USING A GENETIC ALGORITHM OPTIMIZER TOOL TO SOLVE UNIVERSITY TIMETABLE SCHEDULING PROBLEM

Sehraneh Ghaemi Mohammad Taghi Vakili Ali Aghagolzadeh
{Ghaemi, Mvakil, Aghagol} @tabrizu.ac.ir
Faculty of Electrical & Computer Engineering
University Of Tabriz

ABSTRACT

University course timetabling is a NP-hard problem which is very difficult to solve by conventional methods. A highly constrained combinatorial problem, like the timetable, can be solved by evolutionary methods. In this paper, among the evolutionary computation (EC) algorithms, a genetic algorithm (GA) for solving university course timetabling problems is applied. Main goal is to minimize the number of conflicts in the timetable. For this purpose two approaches –Modified GA and Cooperative GA- are applied. Results show the modified GA (MGA) method was significantly enhanced algorithm performance with modified basic genetic operators. Intelligent operators improve overall algorithm's behavior. In addition, algorithm performance is considerably improved by using cooperative genetic method.

1. INTRODUCTION

University course timetabling problems (UTPs) are search problems, in which subjects must be arranged around a set of time slots, so as to satisfy given constraints and optimize a set of objectives. Scheduling under constraints hard or soft is complex task, having an NP-complete degree of complexity [1] and generally is associated to dependant attribute domains so necessitates specific optimization algorithms. This means that, in a space of possibilities of finite time, deterministic and exhaustive search techniques may probably fail. The UTPs are generally characterized as constraint satisfaction problems (CSPs). The problems that NP-hard class are very difficult to solve using conventional optimization techniques, such as backtracking, constraint logic programming, or evolutionary algorithms [2-3]. These techniques minimize the total penalty for constraint violation and generate feasible but not optimal solutions. Recently stochastic techniques have been employed using genetic algorithms (GAs) to tackle the scheduling and allocation problem using tabu search in GSA [4] and a force directed crossover in GABIND [5]. Although GA approaches provide good solutions they require larger execution time. There have been various algorithms proposed in the literature for this specific problem, most commonly using benchmark instances [6]. Although each has their own strengths and weaknesses, but as a trend it seems that the algorithms which address both sub-problems simultaneously (e.g. through the use of weightings in the evaluation function) are generally outperformed by those that employ a two-stage strategy whereby feasibility is first obtained and then soft constraints are optimized using operators that restrict the search to feasible areas of the search space [7]. However, whilst there are plenty of works proposing algorithms specializing in soft constraint optimization, there has been less emphasis on producing algorithms that specialize in finding feasibility in the first place. In particular, when considering "harder" problem instances, some of the existing algorithms in the literature could start to fail on this matter. This, therefore, generates a need for new algorithms that provide more robust searches with regards to finding feasibility.
Many kinds of timetabling problems have been released in the literature during recent years [8], [9] but most of them are very specific and do not take into account a lot of elements.
Addressing this shortage, the algorithm was modified and its performance was significantly enhanced with modification of basic genetic operators and applying the cooperative genetic algorithm. The algorithm was tested on data at the faculty of electrical engineering and computer in university of Tabriz. The core of genetic algorithm was developed in MATLAB.

2. PROBLEM DESCRIPTION

The timetables to be treated in this paper are constructed for the undergraduate courses of faculty of electrical engineering in Tabriz university. The students of faculty are included in four groups (for fall semester: semester.1, semester.3, semester.5 and semester.7 students and for spring semester: semester.2, semester.4, semester.6 and semester.8 students). There are four time periods a day and each time period consists of a 90 minutes lecture and a 30 minutes recess. Each subject is lectured in one or two time periods in a week. The problem is defined using five main entities. Lecturer, Student, Subject, Classroom location, Timeslot are the entities that will need to be directly assigned in a completed timetable.

2.1. Problem definition

The lecture timetable is formulated as follows:
Problem= $\{L, S, D, C, W\}$, where $L = \{l_1, l_2, ..., l_t\}$ is a set of lecturers, $S = \{s_1, s_2, ..., s_s\}$ is the set of subjects, $D = \{t_1, t_2, ..., t_d\}$ is a set of timeslots for each subject, each timeslot consists of a day of week, a time period and a location of Classroom, where the day of the week is Saturday to Thursday and the time period is 1 to 4 and $R = \{r_1, r_2, ..., r_c\}$ is the set of locations of Classrooms. Also $C = \{c_1, c_2, ..., c_m\}$ is a set of constraints and $W = \{w_1, w_2, ..., w_n\}$ is a set of weights (i.e., penalties) for the constraints.

2.2. Constraints

The constraints that we treat are classified as hard or soft. Hard constraints are those to which a timetable has to adhere in order to be satisfied. A timetable is not viable if it does not satisfy them. Soft constraints are also those which should not been violated, but if this is not possible then the resulting timetable isn't optimal. Values of weights for various constraints are shown in parentheses.

2.2.1. Hard constraints

C_1 : Each lecturer can take only one class at a time. (1000)

C_2 : A location of classroom can only have one subject assigned to it at a time. (1000)
C_3 : Clashes should not occur between the subjects for students of one group. (1000)

2.2.2. Soft constraints

C_4 : The second time for each subject should not be in the same day. (400)
C_5 : No subject should be allocated to a time period that heads of department don't demand because of other work. (20)
C_6 : The subject should not be allocated to a time period inconvenient for heads of department because of other work. (10)
C_7 : The subject should not be allocated to a time period that a lecturer doesn't demand it. (10)
C_8 : The subject should not be allocated to a time period inconvenient for a lecturer. (5)
C_9 The second time for each subject should not be in the consequent day. (20)

3. GENETIC ALGORITHM OPTIMIZER TOOL

3.1. Chromosome Representation

Encoding of the chromosomes for the GA model is an essential factor in the success of a GA as it will affect not only the efficiency and performance of the GA but also the speed and quality of the final result.
As previously mentioned, there is one timeslot for each subject so a week has 24 time period (Table.1).

Table.1: Time period representation in weekly lecture timetable

SAT	time period 1	t.p. 2	t.p. 3	t.p. 4
SUN	t.p. 5	t.p. 6	t.p. 7	t.p. 8
MON	t.p. 9	t.p. 10	t.p. 11	t.p. 12
TU	t.p. 13	t.p. 14	t.p. 15	t.p. 16
WED	t.p. 17	t.p. 18	t.p. 19	t.p. 20
THUR	t.p. 21	t.p. 22	t.p. 23	t.p. 24

A chromosome consists of two parts which have the equal genes number. The genes are placed in first part of chromosome, present first time period of the subjects and second part of genes presents second time period of the subjects. If the subject is lectured for one time period in a week then the gene corresponds to the subject in second part of chromosome doesn't contribute in programming. The chosen chromosome to present a timetable solution is a vector in which the index of each gene in first part's chromosome shows the subject number. Structure of chromosome is follows:

$$Chromosome = [\underbrace{\overbrace{g_1}^{Lecturer1}\ g_2\ \cdots\ \overbrace{g_{n.sub}}^{Lecturer.N}}_{Firsttime}$$
$$\underbrace{\overbrace{g_{n.sub+1}}^{Lecturer1}\ g_{n.sub+2}\ \cdots\ \overbrace{g_{n.sub*2}}^{Lecturer.N}}_{Secondtime}]$$

In these chromosomes, the subjects are lectured with the heads of department are placed in first and then the subjects related to other lecturers are placed. Value of gene specifies a timeslots. The number of timeslots in each week is 24 whose value results the day and the time period of the subject (Table.1) but the location number of the classroom can not be determined from this number. Therefore, for considering the location number, integer values between 1 and the number of timeslots in a week multiplied by number of classroom locations is assigned to the genes. The i^{th} classroom location corresponding to integer value between (i-1) multiplied by (the number of timeslots in a week +1) and i multiplied by the number of timeslots in a week is considered. By considering 6 classroom locations for algorithm testing, 8 bits are assigned for each gene. Table.2 shows instance of chromosome obtained information.

Table. 2. Instance of chromosome information for the lecturer that three subjects lecture.

Subject	Electronic1	Machin2	Machin1
Timeslot	(Mon,1,2) (WED,2,2)	(TUE,2,1) (SAT,3,1)	(TUE,1,3) (SUN,2,2)

3.2. Normalization

A gene with 8 bits has $2^8 = 256$ possible integer values. If 6 classroom locations are considered then, we need 144 integer values. Thus, integer values between 0 and 255 normalize to integer values between 1 and 144. Normalization is applied for the binary encoding and decoding of the genes.

3.3. Genetic Operators

The genetic operator for the GA tool is defined in following sections:

3.3.1. Selection

So two selection methods are applied:
a) Roulette wheel weighting in which the Rank Weighting technique is used.
b) Tournament selection in which for selecting each parent, three candidate chromosomes are used.

3.3.2. Crossover

Three types of crossover are applied here as follows:

- Single point crossover (SX)
- Uniform crossover (UX)
- Modified uniform crossover (MUX)

Generally, it has been shown that the uniform crossover is more effective for many problems especially for numerical optimization problems. For improving UX operator performance, the bits in *parent.1* are compared with the corresponding bits in *parent.2*. If the bits of i^{th} gene in *parent.1* are equal to the bits of i^{th} gene in *parent.2* then this gene is passed to the corresponding gene in child. No additional conflicts can be added through copying those equal values. But, if the i^{th} bits in two parents are different, then 1^{st} to $(i-1)^{th}$ genes in child are replaced to the corresponding genes in parents. After calculating the number of conflicts and the cost for both parents, the gene of parent with lower cost value is chosen. By using this crossover, two chromosomes produce only one offspring.

3.3.3. Mutation

So, single point mutation is used which changes a 1 to a 0 Mutation points are randomly selected from the $N_{pop} \times N_{bits}$

total number of bits in the population matrix. We don't allow a bit to be selected several times and also mutation doesn't occur on the best solution.

3.4. The fitness function

The penalty function for a problem with *m* constraints would then be as below:

Cost function $= \sum_{i=1}^{m} w_i \delta_i$; $\begin{cases} \delta_i = 1 \\ \delta_i = 0 \end{cases}$

Where, $\delta_i = 0$ if constraint *i* is satisfied and $\delta_i = 1$ if constraint *i* is violated. Also *m* is the number of constraints, and w_i is a constant imposed for violation of constraint *i*.

3.5. The repair function

After the crossover and mutation operation, the resulting chromosome may become unfeasible or is outside the search space [10]. Two repair methods are available: random or deterministic. From experiments reported by Michalewicz (1996) it seems that deterministic repair gives much better results than random repairs.

4. THE ALGORITHM IMPLEMENTATION

4.1. The modified GA (MGA)

The program starts with two groups of chromosomes known as pop_1 and pop_2. They are random populations which are generated separately. The chromosomes of pop_1 and pop_2 consist of 40 genes. For creating full chromosomes, three candidate chromosomes are randomly selected from pop_1. Then each candidate chromosome of pop_1 contributes with pop_2 for producing full chromosome. This method acts on pop_2 the same way. This method has the advantage that can decrease population size and results are achieved at a shorter execution time. The algorithm of the MGA method is shown in Table. 3.

4.2. The cooperative GA (CGA)

Genetic algorithms slowly operate for large search space. Therefore for acceleration, co_evolution algorithm is applied**.** So pop_1 and pop_2 are placed respectively in $species_1$ and $species_2$. As populations are separately generated, two species are separate. Also they will contribute together for decreasing cost value. Therefore, cooperative type of co_evolution is used. So three candidate chromosomes from $species_1$ and $species_2$ are selected randomly (Fig.1).
The algorithm of the CGA method is shown in Table. 3.
The MGA and CGA methods are done for 6 classroom locations, 40 subjects and 18 lecturers where four of them are the heads of department.

4.3. Results

The results for three types of crossover with the MGA method and Rank Weighting selection for different mutation rates are shown in Table. 4. The results presented in Table. 4 clearly show that SX operator in upper mutation rates and UX and MUX operators in lower mutation rates achieves better results. Comparison between obtained results shows that minimum conflict number occurred with MUX operator. Comparison between Table.4 and Table.5 show the number of unsatisfied constraints with Tournament selection are less than Rank Weighting selection method. In Table 6, CGA Method in mutation rate of 0.02 with Tournament selection has minimum conflicts number. Note that number of unsatisfied constrains are zero for other constrains. These constrains don't appear in Table 5 and 7. Comparison between results in Table 7 shows that the CGA method is better than the MGA method. Also program has been done for varying mutation rate. Results are:
$C_4, C_5, C_6 = 0$, $C_6, C_8 = 49$, $C_9 = 18$, Total of conflicts=67, Cost value=700
Results of Table.7 show that the MGA and CGA methods with lower population size presents better results than GA. The CGA method is also better than the MGA and GA methods. The cost value versus iterations for the CGA method in Fig.2 is shown.

Table.3: Steps of MGA and CGA methods

Step	MGA method	CGA method
1	Generate pop_1 and pop_2.	Generate $species_1$ & $species_2$
2	Produce the full population as mentioned method.	Produce the full chromosomes for each species as mentioned method.
3	Produce the weekly lecturer timetable	
4	Apply the repair functions	
5	Evaluate fitness of the chromosomes	Evaluate fitness of the obtained full chromosomes for each species
6	Repeat steps 6.1 to 6.6 until achieve to stop conditions	
6.1	Selection and Crossover	
6.2	Mutation: a) Constant μ over all generation. b) First generation has μ=0.2 when minimum cost value is fixed in 10 subsequence iteration, decrease the mutation rate. Note that minimum mutation rate is 0.02.	
6.3	Produce the full population as mentioned for pop_1 and pop_2.	Produce the full chromosomes for each species as mentioned method
6.4	Apply the repair functions	
6.5	Evaluate fitness of the chromosomes	Evaluate fitness of the obtained full chromosomes for each species
6.6	stop condition: a) Achieve to the defined minimum cost. b) Achieve to the defined maximum iteration. c) Having unchanged minimum cost value in 15 consequence iteration with iteration number above 30.	
7	Select the best chromosome of full chromosomes	

5. CONCLUSION

This type of chromosome representation needs less number of bits for each gene. Therefore algorithm needs shorter time for running. The obtained results show that SX operator in upper mutation rates and UX and MUX operator in lower mutation rates perform well and MUX operator is better than others. By varying mutation rate during algorithm running better results are achieved. Also Tournament selection operates better than Rank

Weighting selection. The MGA and CGA methods with population size less than GA yield better results and accelerate the algorithm. Note that the unsatisfied constraints number with CGA method is less than MGA method.

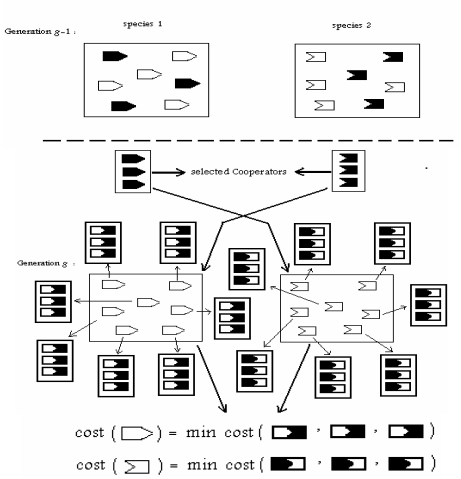

Fig 1: Computing cost value for chromosomes sp_1 & sp_2

Table 4: Results for MGA method with Rank Weighting selection (popsize=50 and applying 3rd stop condition)

Mutation rate	Cost value for types of Crossover			Number of conflict for types of Crossover		
	SX	UX	MUX	SX	UX	MUX
0.02	2080	1540	1980	165	144	140
0.04	1710	1310	1970	146	133	161
0.06	1870	2380	1770	145	185	142
0.08	1750	1490	1300	151	137	119
0.1	1860	1970	1960	168	160	148
0.12	1350	1870	1820	128	157	151
0.14	1850	2240	1540	155	177	144
0.16	1600	2170	1990	140	166	163
0.18	1310	1410	1780	121	135	144
0.2	1930	1650	2010	154	148	154

Table 5: Results for MGA method with Tournament selection and MUX operator (popsize=50&applying 3rd stop condition)

Mutation rate	Number of conflict			Cost value
	C_6, C_8	C_9	Total	
0.02	78	66	144	1860
0.04	72	36	108	1210
0.06	115	60	175	2000
0.08	94	60	154	1850
0.1	81	72	153	2000
0.12	107	60	167	1940
0.14	80	78	158	2110
0.16	105	30	135	1340
0.18	93	36	129	1360
0.2	88	60	148	1800

Table 6: Results for CGA method with MUX (#generation=60 popsize=50)

Mutation rate	Number of conflicts		Cost value	
	Rank weighting	Tournament	Rank weighting	Tournament
0.02	95	82	1120	800
0.04	127	140	1350	1670
0.06	110	133	1390	1540
0.08	126	125	1350	1250
0.1	125	123	1490	1400

Table 7: Results of three methods with Tournament selection, MUX operator and applying 3rd stop condition

Constraint (C)	Number of conflicts with mutation rate=0.04		
	GA Popsize=40	MGA Popsize=30	CGA Popsize=30
C_4,C_5,C_7	0	0	0
C_6,C_8	108	104	86
C_9	66	60	42
Total	174	164	128
Cost value	2070	1920	1430

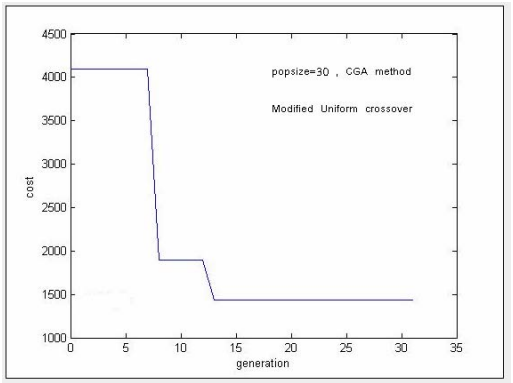

Fig. 2. Evolution of cost value versus iterations with CGA method and population size of 30

REFERENCES

[1] R. Piola, "Evolutionary solutions to a highly constrained combinatorial problem", Dipartimento di informatica, Università degli di Torino, Italy, 1992.

[2] M. Grobner, et al., *"A standard framework for timetabling problems,"* Proc. International Conference on the Practice and Theory of Automated Timetabling 2002, LNCS 2740, pp.24-38, 2003.

[3] H.Rudova, et al., "*University course timetabling with soft constraints,"* Proc. International Conference on the Practice and Theory of Automated Timetabling 2002, LNCS 2740, pp. 310-328, 2003.

[4] Sait, S., Ali, S., and Benten, M.: *"Scheduling and allocation in highlevel synthesis using stochastic techniques,"* Microelectron. J., 1996, 27, pp. 693–712

[5] Mandal, C., Chakrabarti, P., and Ghose, S.: *"GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths,"* IEEE Trans. Very Large Scale (VLSI) Syst., 2000, 8, (6), pp. 747–750

[6] International Timetabling Competiton accessed March 2005. http://www.idsia.ch/Files/ttcomp2002/

[7] Kostuch P. (2004) "The University Course Timetabling Problem with a 3-stage approach", In E. Burke, M. Trick (Eds.) Practice and Theory of Automated Timetabling - PATAT V, pp 251-266.

[8] R. Weare, E. Burke, and D. Elliman, "A Hybrid Genetic Algorithm for highly Constrained Timetabling Problems", Department of Computer Science, January 1995.

[9] A. Hertz, "Tabu Search For Large Scale Timetabling Problem", *European Journal of Operational Research*, Vol. 31, N° 1, 1991, pp. 39-47.

[10] O.Mahdi, R.N.Ainon and R. Zainuddin, *"Using a Genetic Algorithm Optimizer Tool to Generate Good Quality Timetables,"* Proc. of the 2003 10th IEEE International Conference on Volume 3, 14-17 Dec. 2003 Page(s):1300 - 1303 Vol.3