Annales UMCS Informatica AI 1 (2003) 115-125

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Evolutionary algorithms for timetable problems

Paweł Myszkowski*, Maciej Norberciak

Faculty Division of Computer Science, Wrocław University of Technology,
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland

Abstract

The university course timetabling problem is hard and time-consuming to solve. Profits from full automatisation of this process can be invaluable. This paper describes architecture and operation of two automatic timetabling systems. Both are based on evolutionary algorithms, with specialised genetic operators and penalty-based evaluation function. The paper covers two problem variations (theorethical and real-world), with different sets of constraints and different representations. Moreover, specification of both solutions and a proposal of hybrid system architecture is included.

1. Introduction

Timetabling problems arouse interest of many scientists. The most popular methods of solving them are based on heuristics, local search techniques (mostly tabu search), evolutionary algorithms and reduction to graph coloring, Recently, the systems based on expert systems and constraint programming have become popular. Publications concerning heuristic methods are [1-2]. Description of solutions based on local search techniques is located in [3-5] (where tabu search-based methods are described) and in [6-7] (simulated annealing). Authors of [4, 8-13] applied evolutionary algorithms, and proposals of solutions based on graph coloring, and also models and heuristics applicable to this approach can be found in [14-21]. Description of approaches based on expert systems are located in [22-24]. Many of the methods mentioned are described in surveys like [12, 25].

We present some approaches to solve large, highly constrained timetabling problems, based on evolutionary algorithms. In chapter 2 we describe two variations of the problem and constraints connected with them. The third chapter includes the description of specimen representation, evaluation function, initialization of population and approaches to solution in detail. The proposal of

* Corresponding author: *e-mail address:* pawel.myszkowski@ci.pwr.wroc.pl

hybrid system architecture is also included in that chapter. The last chapter points to directions of future work.

2. Problem description

The typical timetabling problem consists in assigning a set of events (e.g. classes) to a set of resources (e.g. rooms) and timeslots, satisfying a set of constraints of various types. Constraints stem from both nature of timetabling problems (e.g. two events with the same resources involved cannot be planned at the same time) and specificity of the institution involved.

The problem we consider is a typical university course timetabling problem. It consists of set of events (classes), to be scheduled in a certain number of timeslots, and a set of rooms with certain features and size, in which events can take place. We've been working on two variations of this problem – the first variation has a defined set of students attending each event, but no constraints are related to teachers. In this variation the number of timeslots is 45 (5 days, 9 timeslots each). Test sets for this variation come from the International Timetabling Competition. The second variation has some constraints related to teachers' availability (each event has a teacher assigned and each teacher has a defined set of forbidden timeslots), but set of students attending each event is undefined (only a number of students and faculty they attend is known) and has to be concluded from other data. In this variation the number of timeslots is 35 (5 days, 7 timeslots each) and each class has a defined course (the class is a part of particular university course). Test sets for this variation come from the Faculty of Computer Science and Management of Wroclaw University of Technology. The timetable in this case consists of almost 1000 events conducted by almost 200 teachers in about 40 rooms.

A feasible timetable is one in which all the events have been assigned a timeslot and a room, and the following hard constraints are satisfied:

- only one event is scheduled in each room at any timeslot (both variations),
- the room is big enough for all the attending students and satisfies all the features required by the event,
- no student attends more than one class at the same time (first variation),
- no teacher carries on more than one class at the same time (second variation),
- no teacher carries on any class in timeslot which is forbidden for him (second variation),
- if particular course has only one class assigned, no class with students from the same faculty is scheduled at the same timeslot with this course (second variation).

In first variation there are also three soft constraints defined; they are broken if:

- a student has a class in the last slot of the day (S1),
- a student has more than two classes in a row (S2),
- a student has a single class on a day (S3).

3. Approaches tested

All approaches we describe are based on evolutionary algorithms, which turn out to be useful as a general-purpose optimization tool, due to their high flexibility accompanied by conceptual simplicity. In this part of the paper we describe three conceptions with the results of their application to the problems illustrated earlier. In all approaches the evaluation function is based on penalty for breaking constraints. In the first variation we decided to represent the specimen (genotype) as a matrix where each row corresponds to a room and each column to a timeslot. Each element of the matrix (gene) may contain an event (no more than one – this allows us to eliminate the first hard constraint – only one event is scheduled in each room at any timeslot). In the second variation representation is direct – each gene was represented by a triple <*event*, *room*, *timeslot*> (where *event* represents teacher and a class). Timeslots can be odd and even (some classes are conducted once a fortnight). Genes are organized in chromosomes, and every chromosome represents one faculty. All faculties together form a genotype. In both approaches each genotype represents particular solution (timetable). Initial population can be generated randomly, heuristically or peckish (peckish initialization strategies are described in [2] – this strategy is used only in the second variation). Random generation strategy chooses events to be scheduled, rooms and timeslots at random; the next uses heuristic event sequencing strategies to create better (in terms of the evaluation function) initial solutions. The third strategy chooses k genes at random and inserts into timetable one that breaks least constraints (induces least conflicts). $K = 1$ corresponds to random initialization, while in the limit as k gets large this becomes equivalent to greedy initialization. Creation of population in subsequent generations is archived by means of classical genetic roulette, as described in [26].

3.1. Random mutation

In “classic” evolutionary algorithm in each iteration after selection some specimens are exposed to genetic operators – mutation and crossover. The contents of operator set and their operation depend strongly on both specifics of problem being solved and approach chosen. In our solution only mutation operators are used – it’s combined with high computational and conceptual complexity of recombination operator. Only places, events and timeslots can be

mutated – it gives us a set of three different mutation operators. We conducted experiments with this approach on both problem variations – test sets for the first variation had about four hundred events to schedule in a dozen or so rooms, the second variation involved about seven hundred to one thousand events, with ca. two hundred teachers involved, to schedule in about thirty rooms. Every iteration of each specimen is mutated once and operator is chosen randomly. For both problem variations the solution wasn't found in reasonable time – the algorithm got stuck in local minimum pretty soon and was not able to escape from it. There are many reasons for this phenomenon. "Blind" mutation is not only able to improve, but also spoil the plan being mutated. The changes made may not be reflected in the value of evaluation function – e.g. mutation can liquidate a conflict (broken constraint) introducing new conflict at the same time, not changing the objective function value as a result of this operation. Additionally, selective pressure in population, where there are many conflicts in specimens, may be too weak, for the sake of very small differences between specimens in terms of evaluation function. These factors disqualify "blind" mutation as a solving tool for complex constraint satisfaction problems and point to more complex, heuristics-based methods.

3.2. Directed mutation

Despite their unquestionable assets, evolutionary algorithms don't copy directly human way of thinking. Human timetabling process is an application of direct heuristics, based on successive augmentation – the events are inserted into timetable one by one, until all have been scheduled. The most constrained events are scheduled (and during improvement of the timetable rescheduled) first. Analysis of this approach allowed us to create an algorithm, which solves the second variation of the problem effectively.

In this approach we use mutation operators directed by broken constraints. The place in genotype (triple <*event*, *room*, *timeslot*>), which breaks the most constraints (so it is most difficult to schedule) is selected to mutate. The set of operators was enlarged by special operator, which is able to "clean" a timeslot for the course, which has only one class assigned. The operators try to reschedule event in such a way, that they would eliminate one particular type of conflict, caused by this event – a dozen or so (this number was established arbitrary) possible variants are examined, and the one, that breaks the least constraints of particular type is chosen. This is an extension of the peckish initialization paradigm on genetic operators. The order of eliminating the conflicts is established after random generation of the initial population, before the first iteration of the algorithm – often a particular type of conflict appears in the initial population, the sooner algorithm tries to eliminate it. Application of any operator can spoil timetable in terms of both evaluation function and number

of constraints broken, but allows the algorithm to escape the local optima efficiently. During the tests on data sets described previously a feasible solution is found after about one thousand iterations – this number depends weakly on the population size. This phenomenon is caused by significant determinism of this approach and weak selective pressure, described earlier. The experimental results are shown in table 1.

Table 1. Experimental results (second problem variation)

Population size	Generation in which feasible solution is found		
	Highest	Lowest	Average (10 runs)
1	1895	423	1101
10	1756	453	1130
50	1788	413	1099
100	1770	397	1084
500	1774	330	1054

The following set of operators has been proposed to solve the first problem variation with the matrix representation of the specimen:

- MOVE – chooses an event at random and moves it to random room and/or timeslot; probability of drawing the event is directly proportional to a number of constraints broken by this event;
- ORDER – tries to sort events in timeslot chosen at random, so constraints concerning room features and size are satisfied;
- REPLACE – chooses two columns (timeslots) at random and swaps them; this operator affects soft constraints.

These operators allow the algorithm to leave local minima and optimize both soft and hard constraints. This gives the algorithm the ability to improve the value of the objective function constantly. We found feasible solutions for all the test problems. Features of datasets used are shown in table 2 and the experimental results in table 3. Algorithm stops when feasible solution is found. Table 3 shows how many soft constraints of each type have been broken on average in the feasible solution.

Table 2. Datasets description (first problem variation)

Problem type	Small	Medium
Num events	100	400
Num rooms	5	10
Num features	5	5
Approx features per room	3	3
Percent feature use	70	80
Num students	80	200
Max events per student	20	20
Max students per event	20	50

Table 3. Experimental results (first problem variation)

	S1	S2	S3
Small1	8	71	14
Small2	13	88	30
Small3	2	58	14
Small4	5	67	2
Small5	1	36	38
Medium1	232	105	2

3.3. Non-Darwinian evolutionary computation

The disadvantages of 'classic' genetic algorithms, such as tendency to get stuck in the local optima have inclined the extension of the GA paradigm. Hybrid systems, which can be a mix of different methods, with changes in specimen representation or adaptation of genetic operators [26] can solve problems from different classes – classification [27], approximation or combinatory problems (e.g. travelling salesman problem), but they still operate on genotypes within the confines of the Darwinian evolution. Such evolution looks more like blind search than the sequence of reasonable acts [27]. The hybrid method which uses GA with the addition of machine learning (ML) module was proposed in [28]. In LEM (*Learnable Evolution Model* [27, 28]) GA produces a solution, which is evaluated and analyzed by ML module, to be consequently evolved by GA. The LEMMATA (*LEM meant for Timetabling*) hybrid we propose is based on LEM concept – picture 1 shows the information flow between GA and ML modules. After generation of initial population all solutions are evaluated. Then we check the stop condition of GA and analyze the parameters, which decide whether the data should be sent to GA or ML modules. GA module creates the next generation by means of selection and genetic operators. This cycle is repeated until the parameters dependent on solution quality decide to start up the ML module. It analyzes solutions and chooses, which are "good", and describes them with a set of rules. The rules are used to create new population, which takes part in the next iteration of the algorithm.

That model allows the algorithm to improve its speed, because only "good" timetables are chosen to generate new generation in the exit point of ML module. Quality of solutions also improves, thanks to the rule-based generation of descendant population. LEMMATA have been implemented already – the experimental results with this model can be found in [29].

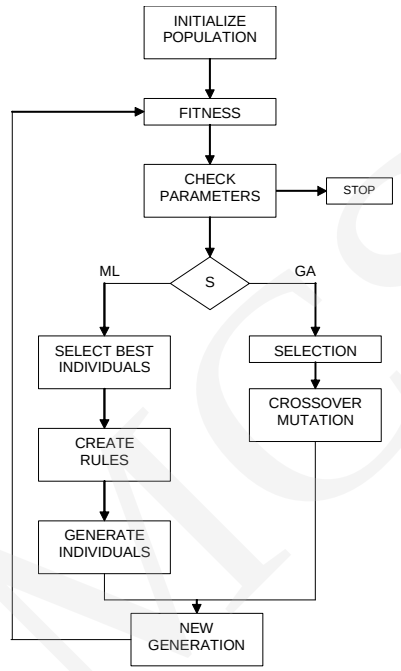

Fig. 1. LEMMATA operation schema

That model allows the algorithm to improve its speed, because only "good" timetables are chosen to generate new generation in the exit point of ML module. Quality of solutions also improves, thanks to the rule-based generation of descendant population. LEMMATA have been implemented already – the experimental results with this model can be found in [29].

4. Conclusions and future work

GA-based systems are good alternative for solutions of university course timetabling problems based on heuristics or local search techniques. Solving large, highly constrained problems using only the "blind" mutation operator seems rather impossible, but taking advantage of specialized genetic operators, adjusted to specificity of the problems allows finding feasible solution relatively fast. Future work on that field will concern hybrid systems, such as LEMMATA, described in this paper.

References

[1] Burke E.K., Newall J.P., Weare R.F., *A Simple Heuristically Guided Search for the Timetable Problem*, Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems. ICSC Academic Press, Nottingham, (1998).

[2] Corne D., Ross P., *Peckish Initialisation Strategies for Evolutionary Timetabling*, Proceedings of the First International Conference on the Theory and Practice of Automated Timetabling, Napier University, Edinburgh, (1995).
[3] Alvarez-Valdes R., Crespo E., Tamarit J.M., *Design and implementation of a course scheduling system using Tabu Search*, European Journal of Operational Research, 137 (2001).
[4] Ross P., Corne D., Hsiao-Lan Fang, *Successful Lecture Timetabling with Evolutionary Algorithms*, Workshop Notes, ECAI'94 Workshop, (1994).
[5] Schaerf A., *Tabu Search Techniques for Large School Timetabling Problems*, Tech. rep. CS-R9611, CWI, Amsterdam, (1996).
[6] Newall J.P., *Hybrid Methods for Automated Timetabling*, PhD Thesis, Department of Computer Science, University of Nottingham, (1999).
[7] Thompson J.M., Dowsland K.A., *A Robust Simulated Annealing Based Examination Timetabling System*, Computers Ops Research, 7/8 (1998) 25.
[8] Colorni A., Dorigo M., Maniezzo V., *Genetic Algorithms and Highly Constrained Problems: the Time-Table Case*, Proceedings of the First International Workshop on Parallel Problem Solving from Nature, Lecture Notes in Computer Science (1990) 496.
[9] Colorni A., Dorigo M., Maniezzo V., *Genetic Algorithms: a New Approach to the Time-Table Problem*, Lecture Notes in Computer Science – NATO ASI Series, F82, Combinatorial Optimalization, (1990).
[10] Colorni A., Dorigo M., Maniezzo V., *A Genetic Algorithm to Solve the Timetable Problem*, Tech. rep. 90-060, Politecnico di Milano, (1992).
[11] Norberciak M., *Algorytm ewolucyjny w rozwiązywaniu silnie ograniczonego, rozległego problemu planowania*, Materiały V Konferencji Naukowej „Sztuczna inteligencja", Wydawnictwo Akademii Podlaskiej, Siedlce, (2002), in Polish.
[12] Norberciak M., *Przegląd metod automatycznego planowania – przykład wykorzystania algorytmu genetycznego w rozwiązaniu prostego problemu planowania*, Prace Naukowe Wydziałowego Zakładu Informatyki Politechniki Wrocławskiej, Sztuczna Inteligencja nr 1, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, (2002), in Polish.
[13] Ross P., Corne D., *Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetabling Problems. Evolutionary Computing*, AISB Workshop, Sheffield 1995, Selected Papers, T. Fogarty, Springer-Verlag Lecture Notes in Computer Science, 993 (1995).
[14] Burke E.K., Elliman D.G., Weare R.F., *A University Timetabling System based on Graph Colouring and Constraint Manipulation*, Journal of Research on Computing in Education, 27 (1994) 1.
[15] Čangalović M., Kovačević-Vujčić V., Ivanović L., Dražić M., *Modeling and solving a real-life assignment problem at universities*, European Journal of Operational Research, 110 (1998).
[16] Hilton A.J.W., Slivnik T., Stirling D.S.G., *Aspects of edge list-colourings*, Discrete Mathematics, 231 (2001).
[17] de Werra D., *Extensions of coloring models for scheduling purposes.* European Journal of Operational Research, 92 (1996).
[18] de Werra D., *The combinatorics of timetabling*, European Journal of Operational Research, 96 (1997).
[19] de Werra D., *Restricted coloring models for timetabling*, Discrete Mathematics, 165/166 (1997).
[20] de Werra D., *On a multiconstrained model for chromatic scheduling*, Discrete Applied Mathematics, 94 (1999).
[21] de Werra D., Mahadev N.V.R., *Preassignment requirements in chromatic scheduling*, Discrete Applied Mathematics, 76 (1997).
[22] Burke E.K., MacCarthy B., Petrovic S., Qu R., *Structured cases in case-based reasoning--reusing and adapting cases for time-tabling problems*, Knowledge-Based Systems, 13 (2000).

[23] Foulds L.R., Johnson D.G., *SlotManager: a microcomputer-based decision support system for university timetabling*, Decision Support Systems, 27 (2000).
[24] Shie-Jue Lee, Chih-Hung Wu, *CLXPERT: A Rule-Based Scheduling System*. Expert Systems With Applications, 9(2) (1995).
[25] Schaerf A., *A Survey of Automated Timetabling*, Tech. rep. CS-R9567, CWI, Amsterdam, (1995).
[26] Michalewicz Z., *Algorytmu genetyczne + struktury danych = programy ewolucyjne*, Wydawnictwa Naukowo-Techniczne, Warszawa, (1999), in Polish.
[27] Michalski R., *Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning*, Machine Learning, 38 (2000), in Polish.
[28] Michalski R., *Learning and Evolution: An Introduction to Non-Darwinian Evolutionary Computation*, Invited paper, Twelfth International Symposium on Methodologies for Intelligent Systems, Charlotte, NC, (2000).
[29] Myszkowski P.B., *Niedarwinowska ewolucja w rozwiązywaniu problemów planowania*, XI Konferencja „Pozyskiwanie wiedzy i jej zarządzanie", Turawa, 16-18 maja 2003. Prace Naukowe Akademi Ekonomicznej we Wrocławiu nr 975, Wrocław, (2003), in Polish.