I T H E A

International Journal

INFORMATION THEORIES & APPLICATIONS

2009 Volume 16 Number 3

International Journal
INFORMATION THEORIES & APPLICATIONS
Volume 16 / 2009, Number 3

IJ ITA is official publisher of the scientific papers of the members of the ITHEA® International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.

IJ ITA rules for preparing the manuscripts are compulsory.
The rules for the papers for IJ ITA as well as the subscription fees are given on *www.ithea.org* .
The camera-ready copy of the paper should be received by http://ij.ithea.org.
Responsibility for papers published in IJ ITA belongs to authors.

General Sponsor of IJ ITA is the Consortium FOI Bulgaria (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.16, Number 3, 2009

Printed in Bulgaria

Edited by the Institute of Information Theories and Applications FOI ITHEA®, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: ITHEA®
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com

ISSN 1310-0513 (printed) ISSN 1313-0463 (online) ISSN 1313-0498 (CD/DVD)

A GENETIC AND MEMETIC ALGORITHM FOR SOLVING THE UNIVERSITY COURSE TIMETABLE PROBLEM

Velin Kralev

Abstract: In this paper genetic and memetic algorithms as an approach to solving combinational optimization problems are presented. The key terms associated with these algorithms, such as representation, coding and evaluation of the solution, genetic operators for the crossing, mutation and reproduction, stopping criteria and others are described. Two developed algorithms (genetic and memetic) with defined computational complexity for each of them, are presented. These algorithms are used in solving the university course timetable problem. The methodology and the object of study are presented. The main objectives of the planned experiments are formulated. The conditions for conducting experiments are specified. The developed prototype and its functionality are briefly presented. The results are analyzed and appropriate conclusions are formulated. The future trends of work are presented.

Keywords: genetic algorithm, memetic algorithm, university course timetable problem.

1. Introduction

1.1. Genetic Algorithms

Evolutionary algorithms (EA) are based on ideas borrowed from natural processes [1]. These algorithms use a set (called population) of possible solutions (called individuals of the population) which are subject to future changes in order to obtain the optimal solution, according to a predefined optimality criterion.

The idea of evolutionary algorithms was presented for the first time in [2] while the idea of genetic algorithms (GAs) was presented for the first time in [3]. Later in 1992 the beginning and the genetic programming were established [4]. Evolutionary computations date back even earlier [5] where a review of earlier approaches was made.

A particularly efficient subclass of the evolutionary algorithms is a class of the genetic algorithms. In these algorithms, the variable is called a gene or allele, while the solution is called an individual or chromosome. Solutions must be coded in such a way that genetic operators can be applied. An encoded solution is called genotype while a decoded solution is called phenotype. To measure the quality of a solution the term fitness is used i.e. a solution with high fitness is a better solution. The set of all solutions is called population. At each iteration the genetic algorithm creates a new generation. This process is called reproduction.

To solve an optimization problem initially several possible solutions are constructed. These solutions are coded properly according to the problem being solved, but so that the genetic operators can be applied. The genetic algorithms start with a set of initialized population of solutions, which are most often generated randomly. The two basic genetic operators are crossover and mutation. The genetic operator "crossover" most often takes two solutions (parents) and combines them after which one or more new solutions (children) come as a result. The parents are selected from all solutions of the current population. However, the choice of solutions is stochastic and is based on solutions with better value of the fitness function. The genetic operator "mutation" changes one solution and forms a new one [6].

The space of the possible solutions is a set of points where each point presents a possible solution. The searching of solution is equivalent to finding the extremes in this space (minimum or maximum). Since it is not always possible to predict the actual optimum, then the best solution found so far, is often regarded as a near-optimal.

After performing a number of genetic operations "crossover" and "mutation", some of the solutions in the old population with new ones are replaced. Thus the creation of a new population of the algorithm ends. The process of creating new generations is repeated until the stopping criterion is satisfied (eg, creating a specific number of generations or a certain number of generations after which a better solution than the last best has not been found).

Typically, the process of seeking a solution by genetic algorithm is independent from the scope. The genetic operators "crossover" and "mutation" do not know which solution is better and which not. This is determined by the fitness function which evaluates each solution. However, it is shown that good results using this approach can be obtained [7, 8].

1.2. Memetic Algorithms

There is still a debate associated with genetic algorithms a whether the genetic operator "crossover" plays a greater role or the local search process. To solve this dispute, usually the genetic algorithms that use techniques for local search are called mimetic algorithms [9].

The concept of memetic algorithms was presented for the first time by Moscato and Norman [10]. They describe an evolutionary algorithm which uses local search. This idea was later formulated by Radcliffe and Surrey [11] who made a comparison between genetic algorithms and mimetic ones. In the mimetic algorithms "meme" is regarded as a unit of information that can be self-replicating in a way in which people exchange ideas. The "meme" is different from the gene in that when it passes between the individual solutions, any solution adapts "meme" in the best possible way, while the gene remains unchanged in this transition [9].

The main advantage in using the memetic algorithms is that the space of possible solutions is reduced to the subspace of acceptable solutions with local optimum. After the execution of an operation such as mutation in the genetic algorithms the new solution may be worse than the original estimate. In the mimetic algorithms through the use of local the best location for a given gene (meme) will be always found. Thus the best possible cost for a solution will be achieved [9].

2. A Genetic and Memetic Algorithms for Solving the University Course Timetable Problem

In developing the genetic algorithm, the basic techniques used are the crossover and mutation operators. In the mimetic algorithm besides the above-mentioned two techniques the method of local search is used as well. Both algorithms are similar and therefore in this section they will be presented together showing only the differences between them.

2.1. Generating a Solution Based on Constructive Heuristics

The basis of the developed genetic algorithm is generating a solution based on constructive heuristics. The main goal of this algorithm is given in advance events so that they be placed on the weekly schedule. The pseudo-code of this algorithm is presented in Fig. 1.

For each event the algorithm seeks a free timeslot. The aim is to put each event in the weekly schedule so that no violation of soft constraints to occur. The computational complexity of the algorithm is square, i.e. $\Theta\left(N^2\right)$, where N is the number of events.

```
for each event
  if the event is fixed then continue to the next event
  for each timeslot
    current event set the current timeslot
    if not possible then continue to the next timeslot
    if any of the soft constraints is not violated
    then continue to the next event
  end
end
```

Figure 1. Pseudo-code for generating a solution based on constructive heuristics.

2.2. Generating a Solution Based on Local Search

The basis of the developed memetic algorithm is generating a solution based on a local search. The main goal of this algorithm is with given in advance a ordinance of events to be placed on a weekly schedule in such a way so that to obtain the best costs of the generated solution. The pseudo-code of this algorithm is presented in Figure 2.

```
for each event
  if the event is fixed then continue to the next event
  for each timeslot
    current event set the current timeslot
    if not possible then continue to the next timeslot
    if not violated any of the soft constraints
    then calculate the cost of the solution
    and continue to the next timeslot
  end
  put the current event at this timeslot
  in which the cost of the solution was best
end
```

Figure 2. Pseudo-code for generating a solution based on local search.

For each event and each timeslot the algorithm calculates the cost of the generated solution. Once all the timeslots have been verified the algorithm sets the current event at this location for the cost of the solution which was best. The computational complexity of the algorithm is cubic, i.e. $\Theta\left(N^3\right)$, where N is the number of events.

2.3. Generating Initialized Population. Genetic Operators of Selection, Combination, Crossing and Mutation

The initialized population of genetic algorithm is created by constructing q solutions through the algorithm described above which is based on constructive heuristics. You will note that q is the size of the population that remains constant in the process of reproduction.

The initialized population of mimetic algorithm is created similarly. Using the method of constructing the solutions by local search we generate q solutions.

At stage "selection" for both algorithms $\frac{q}{2}$ (i.e. 50%) of solutions in the current population which will play the role of parents are chosen. Then $\frac{q}{4}$ (i.e. 25%) pairs form at random. They will be crossed in order to generate new solutions - children.

The operator of the crossing for both algorithms is the same and plays an essential role. For each pair of parents a crossover operator is applied in the following manner: choose two elements (genes) k and r, such that k < r and k <> r. They represent the intersection of the parents. By rotation of a range of genes $[1,..,k-1]$, $[k,..,r]$ and $[r+1,..,N]$ the permutations of events for the two new children are formed of them. In this way of combining and coding it is likely to have missing and/or repetitive genes in descendants. The problem is solved by the repeated genes which replace the missing ones.

The mutation operators for both algorithms genetic and memetic are similar. Choose randomly two genes k and r which exchanged places. For the algorithms of consideration it was adopted 10% of the children to mutate, i.e. $\left\lceil \frac{q}{2}10\% \right\rceil = \left\lceil \frac{q}{20} \right\rceil$.

2.4. Reproduction Operators of the GA and MA

Reproduction operators of genetic and memetic algorithm are similar and will be considered together. Pseudo-code for these operators is presented in Fig. 3.

```
generate initialize population [genetic or memetic]
for I := 1 to reproduction count do
begin
  delete worst solutions from population
  distribute parents to pairs
  crossover [genetic or memetic]
  mutate [genetic or memetic]
  sort population
end
```

Figure 3. Pseudo-code for reproduction operators of GA and MA.

First an initialized population of genetic and memetic algorithm is generated. Then a sequence of operations is performed until it reaches a predetermined number of reproductions. From each population 50% will be deleted of those solutions that have worse costs. Then the parents are combined in pairs and crossed.

Solutions of the children are generated with the method based on constructive heuristics for the GA and the method based on local search for the MA. At each step, the relevant method is called $\frac{q}{2}$ times as the number of new solutions. The next step: the mutation operators are executed. The reproduction cycle ends with sorting out the solutions with increasing cost.

In general, the complexity of the GA is square, and the MA is cubic. The complexity of the algorithms depends linearly on the number of reproductions (GenerationCount), i.e. the number of generations that will be generated.

3. Experimental Results

Three experiments with the following objectives were made:

1. To verify the efficiency of the prototype.

2. To determine the influence of the size of the input data and of the population size on the execution time of algorithms.

3. To determine the impact of population size and number of reproductions on the quality of the solution of the different sets of input data.

4. To make a comparative analysis between the proposed algorithms by comparing the performance quality of the generated solutions and the execution time for all sets of input data.

5. To demonstrate the effectiveness of the prototype and the proposed algorithms as compared to the estimates of the solutions generated by the prototype and the user (expert).

3.1. Development of Prototype

For the purposes of the experiment a prototype was developed. Through it planned experiments were conducted. From the obtained results appropriate conclusions were made. Algorithms that were discussed in the previous section are implemented in the prototype and can be used to generate schedules. Fig. 4 shows an example session using the prototype.

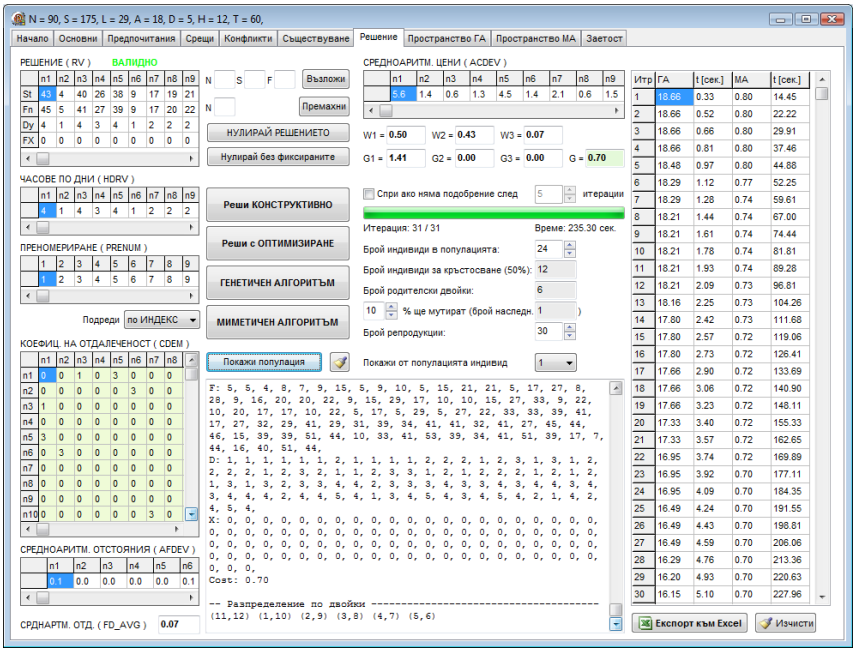


Figure 4. Work session with the prototype.

3.2. Conditions for the Experiment

Experiments were conducted on a PC with 32 bit operating system Windows VistaTM Enterprise (Service Pack 2) and the following hardware configuration:

1) Processor: Intel(R) Core(TM)2 Duo CPU T7500 @2.20GHz 2.20GHz

2) RAM memory: 2.00 GB

3.3. Methodology of the Experiment

For the purposes of the experiment 5 sets of input data were prepared, which are presented in Table 1.

Table 1. Sets of input data used in the experiments.

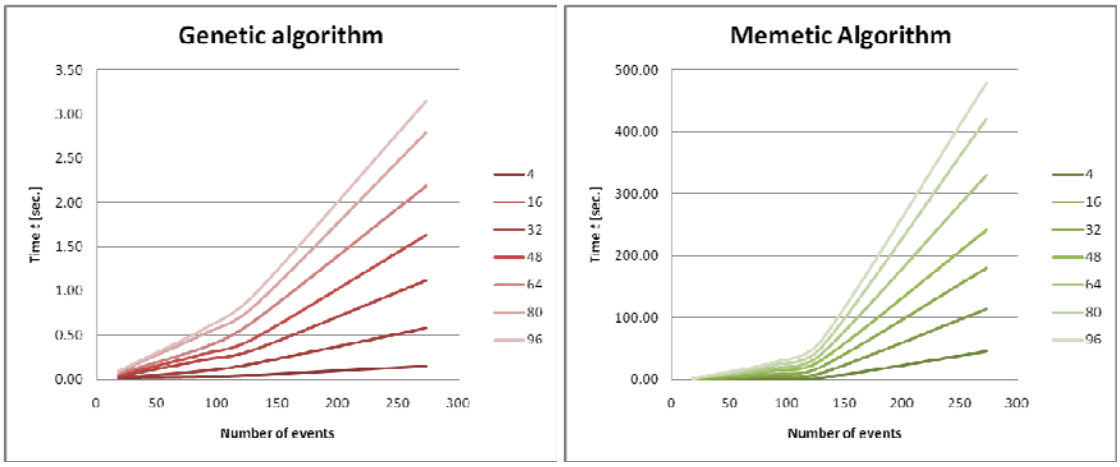
Dataset	DS1	DS2	DS3	DS4	DS5
Events (N)	18	90	130	273	30
Students (S)	52	175	274	549	45
Groups (Gr)	4	14	21	43	-
Lecturers (L)	10	29	37	62	10
Auditoriums (A)	10	18	22	39	10
Total (N+S+L+A)	90	312	463	923	95

The input datasets were chosen with certain reasons. The dataset 1 corresponds to one course. The dataset 2 corresponds to one subject. The dataset 3 is a combination of two subjects which have a common block courses, common lecturers and common auditoriums. The dataset 4 corresponds to approximately half a faculty and is a combination of three subjects sharing common resources. The dataset 5 is a set of 45 students who may not be distributed in different groups of students because they chose different curricula.

3.4. Conclusions from the Experiments

After conducting the experiments the following conclusions were made:

1. With the increase in the number of events, exponentially the execution time for GA and MA increases. This behavior is similar to both algorithms for all values of the parameter q (see Fig. 5a and 5b).

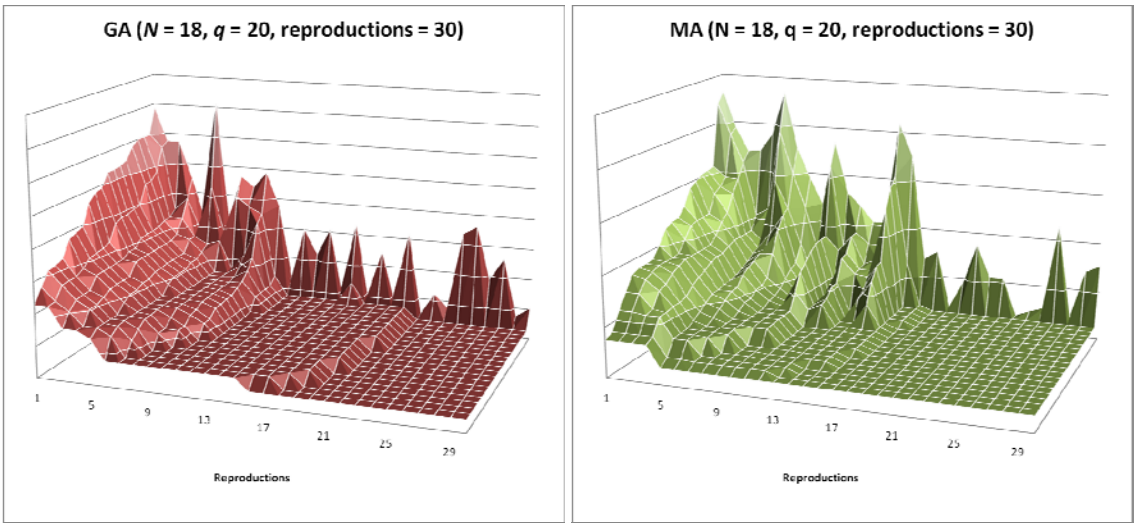


a) b)

Figure 5. Execution time of algorithms with increasing number of events: a) GA, b) MA.

2. With the increase in the number of solutions in the population the execution time for GA and MA increases linearly. This behavior is similar to all tested datasets (1 - 5).

3. With the increase in the number of solutions in the population the quality of solutions for both algorithms, improve. The level of convergence (i.e. overpopulation of the population with similar solutions) occurs differently for each of the algorithms (see Fig. 7a and 7b).



a) b)

Figure 7. Behavior of the GA and MA for a dataset 1 (N = 18, q = 40) a) GA, b) MA.

4. With the increase in the number of solutions in the population, the intensity of improving the quality of solution decreases for both algorithms.

5. In terms of quality of the decisions, MA provides significantly better results for all values of the parameters q and N, compared with GA. In terms of execution time, however, MA runs significantly slower than the GA.

6. GA is able to generate schedules with better cost which are commensurate with the schedules constructed by the user-expert (for the same input data). MA generates solutions with much better estimates than the other two methods, aiming to combine the events in such a way as to obtain a schedule with an optimal cost (see Fig. 8).

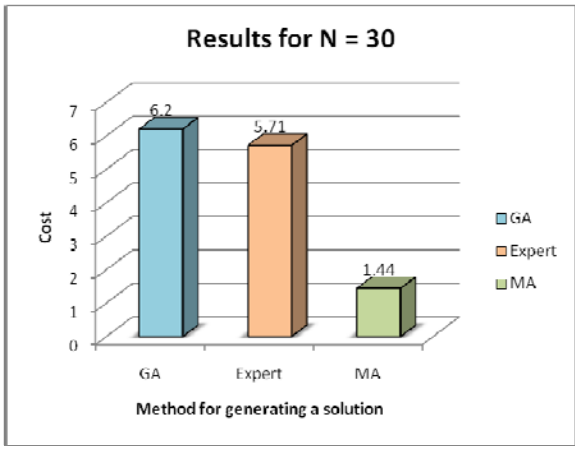
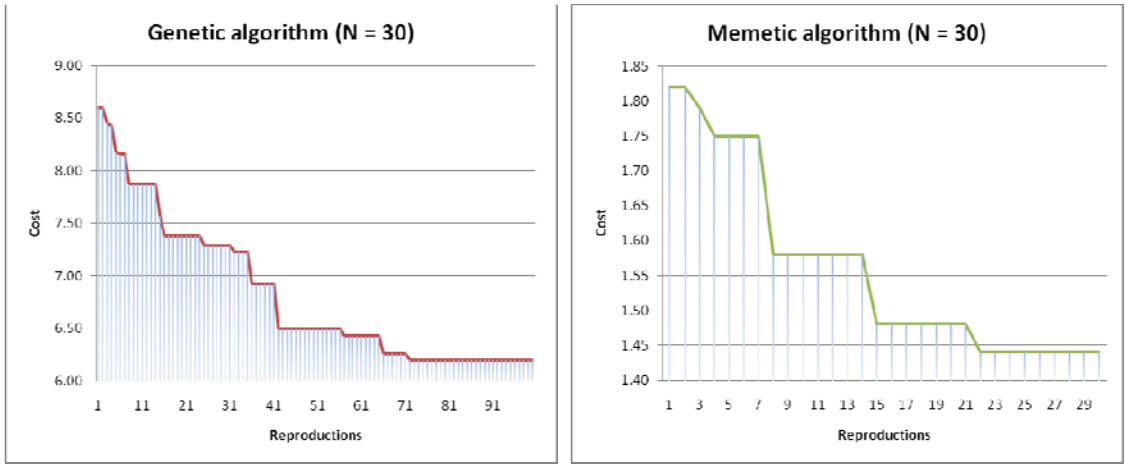


Figure 7. Costs of schedules generated by the GA, user-expert and MA (for a dataset 5 (N = 30)).

In Fig. 8 the behavior of the GA and MA for a set of input data 5 (N = 30) is shown.



а) б)

Figure 8. Behavior of the GA and MA for a dataset 5 (N = 30).

4. Conclusion

In this paper genetic and memetic algorithms as an approach to solving combinational optimization problems are presented. The key terms associated with these algorithms, such as representation, coding and evaluation of the solution, genetic operators for the crossover, mutation and reproduction, stopping criteria and others are described. Two developed algorithms (genetic and memetic) are presented for each of them the computational complexity is defined. These algorithms are used in solving the university course timetable problem. The methodology and the object of study are presented. The main objectives of the planned experiments are formulated. The conditions for conducting experiments are specified. The developed prototype and its functionality are briefly presented. The results are analyzed and appropriate conclusions are formulated. The future trends of work are presented.

Bibliography

1. Fogel D. B. (1992). A brief history of simulated evolution. Proceedings of the First International Conference on Evolutionary Programming, Evolutionary Programming Society, La Jolla, CA.
2. Rechenberg I. (1973) Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart.
3. Holland J. H. (1975) Adaptation in Natural and Artificial Systems. The U. of Michigan Press.
4. Koza J. H. (1992) Genetic programming: on the programming of computers by means of natural selection. Massachusetts Institute of Technology Press.
5. Fogel D. (1998) Evolutionary Computation. IEEE Press.
6. Burke E. K. (1994) A Genetic Algorithm based University Timetabling System. Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, vol 1, pp 35-40.
7. Bruns R. (1993) Knowledge-Augmented Genetic Algorithm for Production Scheduling. IJCAI '93 Workshop on Knowledge based Production Planning, Scheduling and Control.

8. Burke E. K., Elliman D. G. and Weare R. F. (1994) A Genetic Algorithm for University Timetabling", AISB Workshop on Evolutionary Computing, Leeds.
9. Burke E. K., Newall J. P. and Weare R. F. (1996) A memetic algorithm for university exam timetabling. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I). Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag, pp 241-250.
10. Moscato P., Norman G. M. (1992) A "Memetic" approach for the travelling salesman problem – implementation of a computational ecology for combinatorial optimisation on message-passing systems. Proceedings of the International Conference on Parallel computing and Transputer Applications, IOS Press (Amsterdam).
11. Radcliffe N. J., Surry P. D. (1994) Formal Memetic Algorithms. Lecture Notes in Computer Science 865 (Evolutionary Computing) Springer-Verlag, pp 250–263.

Author's Information

Velin Kralev – Department of Informatics, Faculty of Mathematics and Natural Sciences, South-West University "Neofit Rilski"-Blagoevgrad, Bulgaria; e-mail: velin_kralev@abv.bg