SCHOOL TIMETABLE GENERATING USING GENETIC ALGORITHM

Jiří Voráč, Ivo Vondrák, Karel Vlček

VSB – Technical University of Ostrava
Czech Republic

Abstract: This paper describes a possible way to generate timetable. Timetable problem is NP – hard (Cooper and Kingston, 1995), so there is none known deterministic algorithm solving this problem in feasible time. Many stochastic approaches were used including genetic algorithms (GA). In this paper chromosome format (representing one possible timetable) and its fitness evaluation is described and genetic operators: crossover, selection and reproduction and mutation are defined. This implementation uses special mutation: "repair" mutation not only serves to leave local optimum, but it moves GA to more promising areas in search space. Best results giving parameters of GA were measured and stated.

Keywords: Genetic algorithms

1. INTRODUCTION

Paradigms of design theory are defined for the GA by natural selection as a quantitative measurement of life and ability to contribute to the reproduction process. The second property is the random genetic difference – mutation of the genetic information (this is important for small populations. The third typical property is the process of reproduction, which is the opportunity for genetic information exchange – crossing, which is how the algorithm to be effective.

The basic properties of design algorithms are defined as follows:
Algorithm design can be an effective tool for optimisation of processes. (They are used in cases, in which the aim is a searching of global extreme with a lot of several local extremes of the same type exists).

Design algorithms give the tools for monitoring of properties of k -time generation. It can be monitored some changes, which are made by the changes, and by this way of comparison of results of experiments without the changes, and with the change of information in "chromosome".

Design algorithms give the opportunity to "social relations" monitoring between the generations on he basis of information changes of "chromosome". (Kvasnicka, Pospichal and Tino, 2000)

2. PROBLEM DEFINITION

The program is asked to read the input data, and to generate the responsible structure, which will be suited for following processing by GA, and it is asked to start GA process on the data. The GA process will concatenate the rooms and time to the subjects by the way, to not disturb any of the "serious" limits:

- teacher cannot give more the one lecture in the same time
- teaching group cannot learn more subjects in the same time
- room can be devoted to only one subject in the same time
- type of room must fulfil demands of the subject
- room must be sufficient for the number of students
- room must not be too big for the student group

The Java language was used for the implementation.

UDPJ, ben64, L101, size=18, type=uc, length=2, time-period 1.	UDPJ, ben64, L101, size=18, type=uc, length=2, time-period 2.	UDPJ, ben64, L102, size=16, type=uc, length=2, time-period 1.	UDPJ, ben64, L102, size=16, type=uc, length=2, time-period 2.	UDPJ, ben64, L103, size=15, type=uc, length=2, time-period 1.	UDPJ, ben64, L103, size=15, type=uc, length=2, time-period 2.
G317, type=poc, number of students=25	NK320, type=poc, number of students=80	E320, type=poc, number of students=25	E322, type=poc, number of students=16	D312, type=poc, number of students=20	D312, type=poc, number of students=20
5	4	45	12	7	21

Fig. 1: An example of chromosome = timetable.
The first row (Data.sub Array) is common for every chromosomes of population, and it contains the lectures of all subjects in sequence (UDPJ – shortcut for subject Introduction to programming languages, ben64 – teacher ID, L10* - group ID). The second row (Chromosome.rooms) defines the rooms for the individual lectures. The third row defines the time periods for the individual lectures. In this case it is the timetable for the three double units of time for the three different groups. The same teacher gives the lectures.

3. INPUT DATA DESCRIPTION

The input data are described by the XML. The data contain the list of groups including the number of students in the individual groups, the list of teachers, the list of rooms including the number of seats and the type of the teaching room (classical, the computer room, laboratory) and the list of lectures.

The individual lectures are connected with the groups, and the number of tie units, asked type of rooms and the name of teacher equip them. The lectures can be completed by the data of the number of students. If it is not defined, the number of students will be computed from the input data.

4. CHROMOSOME – DATA STRUCTURE

The every row of input data of the teacher, group or room, it is generated object of corresponding class. The description of data of subjects is processed by the different way. Every lecture (group + teacher + type of room and time period) is represented by the duly classes of the Subject, compared to the number of the time periods occupies.
If it is, for example, the lecture with the time period 3, it is generated 3 objects Subject with the same data, which differs by the internal counter only. By other words: one object Subject represents a part of longer teaching unit (lecture, exercise, …) of one time unit (45 minutes). These objects are saved in the array Data.subArray.

Every generated chromosome contains two arrays of equivalent size as Data.subArray. Into the first array Chromosome.room there are saved rooms, and into the second Chromosome.periods are saved time periods. By this way are generated triplets (lecture, room, and time period). The example of chromosome is shown at the Figure 1. We can see (from the left to the right) three double time periods (lectures, examples, …) in the first row.

The both value and structure of these objects is not changed through the algorithm running, and there are common for all chromosomes. Every chromosome contains another two rows (rooms and time periods) which concatenates, by the array index, to every Subjects in Data.SubArray the room and time period. The values in Chromosome.periods and Chromosome.rooms are different in every chromosome, and they generate the variants of timetable by this way.

Note to the time representation: the time period of the whole week is the natural number in the interval from 1 to Periods.getMaxPeriods(). If it is, for example, the maximum time 75, the Monday first time period is 1 and the last is 15, the Tuesday contain 16, … 30, and so on.

5. FITNESS

The fitness function (Chromosome.fitness()) is devoted to solving the suitability of timetable represented by the chromosome. The calculation is based on the evaluation of “penalty” if it is disturbed any limitation. If the fitness is equal to zero, it was found good timetable.
Due to the sequence of elements with the length *n*, Data.subArrays belongs to the one lecture (lecture or exercise), *n* elements must be concatenated with the same room (the both time periods are done in one room) and the time slots must be in the time sequence.

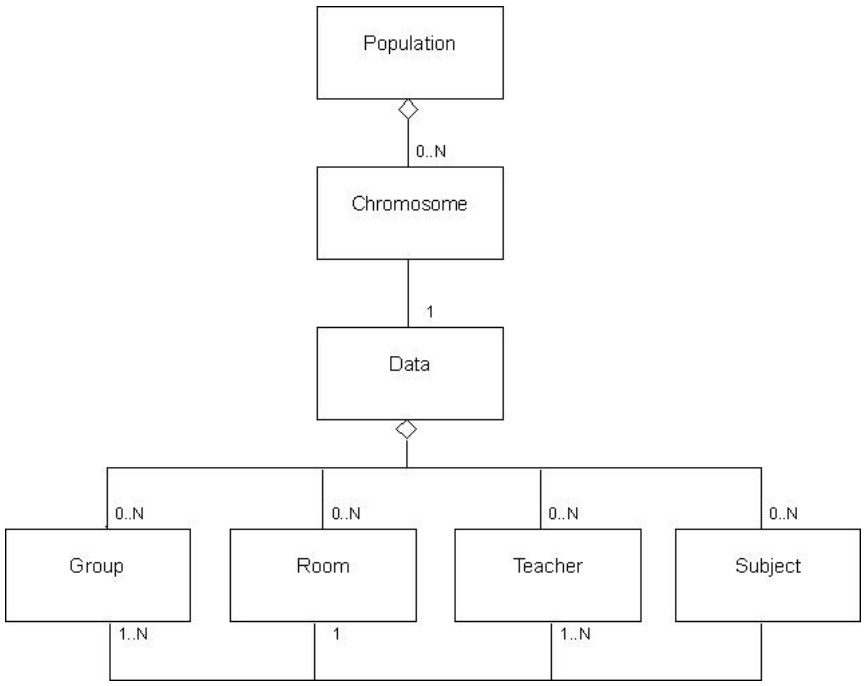

Fig. 2: Class diagram

The method Chromosome.fitness() (respective Chromosome.cFitness()) is penalised in:

- various teaching time periods belonging to one lecture are not given at the same room
- various teaching time periods belonging to one lecture are not given at the same day (the time 15 is the last on Monday, the time 16 is the first on Tuesday)
- teaching time periods are not in the sequence (for triple periods is optimal 5+6+7 or 7+6+5, but it is wrong 4+5+7)
- type of the room not corresponds to the activity (programming exercises without computers)
- room have not sufficient large (the room for 15 persons is not sufficient for the group of 20 persons)
- group is too small for the room
- one room is planned for two lectures in the same time
- one teacher is planned for two lectures in the same time
- one group is planned for two lectures in the same time

6. CROSSING

The new descendant is started (by the construct Chromosome(Chromosome parent1, Chromosome parent2)) by crossing of two parents in every cases. The descendant receives random part of field Chromosome.rooms, and Chromosome.periods for the first parent, the rest it receive from the second parent. The cross-point is various for the two fields.

7. MUTATION

The mutation of the chromosome is realised by the call of the method Chromosome.mutate(). It can be realised one of four possibilities. The first two cases are random changes of the chromosome:

- Exchange of randomly chosen elements of Chromosome.rooms, array, and two elements of Chromosome.periods – the two randomly chosen lectures are exchanged in time periods, and the another two are exchanged in their rooms.
- The randomly chosen lecture is substituted by the randomly chosen time period, and to another lecture is randomly chosen room.

These two kinds of mutation were tested stand alone. The algorithm did not rich asked results, and due to this reason, there were completed another two mutations for their "correction". These new mutations stream to solve the higher complexity of searching, which have its origin in dividing one lecture into more Subjects. The both mutations ask re-computing fitness value by call of the Chromosome.fitness() with the special parameter. After this call, the position of lectures will be (in the first time period) in the chromosome with the error. The process will choose one position randomly and it streams to correct these errors:

- If it exists the error, in which *teaching periods are not concatenated* or *teaching periods are not in one day*. The teaching unit is equipped by the "random time slots" (random is only the first, the others are defined in concatenation with it) by this way, the errors to be eliminated.
- If the error exists, in which *teaching periods have not the same room*, the random room equips the teaching unit with the suited type, and size.

The both these corrections have a local influence only. The corrections can cause collisions in the chromosome. The convergence of the algorithm was dramatically higher, after completing these "correction mutations". Without these corrections it was needed to done more then one half of million generations for the good timetable, after corrections there were necessary only some thousands for the timetable finishing with the same data.

8. SELECTION AND REPRODUCTION

A lot of strategies were used for parents' selection for crossing, and for selection of individuals, which are active in the next generation. The best selection is as follows:

Tournament of parents

The new generation is created by crossing of individuals selected from the original generation by the tournament method. In other words: the two parents are selected for the new individual by the tournament method.

Next strategy is not so efficient and is described only because of its usage in testing runs of algorithm described later in this article:

Tournament of parents + children

Temporary set of individuals is created. Parents of each individual are randomly selected from original gereration. To the next generation are individuals selected by tournament form set of both parents and their children.

9. INFLUENCE OF PARAMETERS FOR GA RUN

The main aim of measurement was to know what parameter influents the tendency to searching and the time of processing. The speed of solving was evaluated by the number of generations needed to creating a "good" timetable. It was considered the cases, when the algorithm did not finish the timetable in the declared number of generation.

All tests were evaluated on the same input data. The data content 13 rooms, 6 lectures (decomposed to 93 time slots), 13 teachers, and 20 groups. The maximal number of generations was 20 000. There were started 10 tests for every combination of parameters. The results in tables are the arithmetic average of results of 10 runs of algorithm.

The average value must not express the algorithm behaviour, when we consider the space of searching space, which are 9.5×10^{277} possible timetables.

10. INFLUENCE OF MUTATION

In the tests was changed mutation probability from 0.0 through 0.5 to 1.0. The size of tournament was 3, the size of population 30, and there were used selection tournament methods parents + children, and tournament parents.

The implementation of mutation is very variable compared to mutation of classic GA. In any cases it has the function of normal mutation (with random change of chromosome part). General property of mutation is its ability to correct any error part of chromosome by the random value (of suited values). The generalisation of mutation is, by the tests, the strongest tool in timetable searching. The probability of mutation is very low value (from 0.01 to 0.2) in classical GA.

The results of testing bring the conclusion, that it is suit to use the much bigger value of probability of mutation. The best results were reached for the value of probability of mutation equal to 1. This is the case, when all individuals are mutated in all generations. The highest probability of mutation makes the number of needed generations dramatically low. It makes the number of generations needed to tendency of fitness = 0, in the same case, it makes higher the necessary time for computation of individuals in one generation.

Table 1: The influence of mutation

	tournament parents + children			tournament parents		
probability of mutation	0.0	0.5	1.0	0.0	0.5	1.0
no finish: fitness > 0	10	0	0	10	0	0
number of generation	20 000	3 813	654	20 000	381	138
time of one generation	11.4	12.0	12.9	13.0	16.6	21.3
time to fitness = 0 [ms]	-	45 756	8 437	-	6 325	2 939

The combination of antagonistic influences gives the best results (lower number of generations). From the tests it is clear, that the higher probability of mutation gives the higher speed of tendency of algorithm, if we consider the both number of generation and total time of computation.

11. SUITED (OPTIMAL) PARAMETER VALUES

The optimal parameter values are considered in three criterions:
Size of population – it is better to start with the low, but important size of population. Thirty individuals are sufficient for deadlock prevent.

Probability of mutation – the higher is better. Of course, the mutation of all individuals is very far from classical GA, the optimal value of probability of mutation 0.8 is chosen.
The number of individuals in a tournament – high number of individuals causes very frequent deadlock in the local minimum. Suited value is 2 or 3, which give the higher selection gradient.

12. CONCLUSION

It was simulated the Genetic Algorithm (GA) is able to solve the problem of timetable effectively. In constrain to generally used GA it was used different implementation of mutation. Of course, this kind of mutation is not generally usable. The mutation is implemented in context of solving problem. The "correcting" mutation changes the values in the place of error location. This activity is not completely random, how it is asked in general GA. The values are chosen randomly, but not from the whole number of possible, but they are chosen from the suited values.

This paradigm is very useful for GA run by the tests. The mutations serve not only for abandoning the situation of deadlocks, but it deals with in searching of space of solutions. The tests show, that frequency of these situations has more serious influence to asked tendency to finishing the goal. It is very good for finishing the solution to use many times higher value of mutation probability, then it is usual. The best results are reached, if it is probability of mutation equal to 1.

It is possible to say, that in the case of suited input data, the genetic algorithm will generate "good" timetable for big schools with the complicated teaching plan.

13. ACKNOWLEDGEMENT

The research is supported by the project GAČR No. 201/00/1531.

REFERENCES

Cooper, T. B., Kingston, J. H. (1995). The Complexity of Timetable Construction Problems. http://www.cs.usyd.edu.au/~jeff/
Kvasnicka, V., Pospichal, J., Tino, P. (2000). *Evolucne algoritmy.* STU Bratislava