ISSN: 2321-7782 (Online)
Volume 3, Issue 3, March 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Comparing Genetic Algorithm with MEME–Timetable Generation

Asif Ansari[1]
PG Scholar,
Alamuri Ratnamala Institute of Engineering & Technology,
Mumbai, India

Sachin Bojewar[2]
Associate. Professor,
Vidyalankar Institute of Engineering & Technology,
Mumbai, India

Abstract: This document proposes an optimized technique to automate time table generation system. Time table generation system involves various challenging constraints of resources including faculties, rooms, time slots etc. The proposed technique filters out the best of active rules and Genetic algorithm to generate the optimized solution. Genetic Algorithm. In genetic algorithm every individual are characterized by a fitness function. After analysis if there is higher fitness then it means better solution and then after based on their fitness, parents are selected to reproduce offspring for a new generation where fitter individuals have more chance to reproduce. The objective of the work is to create a model used to generate the acceptable schedule using probabilistic operators.

Keywords: Genetic Algorithm, fitness function, Timetable Generator, MEME algorithm.

I. INTRODUCTION : GENETIC ALGORITHM

Genetic algorithms (GAs) are a subclass of evolutionary algorithms introduced by John Holland at The University of Michigan based on Darwin's theory of natural selection where the elements of the search space G are binary strings ($G = B^*$) or arrays of other Elementary types. GA is a stochastic global search method that simulates the metaphor of natural biological evolution. GA operates on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators (crossover and mutation) borrowed from natural genetics. This process leads to the evolution of population of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. GA is used to perform global exploration among a population while heuristic methods are used to perform local exploitation around the chromosomes. The behaviors of GA are characterized by the balance between exploitation and exploration in the search space.

The balance is strongly affected by the strategy parameters such as population size, maximum generation, crossover probability and mutation probability. In an iteration or generation in genetic algorithm, the selection, crossover and the mutation only operator operates on the population to produce new population that constitute the new generation.

An outline of the algorithm is given below:

1 Start: Randomly generate a population of N chromosomes.
2 Fitness: Calculate the fitness of all chromosomes.
3 Create a new population:
  » Selection: According to the selection method implemented, select 2 chromosomes from the population.
  » Crossover: Perform crossover on the 2 chromosomes selected.
  » Mutation: Perform mutation on the chromosomes obtained.
4 Replace: Replace the current population with the new population.

5 Test: Test whether the termination condition is satisfied. If so, stop. If not, return the best solution in current population and go to Step 2.

Planning timetable is one of the most complex and error prone application. There are still serious problems like generation of high cost time table are occurring while scheduling and these problems are repeating frequently. [6] Therefore there is a great requirement for an application distributing the course evenly and without collisions. The aim is here to develop a simple, easily understandable, efficient and portable application which could automatically generate good quality time table with in a second. [10] The outline of this paper is as follows: Active rules are described for the knowledge of intelligent agents (i.e. Constraints), GAs are described and their use in optimizing rule based agent is proposed, methods are apply to the problem of optimizing some results of this application are presented and finally, some conclusion and possible direction for future research are presented.

II. Structure of The Automated Time Table Generator

The structure of time table generator consist Input Date Module, relation between the input data module, time interval, time slots module, applying active rules and GA module then extract the reports.

a) Input Data [9]

The input data module is described by a type of data from the database. The data contains:

1) Person: Data describe the name of lecturers.

2) Subject: Data describe the name of the courses in the class.

3) Room: Data describe the name of the classes and capacity of each it.

4) Time Interval: It is a time slot with a starting time and duration.

b) Constraints

Constraints can be divided in to three parts:

1 Validity violation constraints: [10] There are the constraints which are needed to be incorporated necessarily otherwise there is no guarantee of valid time tables generated. They are included as a part of initial generation of population, as they cannot be violated.

» Lab lectures are constrained to appear together. If scheduled lab lectures are odd in number, then last three lectures appear together.

» There are certain lectures that may or may not appear at the same time in more than one class.

» The most trivial violation constraint is that a teacher must not clash in two different tables of a Time table.

2 Hard constraints: [7] Hard constraints are the ones which need to be fulfilled necessarily.

» Classrooms must not be double booked.

» Every class must be scheduled exactly once.

» Classes of students must not have two bookings simultaneously.

» A classroom must be large enough to hold each class booked to it.

» Lecturers must not be double booked.

» A lecturer must not be booked when he/she is unavailable.

3 Soft constraints: [9] these are constraints that are not that obvious but still demanding. They not to be really satisfied but the solutions are generally considered good if large numbers of them are taken care.

» No consecutive lectures of the same teacher in a class.

» No consecutive lectures of the same teacher in a class

» Preference to first lecture.

» Courses must be evenly distributed

» Same teacher must not have consecutive periods unless specified.

Figure (1) The Structure of the e-learning timetable Generator [9]

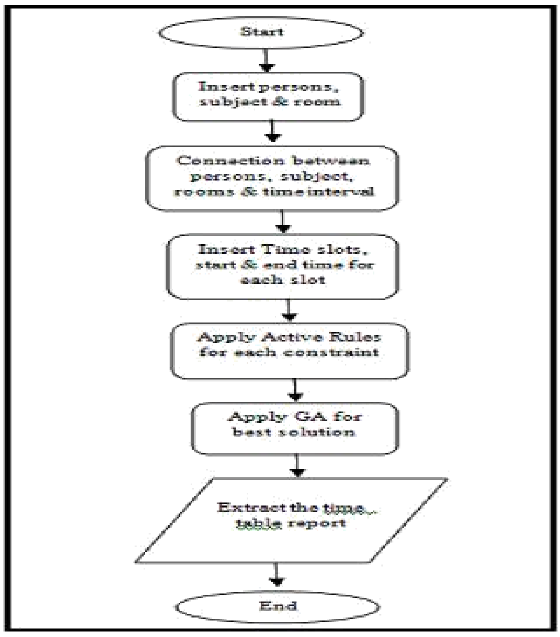
Start
Insert persons, subject & room
Connection between persons, subject, rooms & time interval
Insert Time slots, start & end time for each slot
Apply Active Rules for each constraint
Apply GA for best solution
Extract the time table report
End

Figure 1: Structure of e- Learning

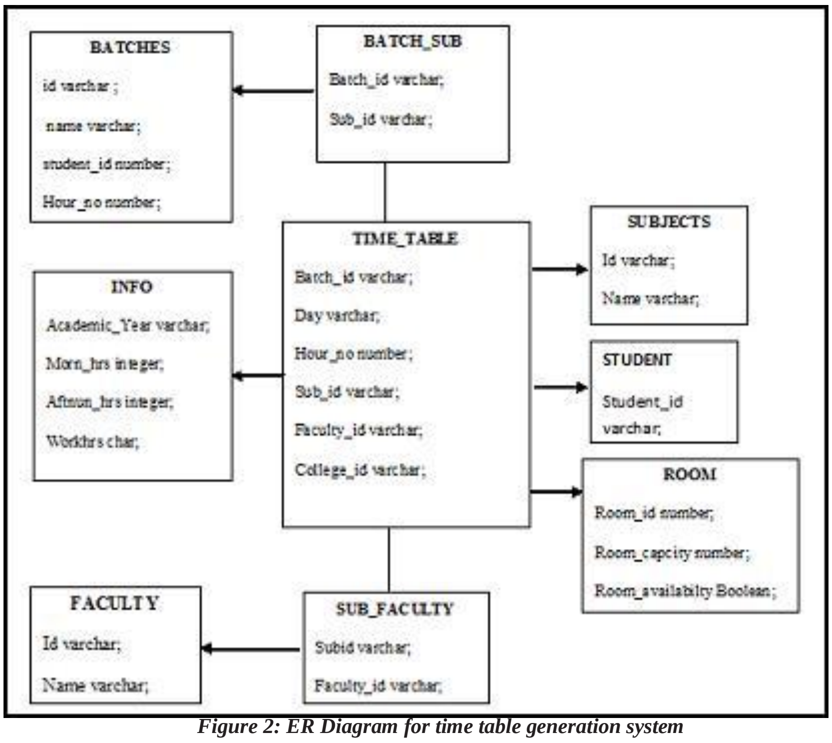

Figure 2: ER Diagram for time table generation system

III. Genetic Algorithm [4]

Genetic algorithms are methods of solving problems based upon an abstraction of the process of Natural Selection. They attempt to mimic nature by evolving solutions to problems rather than designing them. Genetic algorithms work by analogy with Natural Selection as follows. First, a population pool of chromosomes is maintained. The chromosomes are strings of symbols or numbers. There is good precedence for this since humans are defined in DNA using a four-symbol alphabet. The chromosomes are also called the genotype (the coding of the solution), as opposed to the phenotype (the solution itself). In the Genetic algorithm, a pool of chromosomes is maintained, which are strings. These chromosomes must be evaluated for fitness. Poor solutions are purged and small changes are made to existing solutions and then allow "natural selection" to take its course, evolving the gene pool so that steadily better solutions are discovered.

The basic outline of a Genetic Algorithm is as follows: [8]

Initialize pool randomly For each generation

{

Select good solutions to breed new population Create new solutions from parents Evaluate new solutions for fitness Replace old population with new ones

}

IV. Memetic Algorithm

After the development, analyzes the efficiency of a range query over the data that is encrypted by EOB where the proposed OB is used. The main focus was to analyze the searching efficiency in terms of the false positive rate. To do this, the probability distribution of the rate of the width of a bucket to the size of the plaintext space was first analyzed to show that the width of a bucket is not skewed to be extremely large or small. This even-bucket-width property gives the proposed scheme a good querying performance on average. In the proposed OB, the p _ 1 points are randomly uniformly sampled in the plaintext space [0,|M|-1].The width of the *i*th bucket was determined by the selected points of *(i-1)*th order and *i*th order because the width is the difference of these two points. Therefore, to analyze the width of a bucket, it is important to analyze the probability distribution of the position of the selected points.

MEMETIC ALGORITHM

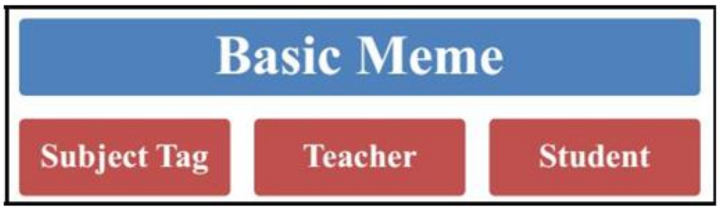

Figure 3: MEME

Genetic algorithm is a general purpose optimization tool based on Darwin's theory of evolution. It has the capability of producing optimized solutions even when the dimensions of the problem increase and for this reason it has been successfully applied to a wide variety of problems.

Genetic algorithm operates on a population of solutions represented by some coding. Each member of the population consists of a number of genes, each of which is a unit of information. New solutions are obtained by combining genes from different population members (crossover) to produce offspring or by altering existing members of the population (mutation). A simulation of 'natural selection' then takes place by first evaluating the quality of each solution and then selecting the fittest ones to survive to the next generation Memetic algorithm (MA) is motivated by Dawkins's notion of a meme as a unit of information that reproduces itself as people exchange ideas [2]. A key difference exists between genes and memes. Before a meme is passed on, it is typically adapted by the person who transmits it as that person thinks, understands and processes the meme, whereas genes get passed on whole. Moscato and Norman linked this thinking to local refinement, and therefore promoted the term mimetic algorithm to describe genetic algorithms that use local search heavily [3]. Radcliffe and Surry gave a formal description of mimetic algorithms [4], which provided a homogeneous formal framework for considering mimetic and GA. According to Radcliffe and Surry, if a local optimizer is added to a GA and applied to every child before it is inserted into the population, then a mimetic algorithm can be thought of simply as a special kind of genetic search over the subspace of local optima. Recombination and mutation will usually produce solutions that are outside this space of local optima, but a local optimizer can then repair such solutions to produce final children that lie within this subspace.

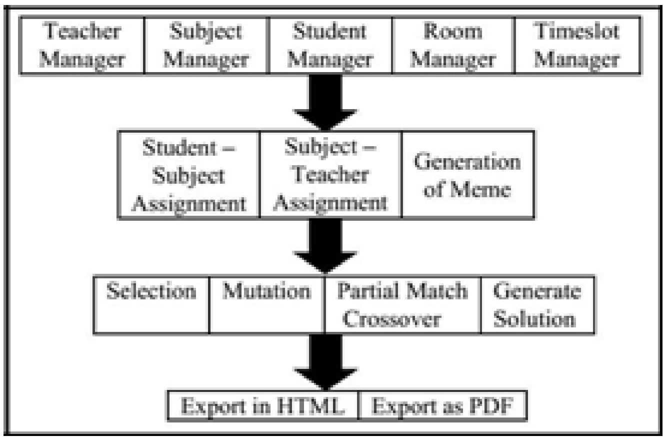

Figure 4: Outline View

An outline of mimetic algorithm is given below:

1 Start: Randomly generate a population of N chromosomes.

2 Fitness: Calculate the fitness of all chromosomes.

» Create a new population:

» Selection: According to the selection method, select 2 chromosomes from the population.

» Local search: search for the best chromosomes

» Crossover: Perform crossover on the 2 algorithm is better suited to handling the NP hard chromosomes selected.

» Local search: search for the best chromosomes

» Mutation: Perform mutation on the chromosomes obtained with small probability.

3 Replace: Replace the current population with the new population.

4 Test: Test whether the termination condition is satisfied. If so, stop. If not, return the best solution in current population and go to Step 2.

V. Conclusion

The two algorithms (genetic and mimetic) converge to the optimal results with 100% accuracy when the population size is about 2.5 times the number of items available in the knapsack. When the population size is greater than 2.5 times the number of items, the accuracy of the optimal results obtained becomes stable at the global optimum but this requires more iterations since the size of the search space will increase for both algorithms.

Also, a major difference between the algorithms is that the time it takes to complete iteration in mimetic algorithm is much more than the time it takes genetic algorithm to complete the same iteration. This is because of the local search algorithm that is embedded in mimetic algorithm

However, it is safe to state that mimetic algorithm is more efficient and better than genetic algorithm since it is guaranteed to converge to an optimal result within a reasonable amount of iterations regardless of the initial population size.

Two Evolutionary Algorithm techniques, that is, Genetic Algorithm and Mimetic Algorithm have been applied to solve knapsack problem. Their performances were measured against metrics such as the optimal value of the objective functions, convergence rate, and accuracy. Results of the experiments show that Roulette wheel selection outperforms the Ranking and Scaling selection by for GA and MA respectively by 4.1% in terms of the accuracy of the optimal result. The results also show that memetic algorithm is more efficient than the genetic algorithm in terms of accuracyand the optimal value of objective function. Consequently, memetic problems.

References

1. P.B Shola (2003), Logic and Program Verification and Development, pp 82.
2. B´elanger, N., Desaulniers, G., Soumis, F. & Desrosiers, J. (2006). Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues, European Journal of Operational Research, 175(3), 1754-1766.
3. B´elanger, N., Desaulniers, G., Soumis, F., Desrosiers, J. & Lavigne, J. (2006). Weekly airline fleet assignment with homogeneity, Transportation Research Part B: Methodological, 40(4), 306-318.
4. Stojkovic, M., & Soumis, F. (2001). An OptimizationModel for the Simultaneous Operational Flight and Pilot Scheduling Problem, Management Science, 47(9), 1290-1305
5. Darrell, W, A Genetic Algorithm Tutorial, Computer Science Department, Colorado State University.
6. Deitel P.J. and H.M. (2007), JAVA How to Program, Seventh Edition. Pearson International Edition.
7. Edemenang E. and Ahmed M. (1994), “Algorithm Design Techniques: Case Study the Knapsack Problem”. The Journal of Computer Science and its Applications (JCSA). 5(1):57-65.
8. Francis A. (2007), Charles Darwin and the Origin of Species, pp 2-94.
9. Hristakeva M. and Shrestha D., Solving the 0-1 Knapsack Problem with Genetic Algorithms Computer Science Department, Simpson College
10. Kim, K. H. & Kim, H. (1998). The optimal determination of the space requirement & the
11. J. J. Grefenstette, editor. Proceedings of the First International Conference on Genetic Algorithms and their Applications.Practice and Theory of Automated Timetabling VI Proceedings of The 6th International Conference on the Practice and Theory of Automa.
12. J. J. Grefenstette, editor. Proceedings of the Second International Conference on Genetic Algorithms and their Applications. Practice and Theory of Automated Timetabling VI Proceedings of the 6th International Conference on the Practice and Theory of Auto.
13. N. R. Jennings. Coordination Techniques for Distributed Artificial Intelligence. University of London Mile End Rd.London E1 4NS UK, 1995. Om Prakash Shukla, Amit Bahekar, Jaya Vijayvergiya, ”Effective Fault Diagnosis and Maintenance Optimization by Genetic Algorithm” Available : http://researchjournals.in/documents/published/2204.pdf
14. Optimization by Genetic Algorithm” Available : http://researchjournals.in/documents/published/2204.pdf
15. Leon Bambrick Supervisor Dr B Lovell “Lecture Timetabling Using Genetic Algorithms” Available : http://secretgeek.net/content/bambrilg.pdf
16. Alberto Colorni, Marco Dorigo “A Genetic Algorithm to solve the time table problem” Available :http://citeseerx.ist.psu.edu
17. Sanjay R. Sutar, Rajan S. Bichkar “University Timetabling based on Hard Constraints using Genetic Algorithm” Available : http://research.ijcaonline.org/ volume42/number15/ pxc3877964.pdf
18. Eng.Ahmed Hamdi Abu ABSA, Dr Sana'a Wafa Al-Sayegh,” Elearning Timetable Generator Using Genetic Algorithms” Available:https://uqu.edu.sa/files2/tiny_mce/plugins /filemanager/files/30/papers/f18 9.pdf
19. Leon Bambrick, “Lecture Timetabling Using Genetic Algorithms” Available: http://secretgeek.net /content/bambrilg.pdf.
20. Evaggelos Nonas, Alexandra Poulovassilis,”optimisation of active rule agents using a genetic algorithm approach pdf” Available : http://ebookbrowse.com/optimisation-of-active-rule-agentsusing-a-genetic-algorithm-approach-pdf-d381872402
21. Cite Seerx , Optimisation of Active Rule Agents using a Genetic Algorithm approach,Available: http://citeseerx.ist.psu.edu/ viewdoc/summary ? doi= 10.1.1.56.2599
22. Wikipedia, Selection (genetic algorithm), Available: http://en.wikipedia.org /wiki/Genetic_algorithm
23. Genetic Algorithms, Conclusion and Future Work Available: http://www.doc.ic.ac.uk/~nd / surprise_96/journal/vol4/tcw2/report.html
24. Wikipedia,Mutation (genetic algorithm) Available: http://en.wikipedia.org/wiki/Mutation%28 genetic_algorithm%29