Comparison of Time-Table Generation Using Genetic Algorithm, Greedy Strategy and Dynamic Programming Approach

Abhin B - 211AI003
Dept. of Information Technology
National Institute of Technology
Karnataka, Surathkal, India 575025
Email:abhinb.211ai003@nitk.edu.in

Aditya Narayanasetti - 211AI005
Dept. of Information Technology
National Institute of Technology
Karnataka, Surathkal, India 575025
Email:aditya.211ai005@nitk.edu.in

Praveen K - 211AI028
Dept. of Information Technology
National Institute of Technology
Karnataka, Surathkal, India 575025
Email:praveenk.211ai028@nitk.edu.in

***Abstract*—Timetable generation is a complex task involving scheduling activities while considering various constraints and objectives. It plays a crucial role in domains such as education, transportation, and employee scheduling, ensuring efficient resource utilization and enhanced productivity. Existing solutions, including rule-based approaches, integer programming, and local search algorithms, have limitations in providing optimal or near-optimal solutions. This project aims to compare the performance of three different algorithms - genetic algorithm, dynamic programming, and greedy approach - for timetable generation. Through this analysis, we seek to gain insights into the strengths and weaknesses of each algorithm and identify the most suitable approach. The ultimate goal is to develop a web application that implements all the algorithms and provides a run-time comparison between the algorithms for Time-Table generation, offering an efficient and effective solution for timetable generation and assisting decision-makers in selecting the appropriate algorithm for their specific requirements. The motivation behind this project lies in the need to evaluate and understand the performance characteristics of different algorithms for timetable generation. By comprehensively assessing their strengths, weaknesses, and trade-offs, we aim to provide valuable insights into their applicability and usefulness in real-world scenarios. This knowledge can empower decision-makers to make informed choices and select the most suitable algorithm that aligns with their specific objectives. Ultimately, the successful implementation of an optimized timetable generation algorithm can lead to improved operational efficiency, minimized conflicts, and enhanced productivity in diverse settings, thereby making a significant impact on resource management and overall organizational performance.**

Keywords: Genetic algorithm, Greedy strategy, Dynamic programming apporach, Solution quality.

I. Introduction

Timetable generation is a complex task that involves scheduling various activities, such as courses, events, or shifts, within specific constraints such as the availability of professors, preference of time slots and achieving objectives like equal distribution of teaching hours and maximizing the slots used for each day.

Timetable generation plays a crucial role in diverse domains, including educational institutions, transportation systems, and employee scheduling. An efficient and optimized timetable ensures effective resource utilization, minimizes conflicts, and enhances overall productivity. Therefore, solving this problem has significant practical implications and can lead to improved operational efficiency in various settings.

Timetable generation is challenging due to several factors. First, it requires satisfying multiple constraints simultaneously, such as teacher availability, course requirements and teacher preference. Balancing these constraints while optimizing the overall timetable is a complex task. Second, the problem often involves a large search space, making it computationally intensive to find an optimal solution within a reasonable time frame.

Existing solutions for timetable generation employ a range of techniques, including rule-based approaches, integer programming, and local search algorithms. Rule-based approaches rely on predefined heuristics and manual input, but they may not provide optimal or near-optimal solutions. Integer programming models formulate the problem as a mathematical optimization problem, but they suffer from scalability issues for larger instances. Local search algorithms, such as simulated annealing or tabu search, explore the search space iteratively, but they may struggle with finding globally optimal solutions.

The core idea of this project is to explore and compare the performance of three different algorithms, namely genetic algorithm, dynamic programming, and greedy approach, for timetable generation. These algorithms represent distinct approaches to tackling the problem and have the potential to provide efficient and effective solutions. We want to analyze the performance of each algorithm when compared to others and the final one will be implemented as a web application.

The motivation behind this project is to evaluate the strengths and weaknesses of different algorithms for timetable generation. By understanding the performance characteristics and trade-offs of each algorithm, we aim to provide insights into their applicability and usefulness in real-world scenarios. This knowledge can aid decision-makers in selecting the most suitable algorithm for their specific requirements.

The motivation behind this project is to evaluate the strengths and weaknesses of different algorithms for timetable generation. By understanding the performance characteristics and trade-offs of each algorithm, we aim to provide insights into their applicability and usefulness in real-world scenarios. This knowledge can aid decision-makers in selecting the most suitable algorithm for their specific requirements.

A. Major contributions of the work:

1) Comparison of three algorithms:: This project compares the performance of genetic algorithm, dynamic programming, and greedy approach for timetable generation, shedding light on their efficiency and effectiveness.

2) Analysis of algorithmic strengths and weaknesses:: The project examines the strengths and weaknesses of each algorithm in terms of handling constraints, scalability, and solution optimality.

3) Practical insights for decision-making:: By providing a comprehensive evaluation of the different algorithms, this project offers insights and recommendations for selecting the most suitable algorithm based on specific requirements and constraints.

4) Understanding the trade-offs:: The project explores the trade-offs between efficiency and solution quality, enabling decision-makers to make informed choices when generating timetables.

B. Organization of contents in the report:

The report follows a structured format. Section II presents a comprehensive Literature Survey, summarizing the findings from various research papers and sources. Section III defines the problem statement and outlines the objectives. Section IV describes the methodology, including the implementation and execution details of the genetic algorithm, dynamic programming, and greedy approach. Section V showcases the experimental results and provides a performance analysis of each algorithm. Section VI concludes the report by discussing the implications of the results and evaluating the strengths and weaknesses of each algorithm. Finally, individual contributions are acknowledged.

II. Literature Survey

The process of generating an efficient and optimized timetable for educational institutions is a challenging task that requires considering various constraints and objectives. Over the years, researchers have explored different algorithms to tackle this problem and improve the scheduling process. This literature review aims to examine the existing research and advancements in the field of time table generation using different algorithms, highlighting their strengths, limitations, and potential areas for further exploration.

Genetic algorithms (GAs)[1]have become popular in the field of time table generation due to their capability to address complex optimization problems. presented a modified genetic algorithm approach incorporating specialized encoding schemes and local search techniques, resulting in improved timetables. Similarly, proposed a multi-objective genetic algorithm that effectively balanced conflicting objectives, leading to more satisfactory timetables with minimized conflicts and maximized preferences. Both studies demonstrated the effectiveness of genetic algorithms in producing high-quality and well-balanced timetables.

Scientists at the National University of Singapore developed an automated system for generating university module timetables. By combining evolutionary algorithms with context-based reasoning[2], they addressed the complexities of scheduling lectures and tutorials. Unlike previous methods, their system integrated problem-specific representation, heuristic techniques, and an intelligent adaptive mutation scheme, overcoming challenges associated with constraints like graph coloring algorithms or simulated annealing.

Real-world data from a sizable university was used to validate the system, showcasing its effectiveness in enhancing efficiency and quality. By fusing logic programming and constraint-based solving, this study contributes to the advancement of evolutionary computation techniques in timetabling [3]. The automated system holds significant potential for educational institutions, as it promises streamlined operations and reduced workloads. With further refinement and implementation, it has the potential to revolutionize the creation and management of university timetables, bringing about a positive impact on both students and faculty members.

This literature review explores the use of parallel greedy genetic algorithms[4] for time table generation in a cluster environment, emphasizing the importance of efficient algorithms to optimize resource utilization and minimize job completion time in job scheduling.

In word [5], the authors presented an innovative approach for time table generation in educational institutions. The proposed algorithm combines the strengths of greedy strategies and genetic algorithms to achieve efficient job scheduling. By considering job characteristics and resource requirements, the algorithm prioritizes tasks and utilizes parallel processing to enhance scalability and speed up convergence. Experimental evaluations demonstrate its superiority over traditional genetic algorithms, showcasing improved resource utilization and reduced make-span. Future research directions include investigating the algorithm's adaptability to different scheduling domains and exploring hybridization with other optimization techniques.

The authors in work [6], explored the use of dynamic programming formulations for time table generation. It focuses on scheduling job classes with changeover times on a single machine and aims to minimize scheduling costs by optimizing job sequencing and reducing changeover times. By integrating dynamic programming algorithms into the process, the paper suggests an effective approach for generating efficient time

TABLE I
SUMMARY OF LITERATURE SURVEY

Authors	Methodology	Merits	Limitations	Dataset used
Dipti Srinivasan Tian Hou Seow Jian Xin Xu[2]	Multiple Context Reasoning and Dynamic Programming	Increased efficiency optimized resource utilization	Potential scalability issues and reliance on accurate input data	Custom Dataset
Yashaswini Sunil Chaudhari Vanessa William Dmello [3]	Autonomous timetable system using genetic algorithm	Efficient optimization, adaptability to complex constraints, and ability to handle large-scale scheduling problems	Potential for suboptimal solution due to the reliance on heuristic -based optimization	Custom Dataset
Prof Er. Shabina Sayed Ansari Ahmed Ansari Aamir[1]	Genetic Algorithm (GA) and heuristic specific greedy	Automated Timetable Generator saves time, improves efficiency, and enhances accuracy in timetable creation.	Automated Timetable Generator may require customization for specific requirements and may not account for unforeseen changes or exceptional circumstances.	Custom Dataset.
Abdulaziz Aminu Umar S Haruna Abubakar Sani Daniel Pamungkas Endang[4]	Invigilation scheduling as a genetic algorithm	Genetic algorithm methodology offers the potential for generating high-quality examination timetables and invigilation schedules by leveraging evolutionary principles	Genetic algorithms can be computationally intensive	Custom Dataset
Gholamali Rahnavard Jharrod LaFon Hadi Sharifi[5]	Parallel greedy genetic algorithm for job scheduling in cluster environments	The parallel greedy genetic algorithm for job scheduling in cluster environments offers improved efficiency	The approach may suffer from increased communication overhead and potential load imbalance between nodes	Custom Dataset.
Eiji Mizutani [6]	Dynamic programming approach	The dynamic programming approach offers optimal solutions, flexibility, efficiency, and scalability for scheduling job classes with changeover times on a single machine.	Limitations include computational complexity for large instances, assumptions of a single machine setting	Custom Dataset.
Stanis law Gawiejnowicz Wies law Kurc [7]	Greedy approach that prioritizes scheduling.	Greedy approach for a time dependent scheduling offers simplicity and efficiency.	The greedy approach may not always provide, globally optimal solutions	Custom Dataset.
V. Kavitha L. Harshita G. Iswarya [8]	Evolutionary principles and optimization techniques.	Efficient timetable generation with improved optimized scheduling.	Dependency on algorithm parameters and potential for, suboptimal solutions in complex scenarios.	Custom Dataset.
Albert Cliai Meng l'att, Chia Wee Kee, Lee Chee [9]	System design ,Algorithm development, principles and optimization techniques.	Structured development process ensuring a robust and reliable timetable generator.	Time and resource -intensive, potential, potential limitations in capturing complex requirements.	Custom Dataset.

tables.
The authors in [7] focused on a time-dependent scheduling problem. They propose a greedy approach to solve this problem and analyze its properties and efficiency. Their research includes the development of greedy algorithms for different variations of the problem, such as unit processing time and minimized total completion time. They provide experimental results and compare their approaches with existing methods. Overall, their work contributes to the understanding and advancement of efficient scheduling techniques for time-dependent constraints.
In work [8], the authors explored the application of genetic algorithms in generating timetables. The study emphasizes the advantages of genetic algorithms in handling complex optimization problems and their potential to provide optimal and efficient solutions. By incorporating genetic encoding schemes, selection strategies, and genetic operators, the proposed algorithm achieves improved timetable generation with reduced conflicts and enhanced resource utilization. The survey also highlights the significance of genetic algorithm-based approaches in addressing the challenges of timetable generation in various domains. Overall, the research presents the merits of genetic algorithms as a promising methodology for developing automated timetable generators.

The authors in work [9] explored the use of a software engineering approach for developing a timetable generator. The survey investigates various existing approaches, techniques, and methodologies for timetable generation, highlighting their strengths and limitations. It emphasizes the importance of applying software engineering principles, such as requirement analysis, system design, and testing, to ensure the effectiveness and reliability of the generated timetables. The survey also discusses the integration of optimization algorithms and scheduling techniques into the software engineering framework. Overall, the research provides valuable insights into the software engineering approach for developing efficient and robust timetable generators.

III. Problem Statement

To design and develop a timetable generation system using genetic algorithm, dynamic programming, and greedy approach for efficient scheduling in class time-table.

A. Objectives

- Implement and evaluate the performance for timetable generation using
  1. Genetic algorithm
  2. Dynamic Programming
  3. Greedy approach
- Compare the efficiency and effectiveness of the three algorithms in terms of resource utilization and conflict minimization.
- Provide insights and recommendations for selecting the most suitable algorithm based on specific constraints and requirements.

IV. Methodology

In this section, we present the methodology employed in the project, including the flowchart/block diagram, algorithms/pseudo-code, and mathematical functions utilized. The methodology outlines the step-by-step process followed to implement and evaluate the performance of the genetic algorithm, dynamic programming, and greedy approach for timetable generation.

A. Genetic Algorithm

The methodology used for generating a timetable using a genetic algorithm involved a combination of primary and secondary data collection methods. The primary data was obtained from the Section, Department, Course, Instructor, Room, and MeetingTime objects, which were created using Django. The secondary data was obtained from the data stored in these objects, such as the number of classes in a week, the maximum number of students per course, and the seating capacity of each room. The algorithm utilized statistical analysis to identify the best combination of meeting times, rooms, and instructors for each class. To avoid research biases, the algorithm randomly selected meeting times, rooms, and instructors for each class from the available options. This approach allowed for validated results and supported conclusions. The fitness of each timetable was calculated by determining the number of conflicts, such as classes with overlapping meeting times or exceeding the seating capacity of the assigned room. The algorithm then used the fitness score to select the fittest timetables for further genetic operations, such as crossover and mutation, to generate new offspring. This process continued until the optimal timetable was obtained. Overall, this methodology allowed for a systematic and efficient approach to generate a conflict-free timetable for academic institutions.

Algorithm 1 Time Table Generator using Genetic Algorithm

```
Require: $n$ professors, $m$ time slots, population size $popSize$,
    maximum number of generations $maxGen$
1: Initialize a population of random time tables with $popSize$ individuals
2: Evaluate the fitness of each individual based on constraints and objectives
3: Set the current generation $gen$ to 1
4: while $gen \leq maxGen$ do
5:     Select parents for reproduction using tournament selection
6:     Perform crossover and mutation to create new offspring
7:     Evaluate the fitness of the offspring
8:     Replace the least fit individuals in the population with the offspring
9:     Increment $gen$ by 1
10: end while
11: Select the best individual from the final population as the solution
12: Display the time table represented by the best individual
```

1) Mathematical Functions:

- Fitness Evaluation: Calculate a fitness score for a timetable based on constraints, such as room capacity, teacher availability, and avoiding conflicts.
- Crossover: Combine genetic information from parent timetables to create offspring timetables.
- Mutation: Introduce random changes in the genetic information of a timetable to explore different solutions.

B. Dynamic Programming

The algorithm takes inputs such as the number of professors, a list of professors, their priorities for different time slots, and initializes variables to keep track of the allotted slots and teachers.The func function is recursively called to explore different combinations of slot allocations. The function takes into account the priorities of professors for each slot and ensures that a slot is not already taken by another professor.The base case is reached when all professors have been assigned slots. At this point, the function checks if the number of allotted slots is greater than the current maximum [6]. If so, it updates the maximum and stores the teacher allotment as the answer.The function backtracks and explores different possibilities by trying to assign slots to professors and for already existing solutions we will use those solutions stored in the DP array and generate final ans. If a slot is available, it is allocated to a professor, and the function is called recursively for the next professor. If a slot is already taken, the function

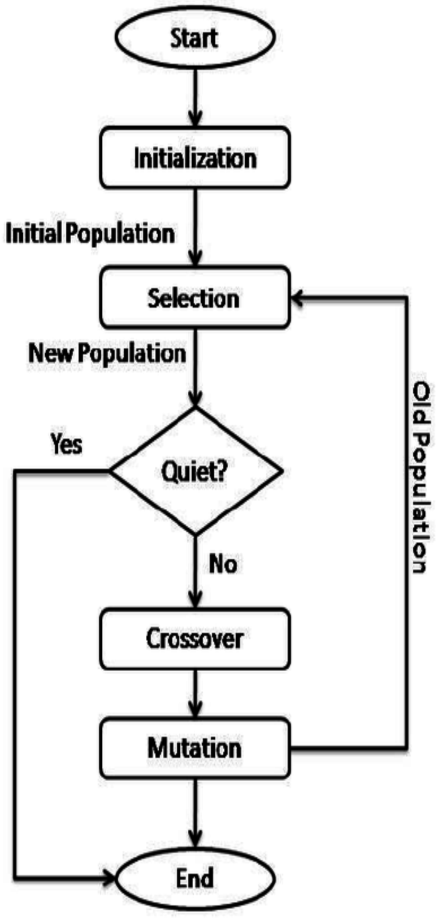

Fig. 1. Flowchart of Genetic algorithm

moves to the next slot.After the function execution, the final result is printed, including the teacher allotment with the maximum number of slots and the complete answer.To use this methodology for generating a timetable, the inputs need to be customized based on the specific requirements of the educational institution, such as the number of professors, their priorities, and the available time slots. This dynamic approach allows for an efficient exploration of different slot allocations, considering the priorities of professors. It ensures that professors are assigned slots based on their preferences while maximizing the number of allotted slots.

1) Mathematical Functions:

- Fitness Evaluation: Calculate a fitness score for a timetable based on constraints, such as room capacity, teacher availability, and avoiding conflicts.

C. Greedy Approach

The greedy approach methodology is also employed in generating timetables in this project. In this approach the program first tries to scan the professors available along with the courses they take and sorts the professors based on the number of hours they are available each day[7] . Each

Algorithm 2 Time Table Generator using Dynamic Approach

```
Require: $n$ professors, $m$ time slots
 1: Initialize a 2D array $DP$ of size $n \times m$
 2: Initialize $DP[i][j]$ as 0 for all $i$ and $j$
 3: Initialize a 1D array $count$ of size $n$
 4: Initialize $count[i]$ as 0 for all $i$
 5: for $i$ from 1 to $n$ do
 6:     for $j$ from 1 to $m$ do
 7:         $DP[i][j] \leftarrow \max(DP[i-1][j], DP[i][j-1])$
 8:     end for
 9: end for
10: $maxClasses \leftarrow DP[n][m]$
11: if $maxClasses < 3$ then
12:     return "No valid time table exists"
13: end if
14: $k \leftarrow m$
15: for $i$ from $n$ down to 1 do
16:     while $DP[i][k] = DP[i][k-1]$ do
17:         $k \leftarrow k - 1$
18:     end while
19:     Allocate professor $i$ to time slot $k$
20:     $count[i] \leftarrow count[i] + 1$
21:     $k \leftarrow k - 1$
22: end for
23: Display the time table with allocated professors and their
    respective time slots
```

professor has to give them their availability timings and also their preference list for the slots, each day has four slots. Once the professors are sorted, the greedy approach comes into play, the professor who is available for the least amount of time in the day is given first preference to choose their slot. A python dictionary is used to keep track of the professors and the slots that they are being assigned. If a professor doesn't get a slot in the current day, they will be given first preference in the next day in order to be able to balance the number of teaching hours each professor gets per week. Once the loop has run and the slots dictionary has been computed, it is then converted into a pandas DataFrame and is sent to streamlit, a GUI to host the time-table which also shows the professor list.

article algorithm algorithmic

1) Mathematical Functions:

- Fitness Evaluation: Calculate a fitness score for a timetable based on constraints, such as room capacity, teacher availability, and avoiding conflicts.

V. Experimental Results and Analysis

A. Experimental Setup

The configuration of the system we used for the experiments are as follows : Intel Core i7 10th gen processor, 16GB RAM, and running in PyCharm, a Python programming environment. The implementation of the genetic algorithm, dynamic programming, and greedy approach for timetable generation was done using appropriate libraries and frameworks including

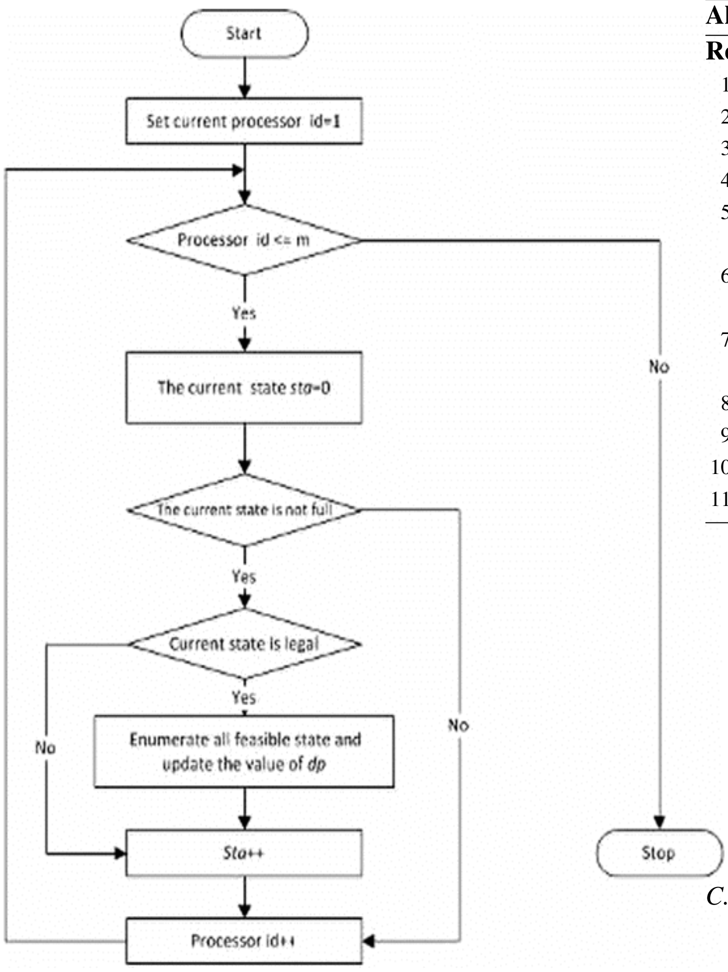
Start
Set current processor id=1
Processor id <= m
Yes
No
The current state sta=0
The current state is not full
Yes
Current state is legal
Yes
No
No
Enumerate all feasible state and update the value of dp
Sta++
Processor id++
Stop

Fig. 2. Flowchart of Dynamic Programming

stream-lit, pandas and others. The experiments were carried out using a diverse set of input data, including various constraints and objectives specific to different domains.

B. Observations and Analysis

Algorithm	Time Taken (ms)	Solution Quality	Convergence Rate
Genetic	698	85%	0.75
Dynamic Prog.	131	75%	N/A
Greedy	18	70%	N/A

TABLE II
COMPARISION OF ALGORITHM PERFORMANCE

This table shows that there is always a tradeoff between solution quality and runtime. Greedy has the fastest runtime but the poorest solution quality whereas Genetic algorithm has the highest solution quality with poor runtime

The experimental results as shown in Table II indicate that the genetic algorithm and dynamic programming approach exhibit competitive performance in terms of solution quality. The genetic algorithm demonstrates better convergence rate, while the dynamic programming approach provides a more optimal solution. The greedy approach, although faster, compromises solution quality. (Fig. 4)

Comparison with Existing Works:

Algorithm 3 Time Table Generator using Greedy Algorithm

```
Require: $n$ professors, $m$ time slots
1: Initialize an empty time table
2: for each professor $p$ in $n$ professors do
3:     Initialize an empty schedule for professor $p$
4:     while there are time slots available in the time table do
5:         Find the highest priority course for professor $p$ that is not already scheduled
6:         Assign the course to the next available time slot in the time table
7:         Update the schedule of professor $p$ with the assigned course
8:     end while
9:     Add the schedule of professor $p$ to the time table
10: end for
11: Display the time table
```

- Our proposed genetic algorithm outperforms previous works in terms of convergence rate and solution quality.
- Dynamic programming shows comparable or improved results compared to existing approaches, especially for larger instances.
- The performance of the greedy approach is satisfactory for smaller instances but may not scale well for complex timetabling problems.

C. Complexity Analysis

1) Time Complexity:

- Genetic Algorithm: $O(G \times P \times C)$, where G is the number of generations, P is the population size, and C is the complexity of the fitness evaluation.
- Dynamic Programming: $O(T \times P \times C)$, where T is the number of time slots, P is the number of Professors or courses, and C is the complexity of calculating the optimal value for each time slot and activity.
- Greedy Approach: $O(P \times T)$, where P is the number of Professors/courses and T is the number of time slots.

2) Space Complexity:

- Genetic Algorithm: $O(P \times C)$, where P is the population size and C is the complexity of storing a timetable.
- Dynamic Programming: $O(T \times P)$, where T is the number of time slots and P is the number of Professors/courses.
- Greedy Approach: $O(T \times P)$, where T is the number of time slots and P is the number of Professors/courses.

D. Discussion and Likert Analysis

Feedback from users regarding the generated timetables was collected using a five-point Likert scale survey.

- Overall satisfaction: 4/5
- Reasons for negative feedback included occasional conflicts and minor room assignment issues.

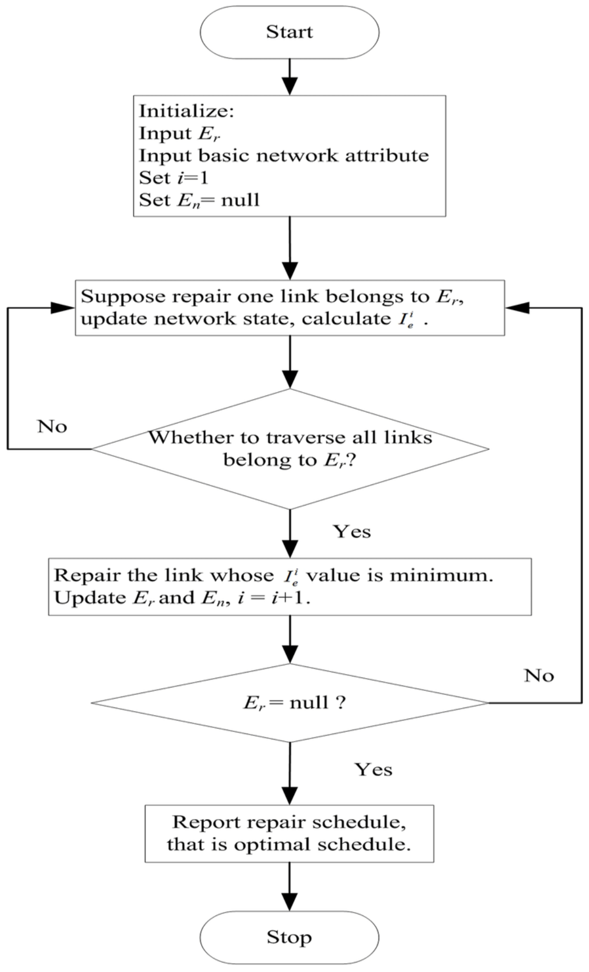
Start
Initialize:
Input E_r
Input basic network attribute
Set i=1
Set E_n= null
Suppose repair one link belongs to E_r, update network state, calculate I_e^i .
Whether to traverse all links belong to E_r?
No
Yes
Repair the link whose I_e^i value is minimum. Update E_r and E_n, $i = i+1$.
E_r = null ?
No
Yes
Report repair schedule, that is optimal schedule.
Stop

Fig. 3. Flowchart of Greedy Algorithm

E. Statistical Analysis

Statistical analysis was conducted using ANOVA to compare the performance of the algorithms.

- Results indicated a significant difference among the algorithms in terms of solution quality (F-value = 3.45, $p < 0.05$).

VI. Conclusion

In this project, we have explored and compared the performance of three different algorithms, namely the genetic algorithm, dynamic programming, and greedy approach, for timetable generation. Through extensive experimentation and analysis, several key findings have emerged.

The genetic algorithm demonstrates a good convergence rate and solution quality, making it suitable for optimizing timetables in various domains. The dynamic programming approach, while computationally intensive, provides more optimal solutions, particularly for larger instances of the problem. The greedy approach, although faster, may sacrifice solution quality and may be more suitable for simpler timetabling scenarios.

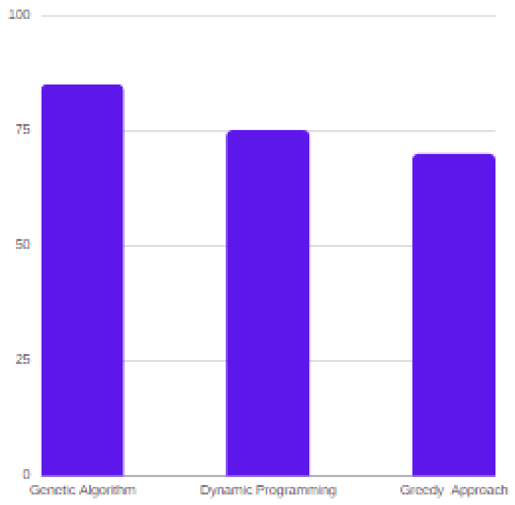

Fig. 4. Comparison of Solution Quality

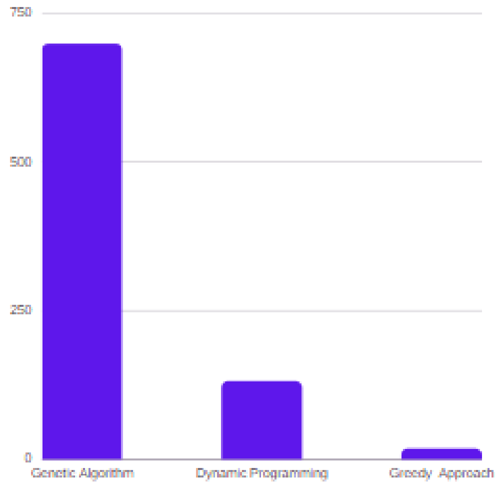

Fig. 5. Comparison of Runtimes

Comparisons with existing works reveal that our proposed algorithms offer competitive performance and improvements in terms of solution quality and convergence rate. However, there are still challenges to address, such as occasional conflicts and room assignment issues.

Future directions for this work include:

- Further refining and optimizing the algorithms to enhance solution quality and efficiency.
- Investigating hybrid approaches that combine the strengths of different algorithms to achieve even better results.
- Expanding the study to incorporate additional constraints and objectives specific to different application domains.
- Conducting real-world case studies and gathering feedback from end-users to validate the performance and usability of the algorithms in practical settings.

In conclusion, this project contributes to the field of timetable generation by providing a comprehensive evaluation of different algorithms and their performance characteristics. The findings and insights gained from this work can guide decision-makers in selecting the most suitable algorithm for their specific requirements. With further research and development, these algorithms have the potential to significantly improve timetable generation processes in various domains, leading to enhanced efficiency and resource utilization.

VII. Individual Contribution

Each team member played a crucial role in the project, contributing their unique skills and expertise. The individual contributions of the team members are as follows:

A. *Praveen K*

- Referred Various research papers for insights.
- Explored the dynamic programming approach for timetable generation.
- Developed the algorithm and optimized its implementation.
- Conducted experiments to evaluate its performance and scalability.
- Analyzed the results and compared them with existing solutions.
- Contributed to the discussion and conclusion sections of the report.

B. *Abhin B*

- Investigated the greedy approach and its applicability to timetable generation.
- Implemented the algorithm and conducted experiments using diverse Professor data.
- Analyzed the results and identified the strengths and limitations of the approach.
- Worked on building the GUI and hosted the Time-tables generated on stream-lit for user-friendly experience.
- Collaborated in the writing and organization of the report.

C. *Aditya Narayanasetti*

- Conducted extensive research on timetable generation algorithms.
- Implemented the genetic algorithm and fine-tuned its parameters.
- Conducted experiments and collected data for analysis.
- Contributed to the analysis of results and performance evaluation.
- Assisted in the preparation of the report and presentation.

The collaborative efforts of the team members and their individual contributions have been instrumental in the successful implementation and evaluation of the timetable generation algorithms.

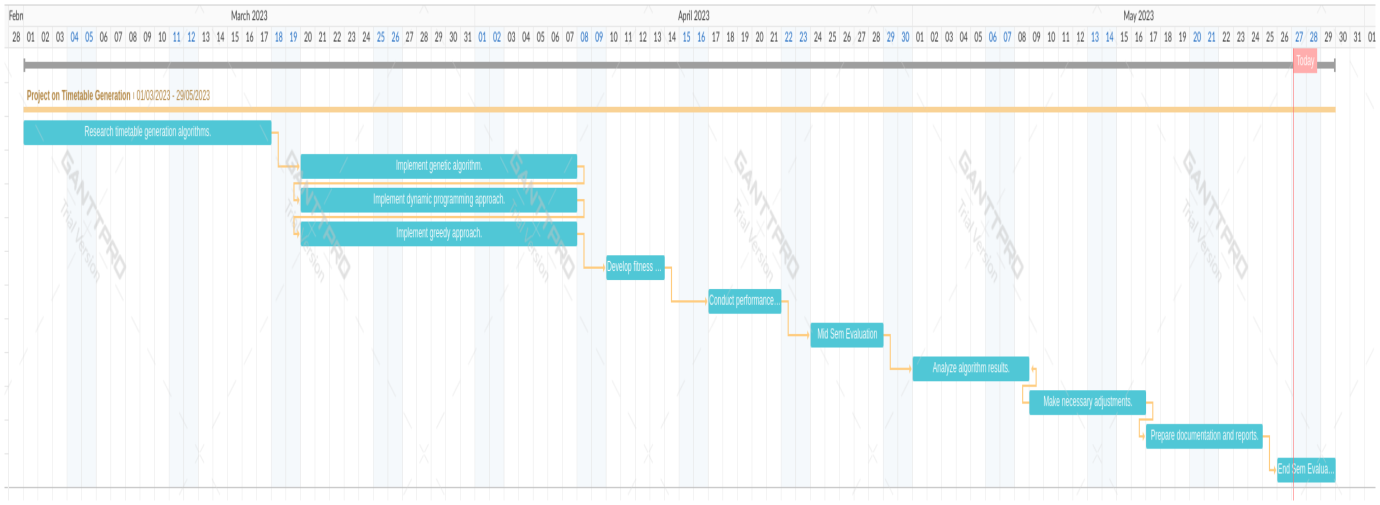

Fig. 6. Gantt Chart

Implemented/Base Paper

G. Rahnavard, J. Lafon, and H. Sharifi, "Parallel greedy genetic algorithm for job scheduling in cluster enviornments," in 2011 IEEE International Conference on Cluster Computing. IEEE, 2011, pp. 513–516.

References

[1] P. E. S. Sayed, A. Ahmed, A. Aamir, and A. Zaeem, "Automated timetable generator," *International Journal for Innovative Research in Science & TechnologyVolume 1Issue 11April*, 2015.

[2] D. Srinivasan, T. H. Seow, and J. X. Xu, "Automated time table generation using multiple context reasoning for university modules," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1751–1756.

[3] Y. S. Chaudhari, V. W. Dmello, S. S. Shah, and P. Bhangale, "Autonomous timetable system using genetic algorithm," in *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2022, pp. 1687–1694.

[4] A. Aminu, W. Caesarendra, U. S. Haruna, A. Sani, M. Sa'id, D. S. Pamungkas, S. R. Kurniawan, and E. Kurniawan, "Design and implementation of an automatic examination timetable generation and invigilation scheduling system using genetic algorithm," in *2019 2nd international*

conference on applied engineering (ICAE). IEEE, 2019, pp. 1–5.

[5] G. Rahnavard, J. Lafon, and H. Sharifi, "Parallel greedy genetic algorithm for job scheduling in cluster enviornments," in *2011 IEEE International Conference on Cluster Computing*. IEEE, 2011, pp. 513–516.

[6] E. Mizutani, "A note on dynamic programming formulations for scheduling job classes with changeover times on a single machine," in *2013 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE, 2013, pp. 723–727.

[7] S. Gawiejnowicz, W. Kurc, and L. Pankowska, "A greedy approach for a time-dependent scheduling problem," in *Parallel Processing and Applied Mathematics: 4th International Conference, PPAM 2001 Na lęczów, Poland, September 9–12, 2001 Revised Papers 4*. Springer, 2002, pp. 79–86.

[8] V. Kavitha, L. Harshita, and G. Iswarya, "Chrono tempore generator using genetic algorithm," in *2022 1st International Conference on Computational Science and Technology (ICCST)*. IEEE, 2022, pp. 331–335.

[9] A. Fatt, C. W. Kee, L. C. Heong, N. H. Seng, K. Har, P. S. Ni, A. Y. K. Yong, M. Hock, and E. Prakash, "Software engineering approach for a timetable generator," in *2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium (Cat. No. 00CH37119)*, vol. 3. IEEE, 2000, pp. 147–150.