ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

A Literature Review on Timetable generation algorithms based on Genetic Algorithm and Heuristic approach

Anisha Jain[1], Ganapathy S C Aiyer[2], Harshita Goel[3], Rishabh Bhandari[4]

Student, Computer Engineering, MPSTME, Mumbai, India[1,2,3,4]

Abstract: The problem of timetable scheduling is described as a highly constrained NP-hard problem. It is known as the timetabling problem by most researchers. A lot of complex constraints need to be addressed for development of an efficient algorithm to solve this problem. In this paper, we present a comparison among the different techniques that have been developed for timetable generation using Genetic Algorithm and heuristic algorithm.

Keywords: Genetic algorithm, Active Rules, Rule based agents, resource scheduling, heuristic algorithms, Bacterial Foraging, Chemotaxis.

I. INTRODUCTION

The basic principle of natural selection has been considered the main evolutionary tool. As generations pass, biological organisms "evolve" following the principle of natural selection. By this process of natural selection, the fittest survives (survival of the fittest) and reaches some remarkable forms of accomplishment. Genetic Algorithms (GAs) were invented to mimic this process of natural evolution and selection. Genetic algorithms were invented with the idea to use this power of evolution to optimize solutions to problems. John Holland was the father of the first genetic algorithm, which he invented in the early 1970's. [1].

Genetic algorithms (GAs) are search algorithms that begin with a set of potential solutions. This set then evolves toward a set of more optimal solutions. Within the sample set, poor solutions tend to die out while better solutions mate and propagate their advantageous traits using the "survival of the fittest" phenomenon, thus introducing more optimal solutions into the set, solutions that have greater potential. For each new solution that is added, an old one is removed. Thus, the total size of the sample set remains constant. [2].

Genetic Algorithms are more robust than conventional AI algorithms. Genetic algorithms do not break easily even if the inputs are even slightly changed, or in the presence of noise. They adapt to such changes. GAs employs the "survival of the fittest" among individuals to generate a solution for a problem. Each generation consists of a population of character strings that represent the chromosomes in our DNA. Each individual represents a point in a search space and a possible solution. These individuals (or points) are then made to go through the process of evolution.

The three most important aspects of using genetic algorithms are: (1) definition of the objective function, (2) definition and implementation of the genetic representation, and (3) definition and implementation of the genetic operators. Once these three have been defined, the generic genetic algorithm should work fairly well. Beyond that one can try many different variations to improve performance, find multiple optima (species - if they exist), or parallelize the algorithms.

II. USE OF ACTIVE RULES AND GENETIC ALGORITHM TO GENERATE THE AUTOMATIC TIMETABLE [3]

In this paper, the authors have proposed an optimized technique to automate timetable generation system. The proposed technique filters out the best of active rules and uses Genetic algorithm to generate an optimized solution that accommodates various complex constraints such as that for faculties, classrooms, labs, etc.

The proposed paper takes four parameters as input:

- Person – name of lecturers
- Subject – name of courses in the class
- Room – name of classes and capacity of each
- Time interval – starting time and the duration

The authors have defined three sets of constraints:
Validity violation constraints – constraints which need to be incorporated necessarily otherwise there is no guarantee of valid time tables generated.

- Hard constraints – constraints that need to be fulfilled necessarily.
- Soft constraints – constraints that are obvious but fulfilling them is not so demanding. Solutions are considered to be better if these can be incorporated.

Next they have defined what Active rules are. Active Rules are Event-Condition-Action (ECA) rules. The ECA rules execute as follows: "when an event occurs, check the condition and if it is true execute the action". The event part signifies which event led to the invocation of the rule. The condition part is the logical test that is carried out. If

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

the test evaluates to true, the action part is executed. The action part carries out the action to be performed. This may further lead to invocation of new ECA rules. In the next section, they have defined what genetic algorithms are. They have explained how Genetic algorithms (GA) work in a manner similar to Natural Selection. A population pool of chromosomes is maintained which is called strings. The chromosomes are strings of symbols or numbers. These are also called the genotype (the coding of the solution), whereas the solution itself is called the phenotype. These chromosomes need to be evaluated for fitness. Poor solutions are ignored. After making small changes to remaining solutions "natural selection" is allowed to take its course. This helps evolve the gene pool so that better solutions are discovered.

In this paper, the authors have proposed an automatic way that selects the best action to execute when an event occurs (ECA rules have not been considered used yet). The genetic algorithm selects the action to be executed. When an event occurs, the system has several actions that it can choose to perform. For each possible event, there is another ordered set of possible actions that can be performed when that event occurs that needs to be maintained. The genetic algorithm always selects the first action initially, but a genetic algorithm running in parallel may dynamically change the order of the actions. The reactive behavior of how the agents (constraints) respond to events is controlled by the genetic algorithm. There is another way (the "rational" level) to control the agent. This can employed especially if the agent is built using two methods (built by one partially and controlled partially by the constructs). In this case, the architecture should be a part of the agent. Some actions may be selected for execution using the traditional approach and some using other Genetic Algorithm approach. This method can be used for a subset of the events and the actions of the system. This simplifies the design and reduces testing time and maintenance time.

This paper explains how the authors have used a set of active rules to express the knowledge of intelligence and how a genetic algorithm can be used to dynamically prioritize rules. The advantages of this approach are: distributed solution, load balancing and fault situations. These help in optimizing the timetable generation solution. The authors propose that in this, many good solutions can be generated and the genetic algorithm finds the best one of them. In cases where there is only one good solution the algorithm may fail. In such a case, the algorithm can be restarted by the use of Active Rules that will help finding a better solution. This approach has a simplified design and reduced development and maintenance times of rule-based agents.

III. DYNAMIC TIME TABLE GENERATION CONFORMING CONSTRAINTS A NOVEL APPROACH [4]

In this paper, the authors have proposed an approach that solves the time tabling problem. This approach takes into account many constraints including allocation of room, teacher, course, time slot, etc. The algorithm builds the timetable in an incremental manner, dynamically adjusting resources in order of complexity. The algorithm proposed is dynamic in nature. It deals with managing certain constraints as input, then using heuristic approach to scanning all the constraints on priority basis. The sequence of checking of constraints is also dynamic in nature. Though this sequence of constraints can also be altered manually.

There are two approaches, in one course registration is done and then time table is generated while the reverse is done for second approach. The first approach requires that course registration be carried out well in time to generate the timetables in time. This was helpful when plenty of resources were available and timetable generation easy. The other approach generates the timetables first and then course registration is done. This approach was used when resources were limited and utilization of those resources was needed. The authors have divided the constraints into ‘Hard constraints’ and ‘Soft constraints’. Hard constraints are those that cannot be avoided whereas soft constraints can be ignored if fulfilling them is not feasible. The main constraint that the authors have taken under consideration is that one person (teacher or student) cannot be at two places simultaneously or that there is limit on the number of persons accommodated in a room.

The proposed algorithm is based on heuristic algorithm. The algorithm takes values as input and manages the constraints and resource scheduling one by one.
The main features of the algorithm are as follows:

- The system generates intermediate level as well many final reports including weekly time table, teacher timetable, room wise time table, student time table, department level time table etc.
- The system generates separate as well as combined timetable for female campus as well as for male campuses.
- It distributes workload of lectures equally among all the specified time slots.
- It prioritizes time slots according to customized priority. If lecture cannot be adjusted then it can be moved up in higher priority slot until adjusted accordingly.
- User can set gap of the number of days among the lectures, it can dynamically be adjusted as well.
- The time tabling algorithm tries to adjust courses to user customized slots according to specified time.
- The time table software adjusts the course lectures for the groups of female and males separately.
- It tries to adjust the lectures of a course on the same time within the weekdays.
- All parameters are customized by the user.
- It depicts the progress of courses adjustment at intermediate report level and if clashes cannot be removed and impossible to adjust then displays that course and number of lectures, which cannot be adjusted.

- Various programming techniques have been ensure to improve the performance of the system.

The following steps were followed to implement the algorithm:

- Load the course which may be considered as having diverse constraints and will be difficult to adjust, named as problematic course.
- Filter the schedule if the course is defined to adjust in any specific slot of time and then find the slot in the filter schedule where minimum number of lectures already adjusted.
- Then manages the next lecture time of class in the week after the gap of number of days specified by user.
- Then it checks all the constraints if any constraint is violated then again find the minimum gap and repeats the process again and again.
- In case of failure, removes gap and tries, if not succeeded then removes slot and continue this activity until adjustment is achieved.

The system has been implemented using latest tools and techniques. The user interface of the system has been designed in such a manner that the user has the flexibility to adjust input parameters at initial as well intermediate level. The system takes input from the university management system and from the user. Timetables are generated at various levels of the architecture such as at campus, department, and class level. Separate timetables are generated for teachers as well.

Thus, the authors have proposed an algorithm that works heuristically based on bottom up approach. They argue that the evolutionary nature of the algorithm makes it effective to be utilized to remove constraints that can cause overlapping also informally known as clashes in resource utilization.

IV. OPTIMAL TIME-TABLE GENERATION BY HYBRIDIZED BACTERIAL FORAGING AND GENETIC ALGORITHM [5]

In this paper, the authors have presented a hybrid approach to timetabling problem, which uses bacterial foraging, and genetic algorithm techniques. In the proposed algorithm, a point in n-dimensional search space is represented by a bacterium. Here, each point is considered to be a potential solution to the timetabling problem. The foraging behaviour of E. coli bacteria is simulated to search for an optimal solution. Genetic algorithm is used at the chemotaxis stage to give sense of biased-movement to the bacteria.

The authors have mentioned several previously devised approaches based on Genetic Algorithms (GA) and hybrids that have proved effective for solving the problem of timetable generation. In this paper, they have proposed an optimization approach based on the search and optimal foraging behaviour of Escherichia coli (E. coli) bacteria. Bacterial foraging optimization algorithm (BFOA) has been applied in optimal watermarking, network scheduling, optimal design of Yagi-Uda array, edge detection, edge detection in noisy images colour image enhancement, etc. Hybrid approach involving Genetic Algorithms (GA) and Bacterial Foraging Optimization algorithms (BFOA) has been used for function optimization problems. The problem with BFOA is that if the bacteria takes very large steps and the optimum value lies in a valley with steep edges, the search will tend to jump out of the valley by swimming through them without stopping. On the other hand, if the step size values of the bacteria are too small, convergence can be slow, but if the search finds a local minimum it will not deviate too far from it.

In the proposed algorithm based on genetic algorithm and BFOA, a virtual bacterium represents a point in n-dimensional search space where each points maybe a potential solution to the timetabling problem. The chemotaxis of the bacteria is used to search for optimal solutions to the problem.

The authors have used the foraging behaviour of E. coli. as an optimization process. Here, the bacterium seeks to maximize the energy obtained per unit time spent. The process of foraging involves the four stages; a) chemotaxis, b) swarming, c) reproduction, and d) elimination and dispersal. The size of the initial set of solutions is equal to the number of bacteria. In order to reach a global optimum, the whole set of bacteria is made to undergo these four stages in an iterative manner.

The chemotaxis of the bacteria, as defined by the paper, is like a biased random walk where the bacteria try to search for places with better nutrient gradient alternating between “swim” and “tumble”. Swarming stage is the cell-to-cell signalling stage. In the reproduction stage, the weaker individuals are eliminated and a fitter bacterium splits into two bacteria. Elimination and dispersal stage is to avoid falling into premature convergence. If numbers of chemotactic steps chosen by the bacteria are too short or if the number of reproduction levels is not sufficient, premature convergence to local minima occurs.

In terms of genetic algorithms, a set of potential solutions is called the population. Each solution item (individual) in the population is measured by a fitness function. Fitness is a quality value by which a measure of the reproductive efficiency of chromosomes is made. The process of evolution is maintained by selection, crossover and mutation. Those processes are called genetic operators.

For the proposed hybrid approach based on BFOA and GA, for timetable generation, a variable length chromosome representation is applied where each structure represents a complete timetable. This includes the number of periods and the rooms. Each ClassID structure has a unique numeric ID and each such structure encapsulates the information of the teacher, the subject and the student groups allocated. The time-slots available in a day for each room are arranged as a vector. The vector arrangement helps in quick lookup of the classes allocated for a room for a particular time-slot. In each time slot, we can have multiple classes (ClassIDs). Thus, a linked list of ClassIDs is considered as one time slot.

The bacterium generates certain motion patterns due to the presence of chemical attractants and repellents. These patterns are called chemotaxes. For E. coli, this process was simulated by two different moving ways: run or tumble. A bacterium has the flexibility to alternate between these two modes of operation its entire lifetime. The bacterium may tumble after a tumble or tumble after a run. It may run after a tumble or keep running. This alternation between the two modes enables the bacterium to move in "search" for nutrients.

The movement of bacterium may be presented by:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(j) \qquad (1)$$

Where C (i) {i = 1, 2,…, P} is the size of the step taken in the random direction and ϕ (j) is the random direction of movement after a tumble. θ^i (j, k, l) ϵ 〖R〗^n represents a point in the search space, which is the position of ith bacteria at the jth chemotactic step, kth reproductive step, and lth elimination dispersal step. This point also represents a potential timetable solution, which may or may not have clashes.

In the proposed hybrid approach, the movement parameter C (i) ϕ (j) is derived using the crossover and mutation operators of GA as follows:

- The bacteria are kept sorted in an ordered list according to their health.
- For the jth chemotactic step of bacterium i, another bacterium h whose health is better than i is chosen from the list and crossover operator of GA is applied between the two.
- The resultant offspring is mutated with some probability to escape local minima.
- If the mutated bacterium has lower number of clashes than the ith bacterium then the bacteria will swim in the same direction, i.e.,
- The ith bacterium is moved to the place of the resultant bacterium and,
- For (j+1) th step the same healthy bacterium h will be used for crossover and mutation.
- If the resulting bacterium is not better than the current bacterium i then it will not be move to the new location and a different healthier bacterium will be selected next time. This technique provides the sense of biased movement as well as swarming for the problem of time table generation.
- Crossover: A crossover operator is used to recombine two chromosomes in chemotactic step to get a better chromosome (in our case chromosome is replaced by bacterium).

The crossover is selected as follows:

$$\tilde{\phi}_j^u = \lambda\overline{\phi}^v + (1-\lambda)\overline{\phi}^u \qquad (2\text{-a})$$

$$\tilde{\phi}_j^v = \lambda\overline{\phi}^u + (1-\lambda)\overline{\phi}^v \qquad (2\text{-b})$$

Where the equation parameters refer to offspring's generations ϕ^ (-v), ϕ^ (-u) refer to parent's generations and j is the chromosome of jth step and λ is the multiplier.

- Mutation: Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima.

The dynamic mutation operator is selected as follows:

$$\phi_j = \begin{cases} \tilde{\phi}_j + \Delta(k, U - \tilde{\phi}_j), \text{if a random number is 0} \\ \tilde{\phi}_j - \Delta(k, \tilde{\phi}_j - L), \text{if a random number is 1} \end{cases} \qquad (3)$$

Where k is the generation number, L and U are lower and upper domain bounds of the variable ϕj. Δ (k, j) is given as:

$$\Delta(k, \phi_j) = \phi_j \times \eta \times \left(1 - \frac{k}{T}\right)^b \qquad (4)$$

Where η is a random number from [0, 1], T is the number of maximum generation, and b is a system parameter determining the degree of dependency on iteration number. Choosing this mutation operator causes the mutation to become more restrained with increasing generations because the function will tend to deviate less with increasing values of k.

The algorithm and pseudo code for the hybrid approach is as follows:

1. Initialize the bacterial foraging parameters:

P-Population size
NC- Number of chemotactic steps taken by bacteria in its lifetime
NRe- Number of reproduction steps
NED-Number of elimination-dispersal events
PED- Probability of Elimination-dispersal

2. Start elimination-dispersal loop: l = l + 1.
3. Start reproduction loop: j = j + 1
4. Start chemotaxis loop: k = k + 1.

For each bacterium in the population (P) take a chemotactic step for bacterium as follows:

a. Compute initial value of health for this bacterium and save it as clashLast.
b. Move the bacterium in the direction whose bias is determined using GA
c. Calculate the health of bacteria in new location and save it as clash.
d. If clash<clashLast, then swim in the same direction.
e. Repeat step (b) to (d) till clashLast < clash
f. Save the bacterium in the set of healthy bacteria to be used in reproduction step.

5. If k< NC, go to Step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.
6. Reproduction: From the set of healthy bacteria clone the healthy ones and the bacteria, which have higher number of clashes, will die.
7. If j< NRe, go to Step 3, otherwise, go to Step 8.
8. Elimination-Dispersal: The bacterium is eliminated or simply dispersed to a random location in the

optimization domain based on Vi > PED. Where Vi is a specific parameter such as a random number in an interval [0, 1] or desired cost.

9. l<NED, then go to step 2; otherwise end.

Hence, the paper concludes that BFOA and Genetic algorithms offer a great mechanism for solving combinatorial problems. The authors have utilized this BFOA-GA mechanism for solving the timetabling. Using BFOA helps reduce the time taken to converge to a solution, helps achieve global optimum and also to avoid the problem of premature convergence. The use of GA helps in selecting the optimum step size and direction of chemotaxis.

To avoid confusion, the family name must be written as the last part of each author name (e.g. John A.K. Smith). Each affiliation must include, at the very least, the name of the company and the name of the country where the author is based (e.g. Causal Productions Pty Ltd, Australia).

V. CONCLUSION

The first paper proposes a technique that filters out the best active rules and uses Genetic Algorithm to generate an optimised solution. The system selects one rule with the highest priority to fire, or arbitrarily selects one rule to fire if there is more than one with the same priority.

The second paper uses two approaches. One in which course registration is done before generation of timetable and the other in which course registration is done after generation of timetables. The former approach was implemented when there was more number of resources whereas the latter was used when resources were limited and need to be properly utilized. This system has been deployed in Al-Faisal University, Kingdom of Saudi Arabia.

The third paper uses a bacterium to represent a point in n-dimensional search space where each point is may be a potential solution to the timetabling problem. The movement of E. coli bacteria leads to search for an optimal solution. The E. coli bacteria moves (tumbles or runs) to search for nutrient. This search for nutrient gives us the solution for the timetabling problem. Genetic algorithm is used at the chemotaxis stage.

The authors have also computed a graph to show that the time taken to solve the timetabling problem using only GA based algorithm was more than the time taken when using their proposed BFOA-GA based algorithm.

In the first and the third paper, optimal solutions are generated. The algorithm used in the first paper is time consuming. Also, in cases where resources are scarce, the time required maybe considerably high. The second paper proposes a practically implemented approach that deals with both, the abundance as well as scarcity of resources. The third paper that uses BFOA-GA based algorithm reduces time taken for generation of timetable significantly. This paper makes use of E. coli bacteria that moves around in search of nutrient thus giving us the best possible solution even in cases where resources are scarce or plenty. Thus, we conclude that this algorithm will work better than the rest. The comparison between the three papers can be seen as below:

TABLE I COMPARISON

Parameter	Genetic Algorithm	Heuristic approach	BFOA-GA algorithm
Performance	Good	Better	Better
Time taken	High	- Less when plenty resources - Higher when less resources	Least
Deployment status	No	Yes	No
Constraints under consideration	- Most trivial: Teacher must not be present in more than one class during the same time slot - Classrooms must not be double booked - Classroom must be large enough to hold the strength of each class - Labs must not be double booked	- Most trivial: A teacher or student cannot be at more than one place at the same time - There is a limit on the number of people that can be accommodated in a room	- No teacher or student can be present in two different classes at the same time - In each class, there must be sufficient seats to accommodate all the students in that class. - Not more than one class per subject per day - Classroom must not be left unallocated for more than 20% of the time per day
Availability of number of resources handled	No	Yes, two approaches proposed, one to be used when scarce resources and other when plenty of resources available	Yes, use of E. Coli bacteria handles the availability of resources

ACKNOWLEDGMENT

We would like to thank **Dr. Sharad Mhaiskar,** Dean, MPSTME and **Dr. Dhirendra Mishra**, Associate Dean, MPSTME and Head of Department, Computer Engineering Department, MPSTME for sharing their valuable expertise and making available all the resources required by us for developing this system.

We would also like to thank **Ms. Krishna Palod,** Professor, MPSTME for her constant guidance and valuable feedback that made the completion of this system possible.

We would also like to thank all members of the Time Table Generation Committee, MPSTME whose insight and expertise greatly aided our research and development.

REFERENCES

[1]. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html
[2]. http://jgap.sourceforge.net/doc/gaintro.html
[3]. Barkha Narang, Ambika Gupta, and Rashmi Bansal, "Use of Active Rule and Genetic Algorithm to Generate Automatic Time-Table," in International Journal of Advances in Engineering Sciences Vol.3 (3), July, 2013.
[4]. Tahir Afzal Malik, Hikmat Ullah Khan, and Sajjad Sadiq, "Dynamic Time Table Generation Conforming Constraints a Novel Approach," in ICCIT 2012.
[5]. Om Prakash Verma, Rohan Garg, and Vikram Singh Bisht, "Optimal Time-Table Generation by hybridized Bacterial Foraging and Genetic Algorithms," in International Conference on Communication Systems and Network Technologies, 2012.