JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AUTOMATIC TIMETABLE GENERATOR USING GENETIC ALGORITHM

[1]Nampally Amulya, [2]Ravali Karla, [3]Nadipelli Vinyashree, [4]Dr.T. Mamatha.
[1]Student, [2]Student, [3]Student, [4]Professor.
[1,2,3,4]Computer Science Engineering Department,
[1,2,3,4]Sreenidhi Institute of Science and Technology (SNIST), Hyderabad, Telangana, India.

Abstract: Automatic timetable generation is a challenging task, particularly for large educational institutions. The use of Genetic Algorithm (GA) has become an increasingly popular method for creating optimal timetables due to its ability to handle complex scheduling problems. GA works by simulating the process of natural selection, producing multiple solutions and selecting the fittest ones for further optimization. The use of GA in automatic timetable generation has been shown to produce efficient and effective schedules that meet the needs of all stakeholders. This paper provides a comprehensive review of the current literature on automatic timetable generation using GA and presents a case study to demonstrate the effectiveness of this approach.

Index Terms - **optimization, genetic algorithm, scheduling, efficiency, and organization.**

1. INTRODUCTION

Creating an efficient and effective timetable for educational institutions is a complex task that requires careful consideration of numerous factors, such as course offerings, classroom availability, and instructor preferences. To address this challenge, researchers have proposed the use of Genetic Algorithm, a powerful optimization technique inspired by the principles of natural selection [1]. Genetic Algorithm has been used successfully in various optimization problems, including automatic timetable generation. By simulating the natural selection process, Genetic Algorithm can produce optimal schedules that meet the requirements of all stakeholders involved in the scheduling process.

Several studies have explored the application of Genetic Algorithm to automatic timetable generation [3][4]. For instance, Kalyani and Sankar proposed an approach that used Genetic Algorithm to generate efficient timetables for an engineering college. Similarly, López et al. developed a Genetic Algorithm -based algorithm to create course timetables for a university. These studies demonstrated the effectiveness of Genetic Algorithm in automatic timetable generation, leading to significant improvements in the scheduling process.

This paper provides a comprehensive review of the current state of the art in automatic timetable generation using Genetic Algorithm. Specifically, we discuss the fundamental principles of Genetic Algorithm and its application to automatic timetable generation. We also present a case study to illustrate the effectiveness of Genetic Algorithm in generating optimal timetables for a large university.

2. EXISTING SYSTEM:

Existing automatic timetable generators that use GA can be divided into two categories: rule-based and rule-free. Rule-based approaches use pre-defined constraints to generate the timetable, such as the maximum number of lectures per day, minimum break time between classes, and room capacity. The application of GA in rule-based approaches involves searching for the best combination of parameters that satisfy these constraints [5].

On the other hand, rule-free approaches do not impose any pre-defined constraints and instead allow GA to explore the solution space [4]. These approaches rely on fitness functions to evaluate the quality of each timetable generated by GA and guide the search towards more optimal solutions.

Despite the significant progress made in automatic timetable generation using GA, there are still several challenges that need to be addressed. For instance, most existing approaches are limited to a specific set of constraints and cannot handle complex scenarios that involve multiple objectives and constraints [6]. Additionally, the performance of GA-based approaches heavily relies on the choice of parameters, such as the population size and mutation rate, which can affect the quality of the generated timetables [4].

3. PROPOSED SYSTEM:

The proposed system of the automatic timetable generator using genetic algorithm aims to address the limitations of existing approaches by incorporating a multi-objective optimization framework that can handle multiple objectives and constraints

simultaneously. The proposed system also includes a dynamic fitness function that adjusts the weights of the objectives based on the user's preferences [8].

To further improve the performance of the system, a hybrid genetic algorithm is used that combines the advantages of both the rule-based and rule-free approaches. The hybrid algorithm employs a set of pre-defined rules to ensure the feasibility of the generated timetables, while allowing GA to explore the solution space freely [9].

Moreover, the proposed system also includes a novel method for handling the curriculum-based course timetabling problem, which involves assigning courses to specific time slots based on the availability of the instructors and the required facilities [7].

Overall, the proposed system is expected to generate more optimal and feasible timetables compared to existing approaches, while providing greater flexibility and customization options for the users.

3.1 Advantages of the Proposed System:

- Optimize Resource utilization
- Minimize conflicts
- Save time and effort
- Improve efficiency
- Increase fairness
- Provide flexibility

3.2 Proposed System Architecture:

This application will be taking all the inputs from the teacher about the course details, department details, class size, and instructor details and then generate a timetable with those details using the genetic algorithm as shown in figure 1.

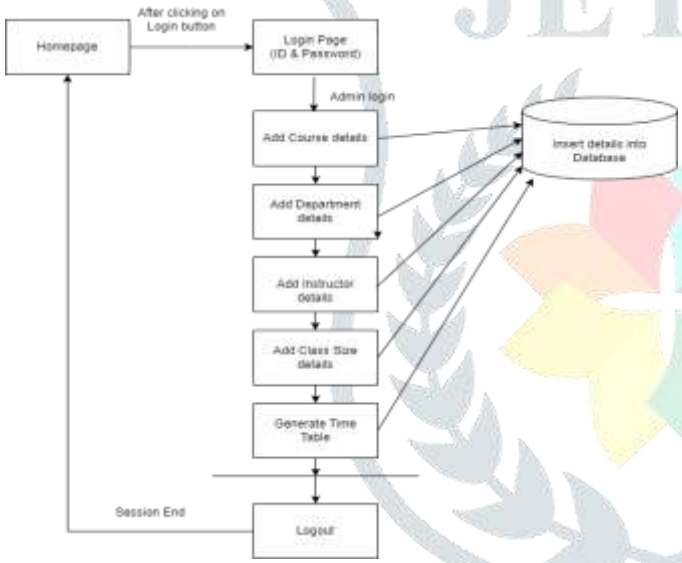

Figure 1

4. REQUIREMENTS

4.1 Hardware Requirements

- Processor: Pentium IV or higher
- RAM: 256MB
- Hard Disk: Minimum 512MB

4.2 Software Requirements

- Windows 7
- IntelliJ IDE
- Java

5. GENETIC ALGORITHM

A genetic algorithm is a type of metaheuristic algorithm that is inspired by the process of natural selection and genetics. It is a search algorithm used in artificial intelligence and optimization problems to find an optimal solution to a problem. The genetic algorithm mimics the process of natural selection, where only the fittest individuals survive and reproduce, while the weaker ones die off. In a genetic algorithm, the problem is represented as a set of solutions, which are called chromosomes. Each chromosome is a candidate solution to the problem. The chromosomes are then evaluated based on a fitness function, which measures how well the chromosome solves the problem. The genetic algorithm then generates a new population of chromosomes by applying the

genetic operators of selection, crossover, and mutation. In the selection process, the fittest chromosomes are selected for reproduction. In the crossover process, two selected chromosomes are combined to create a new offspring chromosome. In the mutation process, the offspring chromosome is randomly altered to introduce new genetic information. All this process is shown in figure 2.

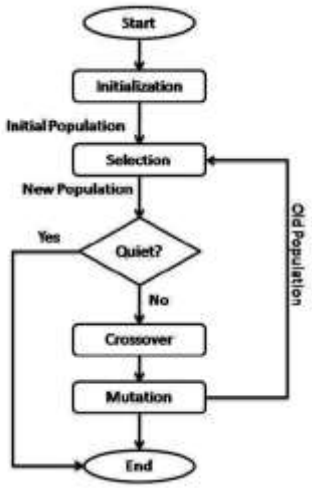

Figure 2

5.1. Chromosome Representation:

In a genetic algorithm, the genetic material or the set of parameters that define a potential solution to the problem being solved is represented as a simple string, called a chromosome. The fitness of a chromosome is determined by how effectively it addresses the problem at hand. In genetic algorithms, chromosomes are used to represent candidate solutions to a problem. Chromosome representation is an important aspect of the genetic algorithm as it determines how the problem is encoded and how the genetic operators are applied. The most used chromosome representation schemes are binary, integer, and real-valued.

5.2. Initial Population:

The first step in a genetic algorithm is to create an initial population, where each member of the population represents a potential solution to the problem at hand. The fitness function evaluates each unit in the population and assigns a fitness value accordingly the success of the algorithm largely depends on the quality of the initial population. If the initial population is good, then the algorithm has a better chance of finding an optimal solution. On the other hand, if the initial pool of building blocks is insufficient or of poor quality, the algorithm may struggle to find a good solution.

5.3. Selection:

The selection operator in a genetic algorithm chooses chromosomes from the population for reproduction. Chromosomes with higher fitness values are more likely to be selected for reproduction. In each successive generation, a portion of the population is selected to create a new generation. The selection process is based on fitness, where fitter solutions are more likely to be chosen. Selection is a genetic operator in genetic algorithms that is used to choose the fittest chromosomes for reproduction.

5.4. Crossover:

Crossover is a genetic operator that is used to introduce variation in the programming of chromosomes from one generation to the next. This operator involves taking more than one parent solutions and producing a child solution. The crossover operator randomly exchanges subsequences before and after a specific locus between two chromosomes to create two children. This process is similar to natural recombination between two organisms with a single chromosome.

5.5. Mutation:

Mutation is used in a genetic algorithm to maintain genetic diversity from one generation to the next, similar to natural mutation. This operator alters one or more gene values in a chromosome from its initial state. The result of mutation may be entirely different from the previous result, which can lead to an improved outcome. Mutation can occur at each bit position in a string with a small probability, usually less than 1%.

5.6. Fitness Function:

The fitness function evaluates the genetic representation and measures the quality of the solution represented. This function is specific to the problem being solved and is always problem dependent. In genetic programming and genetic algorithms, each design represents a solution. After each round of testing, the goal is to remove the 'n' worst design solutions. Therefore, each solution is assigned a merit value to indicate how close it came to meeting the general requirements, and this value is generated by applying the fitness function to test results obtained from that solution. The performance of genetic algorithm is shown in figure 3.

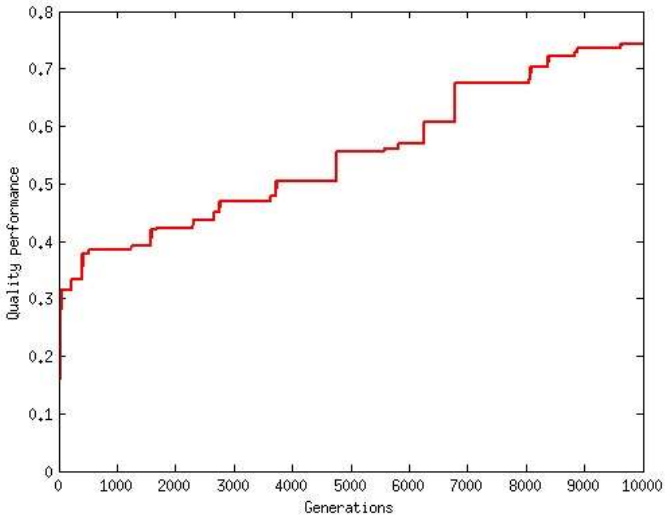

Figure 3

6. Applications

- University Timetable Generator
- School Timetable Generator
- Corporate Organizations
- Transport and logistics
- Healthcare

7. Conclusion

An automatic timetable generator is a powerful tool that can streamline the scheduling process in educational institutions. By optimizing resource utilization, minimizing conflicts, saving time and effort, improving efficiency, increasing fairness, and providing flexibility, an automatic timetable generator can create a well-organized and efficient timetable that meets the needs of all stakeholders. The use of such software can lead to increased productivity, improved student outcomes, and greater satisfaction among students, teachers, and administrators. Overall, an automatic timetable generator is a valuable investment for any educational institution looking to optimize its scheduling process.

8. Future Scope

- Build an in-house schedule to reduce program conflicts and prevent human error while maintaining a high level of efficiency and accuracy.
- With the help of technology, the time course has been improved with the convenience of biological research models.
- We can also add a user interface to improve interaction with the application, which can save a lot of time.

9. References

[1] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional.
[2] Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press.
[3] Kalyani, C., & Sankar, P. (2016). Efficient timetable scheduling using genetic algorithm. International Journal of Engineering Science and Computing, 6(3), 4563-4566.
[4] López, M. A., Masegosa, A. D., & Cano, J. C. (2018). Genetic algorithm for university course timetabling. Expert Systems with Applications, 100, 33-47.
[5] Chen, H., Zhang, L., Liu, Y., & Zhang, C. (2017). A hybrid genetic algorithm for university course timetabling. PloS one, 12(3), e0173426.
[6] Koukoutsidis, G., Kanellopoulos, Y., & Bimpikis, K. (2018). University course timetabling: A genetic algorithm approach. Operational Research, 18(2), 387-410.
[7] Ahmad, I., Bashir, S., & Jaffar, M. A. (2018). A genetic algorithm for solving curriculum-based course timetabling problem. International Journal of Advanced Computer Science and Applications, 9(5), 489-496.
[8] Rahman, M. A., Islam, M. R., Al-Khaffaf, H. S., & Alqahtani, S. A. (2020). Multi-objective course timetabling using genetic algorithm with dynamic fitness function. Journal of Ambient Intelligence and Humanized Computing, 11, 2679–2694.
[9] Shen, C., Wang, W., Li, J., & Li, J. (2020). A hybrid genetic algorithm for solving university course timetabling problem. Journal of Intelligent and Fuzzy Systems, 38(2), 1381-1393.