will be the instructors' pool in the department.
4- Replace the new gene with the old one.
5- Calculate the fitness function for the new chromosome. If the fitness value for the new one is better than that of the parent, then add the new chromosome to the population. Otherwise, discard the new chromosome and repeat the process.

The mutation process is applied every while or when crossover operation does not achieve an improvement. Thus, the probability of applying mutation is increased with respect to number of

unsuccessful crossover operations and with respect to the difference between the best fitness values for the population before crossover and the fitness values for the population after crossover operation. The implementation of randomly applying mutation is achieved through defining a mutation counter. The value of the counter is initially zero and every time the crossover failed to improve the fitness value, this counter is incremented by one. Then, at the end of every loop, we randomly generate an integer between 0 and 10000. If the generated number is less than the mutation counter, we apply the mutation process and reset the mutation counter.

4. PROPOSED ALGORITHM IMPLEMENTATION AND RESULTS

The proposed algorithm for developing course timetables for academic departments using genetic algorithms is implemented using Microsoft Visual Studio. We utilized C# language to develop the application which generates the course timetable. The users for this application include the head of the department and the department instructors. The head of the department (HOD) interacts with the application using a Windows Application. HOD inserts the list of courses and their attributes, list of instructors, as well as list of resources such as lecture rooms and laboratories, list of hard constraints and list of soft constraints. Then, instructors interact with the application through a web application. The instructors insert their soft constraints. Finally, the HOD approves the constraints of the instructors and initiates the process of generating the course timetable.

The developed application was tested and verified for the Department of Computer Engineering at Yarmouk University. HOD inserted lists of 45 courses, 13 laboratories and number of section to be offered for every course and laboratory. Then, every instructor inserted preferred courses as well as preferred and not preferred time slots. Also, each instructor inserted a set of previously taught courses.

The second phase of generating the course timetable was started through initiating the second stage of the algorithm. The system generates 30 chromosomes. It took 33 seconds to generate the initial population. This experiment was conducted utilizing Intel Pentium i3 CPU and 4 GB memory. The next step was applying genetic operations according to the proposed algorithm. Table 1 shows a sample output of applying mutation and crossover operations. The first row of Table 1 shows that the processed crossover operation at iteration 10079 has two parents with 18 and 35 fitness values. The new chromosome has a fitness value of 42, which is better than the parents' value. Thus, this chromosome is added to the population. The second row of Table 1 shows the process of crossover at iteration number 10210. The new chromosome has a fitness value equal to 22, which is less than its parents' values. Thus, the new chromosome is ignored and the process is restarted. The mutation operation is applied in iteration number 1209. The new chromosome fitness value is equal to 41, which is better than its parents' fitness values. Therefore, this chromosome is added to the population.

The implemented algorithm generated a course timetable for the Department of Computer Engineering at Yarmouk University. The best course timetable was generated after more than 32000 iterations. The generated course timetable was distributed to the instructors and they were asked to evaluate it. The evaluation was 9.56 out of 10. This evaluation shows a high degree of satisfaction among the instructors and the department administration. For the purpose of validating the proposed system with other timetabling approaches, we modified our system to read the input from text file. The file contains a dataset for curriculum-based course timetabling prepared by international timetabling competition [29]-[30]. The dataset has information about offered courses, rooms, curricula and unavailability constraints. Moreover, we have modified the scheduling process in our proposed system to consider courses have minimum working days. In our original proposed system, all offered courses span over one semester. Moreover, in the original proposed system, the instructor can be selected from a pool. Thus, the preference of

lecturers to teach a course is considered as soft constraint, while the selection of a lecturer from the pool is a hard constraint.

Table 1. Sample output for crossover and mutation operation.

GA's Operation	Chromosome Situation	Operation Fit Value
Crossover	Chromosome accepted	Iteration = 10079 F(1)= 18, F(11)= 35 F(Pc)= 42
Crossover	Chromosome not accepted	Iteration = 10210 F(1)= 17 , F(11)= 26 F(Pc)= 22
Mutation	Chromosome accepted	Iteration = 1209 F(8)= 24 , F(Pc) = 41

Table 2. Obtained minimum number of satisfied soft constraints of the proposed system for five datasets and comparison with other algorithms.

Dataset	Proposed system	Lü *et al.* [28]	Bonutti *et al.* [29]	Muller [30]
comp01	4	4	5	5
comp02	20	22	75	43
comp03	34	41	93	72
comp04	21	19	45	35
comp05	203	224	326	298
comp06	18	25	62	41
comp07	6	4	38	14

After the modification of the proposed system to comply with all requirements of the international timetabling competition, we downloaded 5 datasets from the international timetabling competition website and computed the number of violated hard constraints and soft constraints for each dataset. The scheduling process is stopped based on timeout condition according to ITC-2007 competition [30]. Because the scheduling algorithm is stopped based on timeout condition, the process may have different results for different runs. Thus, we applied the proposed algorithm fifty times and considered the best results. Table 2 summarizes the obtained results and the results of the other scheduling algorithms that utilized the same dataset. According to Table 2, the proposed algorithm outperforms other approaches on most listed datasets.

5. CONCLUSIONS

In this paper, we proposed a new approach to generate course timetables for academic departments at universities. The proposed approach utilizes Genetic Algorithms to develop course timetables. The proposed approach is divided into three stages. The first stage is inputting the course list and number of sections in each one, instructors and their constraints, rooms available, hard and soft constraints for rooms and lab resources. The second stage generates a set of initial solutions (chromosomes) randomly called initial population. In the third stage, the initial solutions are subjected to GA operations (crossover and mutation) repeatedly

until a near-optimal solution is reached that achieves a predefined (threshold) fitness function. The closer to zero, the better is the solution.

The fitness function was computed for each solution. The value of the fitness function indicates the goodness of the solution. After that, genetic operators which are crossover and mutation, were applied. The goal of applying these operators is to obtain new solutions with better fitness values. As a feature in genetic algorithm, crossover is applied more frequently than mutation. Applying genetic operators was repeated until we obtained a satisfying solution according to an instructors' satisfaction questionnaire. The proposed algorithm was implemented and utilized to generate a course timetable for the Department of Computer Engineering at Yarmouk University. An evaluation process of the generated course timetable among instructors was found to be very satisfactory. Moreover, we modified our code to read inputs from text file formatted according to benchmarked datasets. Then, we tested the proposed algorithm utilizing seven benchmarked datasets. The results of our algorithm outperformed other methods in most datasets.

REFERENCES

[1] J. Nakasuwan, P. Srithip and S. O. Komolavanij, "Class Timetabling Optimization," Thammasat International Journal of Science and Technology, vol. 4, no. 2, pp. 88-98, 1999.

[2] P. Chang, S. Chen and K. Lin, "Two-Phase Sub-Population Genetic Algorithm for Parallel Machine-timetabling Problem," Expert Systems with Applications, vol. 29, no. 3, pp. 705-712, 2005.

[3] H. Babaei, J. Karimpour and A. Hadidi, "A Survey of Approaches for University Course Timetabling Problem," Journal of Computers & Industrial Engineering, vol. 86, pp. 43-59, 2015.

[4] P. Guo, J. Chen and L. Zhu, "The Design and Implementation of Timetable System Based on GA's," IEEE International Conference on Mechatronics Science, Electric Engineering and Computer (MEC), pp. 1497-1500, 2011.

[5] A. George, T. Marwala and F. Nelwamondo, "Use of Data Mining in Scheduler Optimization," arXiv preprint arXiv:1011.1735, 2010.

[6] K. Deb, "Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems," Evolutionary Computation, vol. 7, no. 3, pp. 205-230, 1999.

[7] C. Teoh, A. Wibowo and M. Ngadiman, "Review of State of the Art for Metaheuristic Techniques in Academic Timetabling Problems," Artificial Intelligence Review, vol. 44, no. 1, pp. 1-21, 2015.

[8] S. Mir Hassani and F. Habibi, "Solution Approaches to the Course Timetabling Problem," Artificial Intelligence Review, vol. 39, no. 2, pp. 133-149, 2015.

[9] M. El-Sherbiny, R. Zeineldin and A. El-Dhshan, "Genetic Algorithm for Solving Course Timetable Problems," International Journal of Computer Applications, vol. 124, no. 10, 2015.

[10] N. Sabar, M. Ayob, G. Kendall and R. Qu, "A Honey-bee Mating Optimization Algorithm for Educational Timetabling Problems," European Journal of Operational Research, vol. 216, no. 3, pp. 533-554, 2012.

[11] J. Obit, D. Ouelhadj, D. Landa-Silva and R. Alfred, "An Evolutionary Non-Linear Great Deluge Approach for Solving Course Timetabling Problems," IJCSI International Journal of Computer Science Issues, vol. 9, no. 4, pp. 1-13, 2012.

[12] V. Sapru, K. Reddy and B. Sivaselvan, "Timetabling Using Genetic Algorithms Employing Guided Mutation," IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1-4, 2010.

[13] A. Page and J. Naughton, "Dynamic Task Timetabling Using Genetic Algorithms for Heterogeneous Distributed Computing," Proceedings of 19th IEEE International Symposium in Parallel and Distributed Processing, pp. 189a-189a, April 2005.

[14] S. Wu, H. Yu, S. Jin, K. Lin and G. Schiavone, "An Incremental Genetic Algorithm Approach to Multiprocessor Timetabling," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 9, pp. 824-834, 2004.

[15] E. Vallada and R. Ruiz, "A Genetic Algorithm for the Unrelated Parallel Machine Timetabling Problem with Sequence-Dependent Setup Times," European Journal of Operational Research, vol. 3, pp. 612-622, 2011.

[16] T. Dereli and H. Filiz, "Allocating Optimal Index Positions on Tool Magazines Using Genetic Algorithms," Robotics and Autonomous Systems, vol. 33, no. 2, pp. 155-167, 2000.

[17] J. Chen, D. Goldberg, S. Ho and K. Sastry, "Fitness Inheritance in Multi-Objective Optimization," Proceedings of the Genetic and Evolutionary Computation Conference, pp. 319-326, 2002.

[18] F. Pezzella, G. Morganti and G. Ciaschetti, "A Genetic Algorithm for the Flexible Job-shop Timetabling Problem," Computers & Operations Research, vol. 35, no. 10, pp. 3202-3212, 2008.

[19] M. Carvalho, A. Laender, M. Gonçalves and A. Da Silva, "A Genetic Programming Approach to Record Deduplication," IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 3, pp. 399-412, 2012.

[20] G. Kim and C. Lee, "Genetic Reinforcement Learning Approach to the Machine Timetabling Problem," Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 196-201, 1995.

[21] M. Ghaith and A. Masri, "Scatter Search for Solving the Course Timetabling Problem," 3rd IEEE Conference on Data Mining and Optimization (DMO), Selangor, Malaysia, 28-29 June 2011.

[22] A. Wren, "Scheduling, Timetabling and Rostering – A Special Relationship?," International Conference on the Practice and Theory of Automated Timetabling, pp. 46-75, 1996.

[23] A. Bettinelli, V. Cacchiani, R. Roberti and P. Toth, "An Overview of Curriculum-based Course Timetabling," TOP, vol. 23, no. 2, pp. 313-349, 2015.

[24] V. Cacchiani, A. Caprara, R. Roberti and P. Toth, "A New Lower Bound for Curriculum-based Course Timetabling," Computers & Operations Research, vol. 40, no. 10, pp. 2466-2477, 2013.

[25] E. Özcan, A. Parkes and A. Alkan, "The Interleaved Constructive Memetic Algorithm and Its Application to Timetabling," Computers & Operations Research, vol. 39, no. 10, pp. 2310-2322, 2012.

[26] H. Rudová, T. Müller and K. Murray, "Complex University Course Timetabling," Journal of Scheduling, vol. 14, no. 2, pp. 187-207, 2011.

[27] S. Jat and S. Yang, "A Hybrid Genetic Algorithm and Tabu Search Approach for Post Enrolment Course Timetabling," Journal of Scheduling, vol. 14, no. 6, pp. 617-637, 2011.

[28] Z. Lü and J. Hao, "Solving the Course Timetabling Problem with a Hybrid Heuristic Algorithm," International Conference on Artificial Intelligence: Methodology, Systems and Applications, pp. 262-273, 2008.

[29] A. Bonutti, F. De Cesco, L. Di Gaspero and A. Schaerf, "Benchmarking Curriculum-based Course Timetabling: Formulations, Data Formats, Instances, Validation, Visualization and Results," Annals of Operations Research, vol. 194, no. 1, pp. 59-70, 2012.

[30] T. Muller, ITC2007: Solver Description, Technical Report, Purdue University, 2008.

"Developing a Course Timetable System for Academic Departments Using Genetic Algorithm", Mohammad A. Al-Jarrah, Ahmad A. Al-Sawalqah and Sami F. Al-Hamdan.

ملخص البحث:

إنّ إعداد جداول المحاضرات في الجامعات مسألة بحثية تنطوي على العديد من المحدّدات. وهناك تقنيات يمكن استخدامها من الناحية النظرية من أجل إعداد جداول المحاضرات الخاصة بالأقسام الأكاديمية، إلا أنها لسوء الحظ تحتاج الى الكثير من الحسابات لأن حيز البحث واسع جداً، وهذا يجعل تلك التقنيات غير عملية. في هذه الورقة، يتم استخدام الخوارزميات الوراثية لبناء نظام آلي لإعداد جداول المحاضرات. وهذا النظام مصمّم بحيث يُلائم أيّ قسم أكاديمي. ولا يتطلب النظام المقترح سوى القليل من الجهد من جانب الطاقم الإداري لإعداد الجدول المطلوب. من جهة أخرى، يراعي الجدول المعدّ بهذه الطريقة رغبات أعضاء هيئة التدريس واحتياجات الطلبة والموارد المتاحة مثل الغرف الصفية والمختبرات بفاعلية مثالية.

وقد تم تطوير النظام واختباره باستخدام مجموعات بيانات مرجعية عالمية لإعداد جداول المحاضرات، وطُبّق النظام على قسم هندسة الحاسوب في جامعة اليرموك. وقد أسفر اختبار النظام وتطبيقه عن نتائج مُرضية جداً.