Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st July 2011. Vol. 29 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HYBRID GENETIC ALGORITHMS WITH SIMULATING ANNEALING FOR UNIVERSITY COURSE TIMETABLING PROBLEMS

NABEEL R. AL-MILLI
Nabeel.almilli@bau.edu.jo
Financial and Business Administration and Computer Science Department
Zarqa University College
Al-Balqa' Applied University

ABSTRACT

The Educational Timetabling problem deals with the assignment of course (or lecture events) to a limited set of specific timeslots and rooms, subject to a variety of hard and soft constraints. All hard constraints must be satisfied, obtaining a feasible solution. In this paper we establish a new hybrid algorithm to solve course timetabling problem based on Genetic Algorithm and Simulating Annealing algorithm. We perform a hybrdised method on standard benchmark course timetable problems and able to produce promising results.

Keywords: *Course Timetabling, Genetic Algorithms, Simulating Annealing*

1. INTRODUCTION

University Course Timetabling Problems (UCTPs) is an NP-hard problem, which is very difficult to solve by conventional methods and the amount of computation required to find optimal solution increases exponentially with problem size. The main idea of this problem is to assign a set of lectures to rooms and time periods satisfying a number of constraints. The set of constraints are usually divided in two sets: hard constraints and soft constraints. Hard constraints have a higher priority than soft. The objective of this problem is to satisfy the hard constraints and to minimise the violation of the soft constraints. It is therefore necessary to use efficient search methods to produce optimal or near optimal timetable that satisfy the constraints.

A large number of diverse methods have been already proposed in the literature for solving timetabling problems. These methods come from a number of scientific disciplines like Operations Research, Artificial Intelligence, and Computational Intelligence [1], [2], [3], [4], [5], [6] and can be divided into four categories:
Sequential Methods, that deals timetabling problems as graph problems. Generally, they order the events using domain-specific heuristics and then assign the events sequentially into valid time slots in such a way that no constraints are violated for each timeslot [7].
Constraint Based Methods, according to which a timetabling problem is modeled as a set of variables (events) to which values (resources such as teachers and rooms) have to be assigned in order to satisfy a number of hard and soft constraints [8].
Cluster Methods, in which the problem is divided into a number of events sets. Each set is defined so that it satisfies all hard constraints. Then, the sets are assigned to real time slots to satisfy the soft constraints as well [9].
Meta-heuristic methods, such as genetic algorithms (GAs), simulated annealing, tabu search, and other heuristic approaches, that are mostly inspired from nature, and apply nature-like processes to solutions or populations of solutions, in order to evolve them towards optimality [1], [3], [4], [10], [11], [13],[14].

Since then, the literature has hosted a large number of papers presenting evolutionary methods and applications on such problems with significant success [12].

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st July 2011. Vol. 29 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The paper is organised as follows, the next section introduces the university course timetable problem with a set of all hard and soft constraints. In section 3 we represent the main concepts about Genetic algorithm. Section 4 introduces the great deluge algorithm. Hybridization between genetic algorithms and great deluge are represented in section 5. The simulation results are represented in section 6, and finally conclusion and future work are represented in section 7.

2. PROBLEM DESCRIPTION

The general timetable problem can be expressed in the following way: a number of events must be timetabled by associating them with timeslots. In university course timetable, a set of events (courses) is scheduled into a fixed number of rooms and timeslots within a week. In this paper, we test our method on the problem instances introduced by Socha et al [13]. The problem presents a set of N courses to be scheduled in 5 days of 9 periods each, which time T = 45 timeslots, a set R rooms (each room have a set of F features and capacity), a set of M students and a set of features required by courses. Each student attends a subset of courses. Solutions in which all courses are assigned to periods and rooms and satisfy all hard constraints are called feasible solutions. The hard constraints considered for this problem are:

1) No student can be assigned to more than one course at the same time.
2) The room should satisfy the features required by the course.
3) The number of students attending the course should be less than or equal to the capacity of the room.
4) No more than one course is allowed at a timeslot in each room.

The soft constraints considered for this problem are:

1) A student has to attend only one course in a day.
2) A student has to attend more than two courses consecutively.
3) A student has to attend a course in last period in any day.

3. GENETIC ALGORITHMS

GA is the most famous among EA algorithms. GAs have been employed as a tool that can handle multi-model function and complex search space. They have the capability to search complex spaces with high probability of success in finding the points of minimum or maximum on the search space (i.e. landscape). Genetic Algorithms (GAs) are derivative-free stochastic search algorithms. GAs applies the concept of natural selection. This idea was first introduced by John Holland at the University of Michigan in 1975 [1]. GAs have been successful used in solving numerous applications in engineering and computer science [12, 13, 14, 15]. GA gains a great popularity due to their known attributes. These attributes include:

- GAs can handle both continuous and discrete optimization problems. They require no derivative information about the fitness criterion [16, 17].

- GA has the advantageous over other search algorithm since it is less likely to be trapped by local minimum.

- GA provide a more optimal and global solution. They are less likely to be trapped by local optimal like Newton or gradient descent methods [18, 19].

- GA has been shown to be less sensitive to the presence of noise and uncertainty in measurements [5, 20].

- GAs use probabilistic operators (i.e. crossover and mutation) not deterministic ones.

Genetic algorithms code the candidate solutions of an optimization algorithm as a string of characters which are usually binary digits [23]. In accordance with the terminology that is borrowed from the field of genetics, this bit string is usually called a chromosome (i.e. individuals). A number of chromosomes generate what is called a population. The structure for each individual can be represented as follows:

$gene_1$	$gene_2$	$\dots\dots$	$gene_n$
11101	00101	$\dots\dots$	11011

This chromosome has number of genes equal to n. These genes are used in the evaluation function f. Thus, f(gene1, gene2, . . . , genen) is the function to be minimized or maximized.

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st July 2011. Vol. 29 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A. Evolutionary Process

The evolutionary process of GAs starts by the computation of the fitness of the each individual in the initial population. While stopping criterion is not yet reached we do the following:

- Select individual for reproduction using some selection mechanisms (i.e. tournament, rank, etc).
- Create an offspring using crossover and mutation operators. The probability of crossover and mutation is selected based on the application.
- Compute the new generation of GAs. This process will end either when the optimal solution is found or the maximum number of generations is reached.
- A flowchart for a simple GA process is given [21] in Figure 1

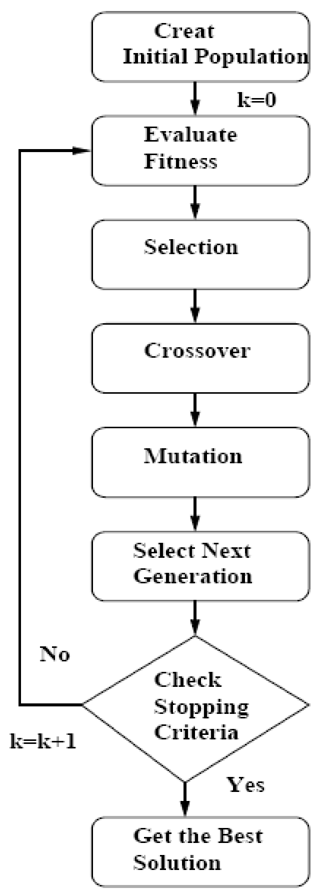

Fig. 1. Flowchart of a simple GAs process

B. Selection Mechanism

Selection is the process which guides the evolutionary algorithm to the optimal solution by preferring chromosomes with high fitness. The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated, using some measure of fitness. To create the next generation, new chromosomes, called offspring, are formulated by using some operators called crossover and mutation. Thus, a new generation will be created by selecting the best chromosomes (parents) from the previous generation and the best chromosomes from the offspring [22].
After several generations of creation the algorithm hopefully converges to the optimal solution or at least the optimal domain of solution. After computing the fitness of each individual, a new population must be created. To do this, two operators borrowed from natural genetic, crossover and mutations, are used [16, 17]. Crossover operator is used to produce new pairs of individuals from their parents. The produced individuals (i.e. childes) have many features from their parents. There is a high probability that the child's will provide a better fit to the problem.

C. Crossover Mechanism

Crossover is the main genetic operator. In [1] Holland indicates that crossover provides the main search operator while bit mutation simply serves as a background operator to ensure that all possible solutions can enter the population. The probabilities commonly assigned to crossover and bit mutation reflect this philosophical view. It operates on two chromosomes at a time and generates offsprings by combining both chromosomes' features.

One way to do crossover is to choose a random cut-point and generate the offspring by combining the segment of one parent to the left of the cutpoint with the segment of the other parents to the right of the cut-point. This type of crossover operates with the bit string representations. Single point crossover of two binary string chromosomes is presented in Figure 2.

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st July 2011. Vol. 29 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

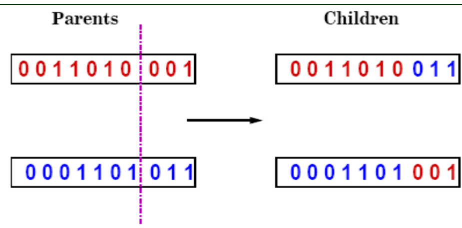

Fig. 2. Single-point crossover of two binary string chromosomes

For other types of representation other crossover types are suggested. Syswerda [23] conducted function optimization experiments with number of mutation mechanism. They include uniform crossover, two-point crossover and one-point crossover. He found that uniform crossover can provide better solutions with less computational effort.

D. Mutation Mechanism

Mutation is a background operator which produces spontaneous random changes in various chromosomes.
In genetic algorithms, mutation serves the role of either replacing the genes lost from the population during the selection process so that they can be tried in a new form or providing for genes that were not present in the initial population. One way to do mutation would be to alter one or more genes. In Figure 3, we show binary string chromosomes mutation.

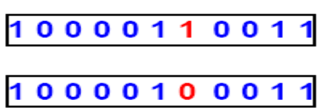

Fig. 3. Mutation of binary string chromosomes

GA evaluates the individuals in the population using a selected fitness function (criterion). This function indicates how good or bad a candidate solution is. The way to select the fitness function is a very important issue in the design of genetic algorithms, since the solution of the optimization problem and the performance of the algorithm count mainly on this function.

It is important to recognize that GAs is different from other optimization techniques like gradient descent, since they evaluate a set of solution in the population at each generation, makes them more likely to find the optimum solution.

The fitness of the individuals within the population is evaluated, and new individuals are generated for the next generation using a selection mechanism. Although convergence to a global optimum is not guaranteed in many cases, these population-based approaches are much less likely to converge to local optimal and are quite robust in the presence of noise [16, 17].

4. SIMULATING ANNEALING ALGORITHM

Simulating annealing is considered as a randomized algorithm that tries to avoid from being trapped in local optimum by assigning probabilities to deteriorating moves (Aarts and Korst 1988). A threshold value is chosen in simulating annealing. The decrease in the cost of two moves is compared with a threshold value. If the difference is less than the threshold value, then the new solution is accepted. A higher threshold value may be chosen to explore various parts of the solution space while a lower threshold value may be chosen to guide the search toward good solution value. The threshold value is redefined in each generation in order to enhance both diversification and intensification (Aarts and Korst 1988). Starting with a higher value for threshold value and then decreasing the value may result in finding good solutions. Also a simulating annealing algorithm uses a threshold value as a random variable. Figure 2.4 shows a generic simulating annealing for a minimization problem.

Algorithm *Simulating annealing*
begin
  s:=*initial solution*;
  $k = 1$;
  repeat
    generate an $s' \varepsilon N(s)$;
    if $f(s') \le f(s)$ **then** $s = s'$;
  else
    if $exp\frac{f(s')-f(s)}{c_k} > random[0,1)$ **then** $s = s'$;
    $k = k + 1$;
  until *stopping criterion*;
end;

Fig. 4: The pseudo code for Simulating Annealing algorithm

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st July 2011. Vol. 29 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. HYBRID GENETIC ALGORITHMS WITH GREAT DELUGE

Figure 5 shows a pictorial illustration for the hybridization between genetic algorithms and simulating annealing, as shown, the sequence of our proposed method start from initial solutions. Applying genetic algorithms as a second step, finally applying simulating annealing mechanism to enhance the quality of solution. Great deluge is consider one of the powerful local optimization algorithm while GA is one of the best known methods of global optimization, combining both algorithms means combine the advantages of both algorithms.

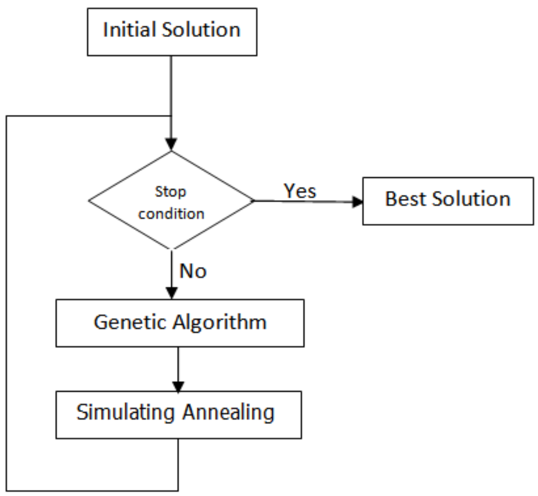

Fig.5: Pictorial Diagram for Hybrid GA with SA

6. SIMULATION RESULTS

The proposed algorithm was programmed using C++ and simulations were performed on the Intel Pentium 4 2.33 GHz computer and tested on a standard benchmark course timetable problem as presented in SectionII. Table I shows the parameter for the GA algorithm chosen after some preliminary experiments, and almost similar with the papers in the literature [16], [19].

TABLE I
PARAMETER SETTING FOR EM ALGORITHM

Parameter	**Value**
Generation Number	100000
Population size	50
Crossover Rate	0.6
Mutation Rate	0.06
Selection Method	Roulette Wheel selection
Crossover Type	Single point

Our algorithm is capable to find a feasible timetable for all eleven cases. Table II shows the results obtained and the comparison with other approaches in the literature which they are:

- M1: The extended great deluge by McMullan (2007).
- M2: The non linear great deluge by Linda-Silva and Obit (2008).
- M3: The hybrid evolutionary approach by Abdullah et al. (2007). They used a randomised iterative improvement algorithm as a local search with a mutation operator.
- M4: The genetic algorithm and local search by Abdullah and Turabieh (2008). They tested a genetic algorithm with a repair function and local search on course timetabling problems.

From Table II, it is clear that the result of our proposed algorithm produces acceptable timetables.

TABLE II
Results

Instance	**our result**	**M1**	**M2**	**M3**	**M4**
Small1	**0**	**0**	3	**0**	**0**
Small2	**0**	**0**	4	**0**	**0**
Small3	**0**	**0**	6	**0**	**0**
Small4	**0**	**0**	6	**0**	**0**
Small5	**0**	**0**	**0**	**0**	**0**
Medium1	89	80	140	221	254
Medium2	**100**	105	130	147	258
Medium3	120	139	189	246	251
Medium4	**77**	88	112	165	321
Medium5	99	88	141	130	276
Large	677	730	876	**529**	1026

7. CONCLUSION

In this paper, we employed Genetic Algorithm (GA) and simulating annealing for course timetable problem local search. Even though the experiments carried out in this work demonstrate that the method presented here only obtains two best result, However the proposed method can produce a feasible and good quality timetable Moreover, it provides results that are consistently good across the all the benchmark problems.

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st July 2011. Vol. 29 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFRENCES:

[1] D. Abramson, "Constructing school timetables using simulated annealing: sequential and parallel algorithms" Management Science, vol. 37, pp. 98–113, 1991.

[2] E. K. Burke and J. P. Newall, "A new adaptive heuristic framework for examination timetabling problems," in Technical Report NOTTCSTR- 2001-5 (submitted to Annals of Operations Research), University of Nottingham, UK, School of Computer Science & IT, 2002.

[3] A. Hertz, "Tabu search for large scale timetabling problems," European journal of operations research, vol. 54, pp. 39–47, 1991.

[4] B. Paechter, M. G. Norman, and H. Luchian, "Extensions to a memetic timetabling system,," in E.K. Burke and P.M. Ross, eds., Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling, Lawrence Erlbaum Associates, 1995.

[5] A. Schaerf, "A survey of automated timetabling," Artificial Intelligence Review, vol. 13 (2), pp. 87–127, 1993.

[6] A. Tripathy, "A lagrangian relaxation approach to course timetabling," Journal of the Operational Research Society, vol. 31, pp. 599 603, 1980.

[7] M. W. Carter, "A survey of practical applications of examination timetabling algorithms," Operations Research, vol. 34, pp. 193–202, 1986.

[8] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," European Journal of Operational Research, vol. 119, pp. 557–581, 1999.

[9] G. M. White and P. W. Chan, "Towards the construction of optimal examination timetables," INFOR 17, pp. 219–229, 1979.

[10] P. Adamidis and P. Arapakis, "Evolutionary algorithms in lecture timetabling," in Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC 99), pp. 1145–1151, IEEE, 1999.

[11] A. Colorni, M. Dorigo, and V. Maniezzo, "Genetic algorithms - a new approach to the timetable problem," in Lecture Notes in Computer Science - NATO ASI Series, vol. F 82, pp. 235–239, Combinatorial Optimization, (Akgul et al eds), Springer-Verlag, 1990.

[12] M. W. Carter and G. Laporte, "Recent developments in practical course timetabling," in In: Burke, E., Carter, M. (Eds.), The Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd Int'l Conf. on the Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science Series, vol. 1408, pp. 3–19, 1990.

[13] M. Chiarandini, K. Socha, and O. R. Doria, "An effective hybrid approach for the university course timetabling problem," 2002.

[14] S. Abdullah, E. K. Burke, and B. McCollum, "An investigation of variable neighbourhood search for university course timetabling," in The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), New York, USA, July 18th-21st, pp. 413–427, 2005.

[15] S. Abdullah, E. K. Burke, and B. McCollum, Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling. In Metaheuristics: Progress in complex systems optimization (Operations Research / Computer Science Interfaces Series), chapter 8. published by Springer, ISBN:978-0-387-71919-1, 2007.

[16] S.I , Birbil :Stochastic global optimization techniques. PhD thesis, North Carolina State University (2002).

[17] S.I , Birbil, S.-C. Fang : An electromagnetism-like mechanism for global optimization. Journal of Global Optimization 25, 263–282 (2003)

[18] S.I , Birbil, S.-C. Fang, R.-L. Sheu: On the convergence of a population-based global optimization algorithm. Journal of Global Optimization 30, 301–318 (2004).

[19] G. Dueck: New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel. Journal of Computational Physics 104, 86-92. 1993.

[20] D. Landa-Silva, J. H Obit: Great Deluge with Non-linear Decay Rate for Solving Course Timetabling Problem. In the fourth international IEEE conference on Intelligent Systems. Varna, Bulgaria, September 6-8, 2008.

[21] S. Abdullah, H. Turabieh, Generating University Course Timetable Using Genetic Algorithms and Local Search. In the Third 2008 International Conference on Convergence and Hybrid Information Technology ICCIT, vol. I, 254-260, 2008.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[22] Paul McMullan: An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, Computational Science – ICCS, Part I, LNCS 4487, 538–545, Springer-Verlag Berlin Heidelberg 2007.
[23] J. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.
[24] K. A. De Jong, Analysis of Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.
[25] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. New York: Addison-Wesley, 1989.
[26] K. Kristinsson and G. Dumont, “Genetic algorithms in system identification,” in Third IEEE International Symposium Intelligent Control, pp. 597–602, IEEE Press, 1988.
[27] A. Chipperfield, P. J. Fleming, and C. Fonseca, “Genetic algorithms tools for control systems engineering,” in Proceedings of First International Conference on Adaptive Computing in Engineering Design and Control, pp. 128–133, UK, 1994.
[28] A. Sheta and K. De Jong, “Time-series forecasting using GA-tuned radial basis functions,” in special issue of the Information Scienec Journal, pp. 221–228, 2001.
[29] A. Sheta and K. DeJong, “Parameter estimation of nonlinear systems in noisy environment using genetic algorithms,” in Proceedings of the IEEE International Symposium on Intelligent Control (ISIC’96), pp. 360–366, 1996.
[30] H. Al-Duwaish and W. Naeem, “Nonlinear model predictive control of hammerstein and winner model using genetic algorithms,” in Proceedings of the IEEE International Conference on Control Applications, pp. 465–469, 2001.
[31] M. Gen and R. Cheng, Genetic Algorithms and Engineering Design. New York: Jonh Wiley and Sons, Inc, 1997.
[32] G. Syswerda, “Uniform crossover in genetic algorithms,” in Proceedings of the Third International Conference on Genetic Algorithms, pp. 2–9, 1989.

AUTHOR PROFILES:

Nabeel R. AL-Milli received the B.S. in Computer Science and Computer Information Systems from Philadelphia University and M.S degree in Computer Science from Al-Balqa Applied University in 2003 and 2006, respectively. He worked as a developer in ESKADENIA software solutions – Amman. He was an administrator and developer at Free zones Corporation, currently he is a lecturer in Financial and Business Administration and Computer Science Department, Zarqa University College, Al-Balqa' Applied University. His research interests include Artificial Intelligence, Evolutionary Computation and Image Processing.

.