SOLVING WEEKLY COURSE TIMETABLING PROBLEM WITH GENETIC ALGORITHM AND LOCAL SEARCH

Ozcan Dulger
Department of Computer Engineering, Middle East Technical University, Ankara, Turkey
{odulger@ceng.metu.edu.tr}

ABSTRACT

In this paper, generating weekly course timetable for a university department is studied. In a course timetabling problem, there are time slots in a week and the courses are assigned to these time slots by considering the conflictions about the classrooms, instructors, students. In this problem, there are some constraints which are grouped as hard constraints and soft constraints. The hard constraints should be satisfied for an optimal solution. But the soft constraints cannot be always satisfied because of the limited resources, but it is good if they can be satisfied as much as possible. In this study, genetic algorithm is used to solve this problem. The representation, fitness functions are explained in detail. Although genetic algorithms are well-suited for the global exploration, they usually fail in local exploitation. Because of this situation, some hybridization methods are designed to improve the genetic algorithm's performance. And also a multi objective fitness function is designed for the two important rules in the department. These methods are tested in computer environment with real computer engineering department's data of a university. The experiment results show that this improvement increases the genetic algorithm's performance incredibly.

KEYWORDS

Weekly course timetable; Local search; Hybrid genetic algorithm.

1 INTRODUCTION

Scheduling by human hand is difficult because of its constraints and complex structure. In general, scheduling is an assignment of the events to the time slots optimally. Earlier the mathematical methods were used to solve these problems but they failed because of the complex structure of scheduling. Later, heuristics methods have been used such as simulated annealing, tabu search, honey bee optimization because to find an optimal solution which satisfies the whole constraints requires logical reasoning and requires searching possible solutions with a well-determined heuristic function. Abramsom used simulated annealing method in his timetable study and then compared his study with other six methods (Abramson, 1991, Abramson, Dang and Krisnamoorthy, 1999). Schaerf tried to solve scheduling problems with tabu search method (Schaerf, 1996). Nowadays, genetic algorithm (GA), which is fast and prefers different candidate solutions, has been chosen because of its four main characteristics (Goldberg, 1989):

- GA uses coding of parameters instead of their real values.
- GA evaluates a lot of candidate solutions instead of one solution. By this, it can avoid from the local optimum.
- To obtain optimal solution GA uses independent fitness function. It doesn't need any derivative or similar information. So it can be used in a lot of problems.
- GA shows different approach based on probabilities at every time. So it obtains results that the other techniques cannot achieve.

In this paper, generating weekly course timetable of a university department is studied. This problem is solved with genetic algorithm. In Section 3, the problem definition and methods are explained in detail. Later, in Section 4, the proposed local search methods, which are incorporated with genetic algorithm to improve the genetic algorithm's performance, are given. In Section 5, a multi objective fitness function for unsolvable constraints because of limited resources is proposed. In Section 6, experiment results are shown. Finally, in Section 7, the comments about the study are given.

2 RELATED WORK

One of the problems of scheduling is weekly course timetabling problem. In course timetabling problem, the courses are assigned to the time slots by taking consideration to the constraints. Genetic algorithm first was used by Colorni in course timetabling problems (Colorni, Dorigo and Maniezzo, 1992). In his study the solution is represented by matrix, where the rows are instructor's information and the columns are lecture hours. He obtains better results against the other heuristic methods. Yu and Sung also used sector based GA in their university course timetabling problem (Yu and Sung, 2002). In their study, the rows of solutions

represent the whole classrooms and the columns represent all the time slots from Monday to Friday. Özcan and Alkan proposed steady state GA for a university course timetabling problem (Ozcan and Alkan, 2002). In their representation, each gene has two values which are the lecture day of the course and the lecture hour of the course in that day. Yiğit solved high school course timetable with GA (Yigit, 2006). The genes represent the time slots and their values are the code of instructors, courses and classrooms. Beligiannis and his friends proposed an adaptive algorithm based on evolutionary computation (Beligiannis, Moschopoulos, Kaperonis and Likothanassis, 2008). In their study, solutions are represented by matrix where the rows are whole classrooms, columns are all the time periods in the week and the cells are instructor information.

3 PROBLEM DEFINITION AND METHODS

In this section, firstly problem definition of weekly course timetable of a university department is given. Then the representation and the other methods of genetic algorithm are explained in detail. There is some information about course timetable which are given below:

- There are classrooms for lectures.
- There are instructors for each course.
- There are 6 × 8 time slots in a week. Six days and eight lecture hours between 8.00 – 17.00 (except lunch time) in each day.
- There are students in department. These are first year students, second year students, third year students and fourth year students.
- Some classrooms have materials such as projector, computers. And some courses need these specific classrooms.

The course timetable that uses this information should satisfy some constraints. These constraints are divided into two categories. The first one is the hard constraints. The second one is the soft constraints. In hard constraints, all the constraints should be satisfied. But in soft constraints, all the constraints cannot be satisfied because of limited resources but it is good if they can be satisfied as much as possible. The hard constraints are given below:

- There shouldn't be more than one lecture in a classroom at the same time.
- The students shouldn't have more than one lecture at the same time.
- The instructors shouldn't have more than one lecture at the same time.
- There shouldn't be any lecture in lunch time (12.00 – 13.00).
- Block lectures shouldn't be separated by lunch time or following day.
- There shouldn't be any lecture on Saturday.
- The courses should be in a correct classroom. For example, lab course needs a classroom that has computers.
- Common courses which are given in all the departments such as history should be in a correct time slots.

The soft constraints are given below:

- The preferences of instructors for lecture times in the week.
- The preferences about student. For example, there is no any lecture on Monday for the first year students.
- The first year and second year courses shouldn't conflict.
- The second year and third year courses shouldn't conflict.

To solve this problem genetic algorithm is used. There are some operations which are based on genetic structure. The solution should be represented in chromosomes, so the phenotype of the candidate solutions should be converted to genotype. Then these individuals reproduce at each generation by crossover operation. At the end individuals that survive for the next generation represent the candidate solutions. The best individual that fits to excepted solution is chosen. These operations are explained below.

3.1 Representation

Deciding the chromosome structure is important for this problem. So this operation should be performed carefully to obtain optimal solutions. If it is considered that no more than one lecture in a classroom at the same time, the gene values in the chromosome can be permutation. In this problem, the count of genes in a chromosome is equal to (total course count × maximum block lecture hour). For example, assume that there are 30 courses and the maximum block lecture hour of a course is 3 so, the gene count becomes 30 × 3 = 90. The first three genes represent the first course in database. And these genes represent

the first lecture hour, second lecture hour and third lecture hour of that course respectively. The next three genes represent the second course and so on. There are 48 lecture hours in a week as it is given before. The genes values are between 0 and (48 × classroom count), so the first hard constraint which is no more than one lecture in a classroom at the same time is satisfied at the beginning (Karaboga, 2004). The chromosome structure is shown in Figure 1.

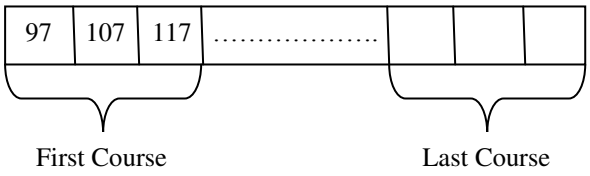

Figure 1: The structure of chromosome.

This chromosome represents the weekly course timetable. In here the first gene is equal to the first lecture hour information of the first course in database. The second gene is the second lecture hour information of the first course and the third gene represents the third lecture hour information of the first course. Assume there are ten classrooms in a department, for the first gene 97 ÷ 10 = 9 and 97 % 10 = 7 values are obtained from value 97. In here value 9 is equal to 10th lecture hour of the week, in other words Tuesday at 9.00 am, value 7 is equal to 8th classroom in database.

3.2 Fitness Function

After deciding the chromosome structure, the suitable fitness function should be determined. In this problem, the constraints are tried to be satisfied, so a penalty cost is given for an unsatisfied constraint depending of its priority. The aim to solve this problem is minimizing the fitness value. The penalty costs for each constraint are given below:

- If an instructor or students have more than one lecture at the same time, the penalty cost is 10000.
- If a lecture is separated by lunch time or following day, the penalty cost is 5000.
- If a lecture is on Saturday, the penalty cost is 3000.
- If a course isn't in a correct classroom, the penalty cost is 2103.
- If an instructor's preferences aren't satisfied, the penalty cost is 500.
- If student's preferences aren't satisfied, the penalty cost is 10.

In this fitness function, the hard constraints have higher penalty cost value. So the chromosome that has higher fitness function value, have a less chance to survive for the following generations. The soft constraints have lower penalty cost value. Because they cannot always be satisfied, their effects to the fitness function are a little. One can predict the unsatisfied constraints at each generation by looking to fitness value of a chromosome. The 2103 value is given to the fourth constraint above in order to make this prediction clearly.

3.3 Other Operators

To select parents roulette wheel algorithm is used (Eiben and Smith, 2007). In this algorithm individuals have a chance proportional to their fitness values. The better individuals have much chance to be selected.

In crossover operation, the parents which are selected in selection operation reproduce. For this operation, order crossover is used (Davis, 1985).

In mutation operation, the child's genes are mutated by swapping the values randomly in the selected substring (Eiben and Smith, 2007). Also second mutation operator is used. In this operator the selected gene value is replaced with a new value.

In survival operation, the individuals are selected for the next generation. The new generation consists of parents and children. Elitism is used to survive some of the better parents and remaining consists of children (Eiben and Smith, 2007).

Finally, in termination the algorithm is finished when the optimal solution is obtained or when maximum generation count is reached or when the user performs the stop operation.

4 IMPROVING GENETIC ALGORITHM WITH PROPOSED LOCAL SEARCH METHODS

In this section we study how the performance of the method, which is explained in Section 3, is improved by local search methods. Because of the complex and combinatorial structure of timetabling problems, finding optimal solution is very hard. Although genetic algorithms are well suited to perform global exploration, they usually fail in local exploitation. Various hybridization approaches have been suggested to solve this problem. Cheng and his friends give a detail survey of hybrid genetic algorithm approaches in their study. They especially mention to local search approaches which are incorporated with genetic

algorithms as an add-on extra to the main loop of recombination and selection. With these approaches, genetic algorithms perform global exploration among the population, while heuristic methods perform local exploitation around chromosomes. They also mention that in genetic algorithm the selection of chromosomes is based on fitness at their birth, however with hybrid methods the selection is based on fitness after local search and this hybrid approaches can be viewed as the combination of Darwin's evolution with Lamarck's evolution (Cheng, Gen and Tsujimura, 1999).

Also there have been other studies in the literature about this approach. Renders and Bersini used hill climbing search method to hybridize the genetic algorithm (Renders and Bersini, 1994). Ishibuchi and Murata proposed a local search algorithm for a flow-shop scheduling (Ishibuchi and Murata, 1998). In their purpose, the current solution's neighbours are examined. If the neighbours' solution is better than current solution, these are replaced with each other. This operation has been done until all the neighbours of current solution are examined. Abdullah and Turabieh proposed a local search method for a course timetabling problem (Abdullah and Turabieh, 2008). In their study, they assign time slots to each event in order until an improvement occurs for that event. This is done for all the events in that chromosome.

Now the proposed local search methods which hybridize the method in Section 3 are explained in detail. Two different local search methods are determined for hard constraints and soft constraints. These methods are performed after crossover and mutation operations. In the first method, the hard constraints are examined and are tried to be satisfied if it is possible. This method is explained in detail below:

- Consider the whole genes of the chromosome one by one.
- If a hard constraint occurs in a gene, assign whole available values one by one to that gene.
- If an improvement is obtained, break the assignment and go to the next gene.
- Do these operations until all the genes are examined.
- After above operations, if the current chromosome is better than the initial chromosome, replace them.

In the second method, the soft constraints are examined. Unlike the first method, the second method isn't too strict because the soft constraints don't have to be satisfied at each time. The second method is explained below:

- Consider the whole genes of the chromosome one by one.
- If a soft constraint occurs in a gene, assign an available value randomly to that gene.
- Do these operations until all the genes are examined.
- After above operations, if the current chromosome is better than the initial chromosome, replace them.

Apart from these, a repair function is used to improve the chromosomes. In this function, if a block lectures are separated by lunch time or following day, it is corrected by shifting. In this operation, if the earlier time slot is available, block lectures are shifted to that time slot.

Furthermore, a neighbourhood search-based mutation is used. Various examples are shown in literature (Cheng, Gen and Tsujimura, 1999). In this method, the chromosome is mutated more than once under the same circumstances and the best of the solutions is chosen.

5 A MULTI-OBJECTIVE FITNESS FUNCTION FOR THE TWO RULES

Generally, in a multi objective optimization (MOO) more than one fitness functions are described for multi criteria of the problem. These fitness functions are performed simultaneously and the non-dominated solutions are chosen for the result. There are some studies about MOO. Mumford proposed two different fitness functions in his timetabling study (Mumford, 2007). The first fitness function minimizes the overall length of the examination period and the second fitness function minimizes the total proximity cost. Burke and Petrovic also proposed multi criteria objective fitness function for a timetabling problem (Burke and Petrovic, 2002).

In our problem, the department has two important rules. The first rule is the students cannot take any course from third year courses unless they pass the whole courses from the first year. The second rule is the students cannot take any courses from fourth year courses unless they pass the whole courses from second year. So a second year student can take any course from first year again, and similarly third year student can take any course from second year again. Because of this, first year courses shouldn't conflict

with second year courses as much as possible and second year courses shouldn't conflict with third year courses as much as possible. But these constraints may not always be satisfied completely because of too many courses and limited time slots in a week. To obtain optimal solutions, a multi objective fitness function is proposed. In this function, there are two fitness functions: The first one minimizes the first rule, and the second one minimizes the second rule. To calculate the multi objective fitness function value, firstly the numbers of conflictions are calculated for both fitness function and then the pareto values are calculated for the chromosomes. The non-dominated chromosomes have 0 pareto values. If a solution is dominated by one solution, its pareto value becomes 1, if dominated by two solutions, the value becomes 2 and so on. At the end, pareto values are used to update the main fitness function which is described in Section 3. In update operation the multi objective fitness function value for a chromosome is added to the main fitness function by multiplying its pareto value with 1000 (pareto value ×1000). For example, if the pareto value of an optimal solution (main fitness value is 0) is 0, its main fitness value becomes 0. But if the pareto value of it is 1, its main fitness value becomes 1000, now this is not an optimal solution, so it isn't selected for an optimal solution again.

6 EXPERIMENTAL RESULTS

The implementation is made with Microsoft Visual Studio 2008 compiler. To run the methods firstly the parameters of genetic algorithm should be determined. These parameters are population size, crossover probabilities, mutation probabilities, number of keeping parents (Elitism). Values of these parameters are shown in Table 1. And then the parameters of weekly course timetable are determined. To do this, a computer engineering department's data of a university are used. Values of these parameters are given in Table 2.

Table 1: The parameters of genetic algorithm.

Parameters	Values
Population Size	50
Crossover Probability	%75
Mutation1 Probability	%30
Mutation2 Probability	%10
Elitism	%50

Table 2: The data of weekly course timetable.

Data	Values
Course count	35
Classroom count	9
Instructor count	15
Maximum block lecturehour	3
Gene count	105
Maximum gene value	432
Total time slot	48

Firstly, genetic algorithm which is described in Section 3 has been executed. Then hybrid method has been executed under these parameters several times. After hybridizing genetic algorithm with proposed local search methods, the performance of genetic algorithm is increased incredibly. The comparisons are shown in Table 3.

Table 3: The comparisons of the methods.

Method	Generation	Duration
GA without soft constraints	30000	1 hour
GA with soft constraints	150000	8 hour
GA and proposed local search without soft constraints	3	30 second
GA and proposed local search with soft constraints	10000	20 minute

In all results the hard constraints are satisfied which means a valid weekly course timetable. The local search methods improve the candidate solutions in each generation if possible. Although the local search methods lead to extra time for duration of a generation, they reduce the generation count of the solution. This results in better duration time for the solution. The result of the sole use of genetic algorithm without considering the soft constraints is obtained for about 30000 generation which takes about 1 hour. If the soft constraints are considered, the result is obtained for about 150000 generation which takes about 8 hour. Because of the conflictions and limited resources, getting results by considering the soft constraints takes too much time. After hybridizing the genetic algorithm with proposed local search methods, the performance of genetic algorithm is increased. The results without considering the soft constraints are obtained for about 3 generation which takes about 30 second. The improvement is remarkable which reduces the 30000 generation to 3 generation and 1 hour to 30 second. If the soft constraints are considered, the result is obtained for about 10000 generation which takes

about 20 minute. The improvement is again remarkable when compare with the sole use of genetic algorithm.

7 CONCLUSION

Generating weekly course timetable by human hand is difficult because of its constraints and complex structure. Earlier, the mathematical methods were used but they failed because of the complex structure of the timetabling problem. Later heuristic methods were used such as simulated annealing, hill climbing. Nowadays, genetic algorithm has been used because it searches a lot of candidate solutions at the same time and it does not need any derivative or mathematical information. Although genetic algorithms are well-suited for the global exploration, they usually fail in local exploitation. Because of this, various hybridization methods are proposed. With these approaches, genetic algorithms do global search within the population, while heuristic methods do local search around chromosomes. In this paper, firstly the representation and other methods of genetic algorithm are designed for the weekly course timetabling problem. Then the proposed local search methods are incorporated with genetic algorithm. With this hybridization the performance is increased incredibly. The solution without considering the soft constraints is obtained about 3 generation and with considering the soft constraints is obtained about 10000 generation. These are performed in less than one minute and twenty minute respectively. If it is compared with the operation of weekly course timetable by human hand, it is seen that this method provides gain for time and task force. In the future, this study can be extended for obtaining a faculty course timetable by considering the whole departments in that faculty. And also university scheduling and exam scheduling can be done by considering this study.

REFERENCES

Abramson, D., 1991. Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms. Management Science, 37(1), pp. 98-113.

Abramson, D., Dang, H., Krisnamoorthy, M., 1999. Simulated Annealing Cooling Schedules for the School Timetabling Problem. Asia-Pacific Journal of Operational Research, 16(1), pp. 1-22.

Schaerf, A., 1996. Tabu Search Techniques for Large High-School Timetabling Problems. In *Proc. of the Fourteenth National Conference on AI*. Vol(1), pp. 363-368.

Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley. USA.

Colorni, A., Dorigo, M., Maniezzo, V., 1992. A Genetic algorithm to solve the timetable problem. Technical Report 90-060, Politecnico di Milano. Milano Italy.

Yu, E., Sung, K., 2002. A genetic algorithm for a university weekly courses timetabling problem. International Transactions in Operational research, Vol(9), pp. 703-717.

Ozcan, E., Alkan, A., 2002. Çok Nüfuslu Kararlı Hal Genetik Algoritması Kullanarak Otomatik Çizelgeleme. In *TBD 19. Bilişim Kurultayı*. pp. 149-155.

Yigit, T., 2006. Meslek Liseleri Haftalık Ders Çizelgelerinin Genetik Algoritmalar Yardımıyla Oluşturulması. Gazi Üniversitesi Endüstriyel Sanatlar Eğitim Fakültesi Dergisi, Sayı: 19, pp. 25-39.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., Likothanassis, S. D., 2008. Applying evolutionary computation to the school timetabling problem: The Greek case. Science Direct, Computers & Operations Research, 35(4), pp. 1265-1280.

Karaboga, D., 2004. *Yapay Zeka Optimizasyon Algoritmaları*, Atlas Yayın Dağıtım. Turkey.

Eiben, A. E., Smith, J. E., 2007. *Introduction to Evolutionary Computing*, Springer, ISBN: 978-3-540-40184-1.

Davis, L., 1985. Applying adaptive algorithms to epistatic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*. pp. 162-164.

Cheng, R., Gen, M., Tsujimura, Y., 1999. A tutorial of job-shop scheduling problems using genetic algorithms, part 2: hybrid genetic search strategies. Elsevier, Computers & Industrial Engineering, Vol(36), pp. 343-364.

Renders, J., Bersini, H., 1994. Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. In *Fogel D, editor. Proceedings of the First IEEE Conference on Evolutionary Computation*. FL: IEEE Press, pp. 312-317.

Ishibuchi, H., Murata, T., 1998. A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 28(3).

Abdullah, S., Turabieh, H., 2008. Generating University Course Timetable Using Genetic Algorithms and Local Search. In *Third 2008 International Conference on Convergence and Hybrid Information Technology*. Vol(1), pp. 254-260.

Mumford, L. C., 2007. An Order Based Evolutionary Approach to Dual Objective Examination Timetabling. Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling, (CI-Sched).

Burke, K. E., Petrovic, S., 2002. Recent research directions in automated timetabling. Elsevier, European Journal of Operational Research, 140(2), pp. 266-280.