# Evaluating Timetabling Algorithms: A Comparative Analysis with Interactive Visualization

1st Gabriel Feitosa Melo Coelho
*Eng. de Software*
*ICEV*
Teresina, Piauí
0009-0002-1531-8276

2nd Arthur Silva Mousinho
*Eng. de Software*
*ICEV*
Teresina, Piauí
arthur.mousinho@grupocev.com

3rd Maria Eduarda Araújo Rodrigues de Freitas
*Eng. de Software*
*ICEV*
Teresina, Piauí
maria_eduarda.freitas@somosicev.com

***Abstract*—In recent decades, technological advancements have significantly impacted various sectors, including education, where optimizing resources is crucial. Timetabling, the process of scheduling classes, exams, and events in educational settings, presents a complex challenge due to the need to satisfy multiple constraints and preferences. Effective timetabling is vital as it influences the daily operations of educational institutions and the experiences of students and teachers. This paper explores the effectiveness of different timetabling algorithms—genetic algorithms, local search algorithms, and randomized approaches—in optimizing school schedules. Through comparative analysis and interactive visualization, the study examines how these algorithms can enhance resource utilization, reduce idle periods, and meet the preferences of students and teachers. The findings demonstrate that AI-driven timetabling aligns with contemporary educational policies, promoting strategic time management and potentially improving educational outcomes and stakeholder satisfaction.**

***Index Terms*—timetabling, genetic algorithm, local search algorithm, comparative analysis**

## I. Introduction

The past few decades have been marked by a technological revolution that has transformed various aspects of society. These changes extend from the industrial sector to education, where the need for resource optimization is increasingly present. In this context, the problem of timetabling, which involves scheduling in educational environments, represents a significant challenge due to the complexity of meeting multiple constraints and preferences.

Timetabling is the process of organizing and allocating schedules for classes, exams, or other events in an educational environment. This task involves coordinating times for various classes, teachers, and available resources such as classrooms or equipment, ensuring there are no conflicts like two courses happening at the same time in the same room or a teacher being scheduled for different classes simultaneously. The complexity increases with the size and diversity of educational institutions, turning it into a multifaceted puzzle that must satisfy a range of academic, institutional, and personal constraints.

The effectiveness of timetabling is crucial because it directly impacts the daily logistics of schools and universities, as well as the educational experience of students and teachers. A good timetabling system must maximize the efficient use of all available resources, minimize idle periods, and accommodate the preferences of students and teachers to optimize learning and teaching.

Recent legislation and educational policies have emphasized the importance of learning environments that maximize the effective use of time and available resources. In this light, the application of AI in timetabling not only aligns with these directives but also promotes more strategic management of school time. Research indicates that optimizing schedules can significantly improve educational performance indicators and the overall satisfaction of those involved in the educational process.

This paper aims to investigate the effectiveness of genetic algorithms in optimizing school schedules, exploring how these tools can be strategically implemented to foster a more efficient and integrated educational environment.

## II. Timetabling Algorithms

### A. Genetic Algorithm-based Approach

The representation of the chromosome in the implementation of genetic algorithms for timetabling issues is crucial because it defines how the problem and its solutions are structured and manipulated. In this context, each chromosome represents a complete potential solution for the school schedule, where:

- Each genome corresponds to a pair of time slot and classroom: This means that each genome holds information about when (the time slot) and where (the classroom) a particular class will occur. This granularity is essential because scheduling conflicts and resource allocation are central issues in timetabling.
- Unidimensional structure of the chromosome: By opting for a unidimensional structure, the complexity of the problem is reduced both visually and operationally. Instead of manipulating a two-dimensional table or matrix (which would naturally represent days and times), a linear sequence is used. This simplifies genetic operations like crossover and mutation, as it allows treating the chromosome as a string or list of genomes.
- Encapsulation of all necessary information: By including all essential allocation information in each genome, the

integrity of the solution is maintained through each genetic operation. There is no need to seek external information or perform additional calculations to understand or modify the state of the chromosome. This streamlines the fitness evaluation process and genetic operations, making the algorithm more efficient.

- Simplification of genetic o: With a unidimensional representation, operations like crossover and mutation become more straightforward. Crossover can involve simply swapping segments between two chromosomes or altering specific genomes through mutation. This is easier to implement and adjust to achieve better performance, compared to bidimensional approaches where spatial relations between elements need to be constantly checked.

*1) Crossover and Mutation Processes:* Crossover is a process by which two parent chromosomes are combined to create one or more offspring chromosomes. This operator is vital because it allows for the exchange of genetic information between solutions, potentially combining beneficial characteristics of both parents to produce more viable or even superior solutions. In the context of timetabling, crossover might, for example, mix different schedule configurations that have proven effective in terms of minimizing scheduling conflicts and optimizing resource use.

Mutation introduces random variations in a chromosome, altering one or more of its genomes. This is essential to maintain genetic diversity within the population and to explore new areas of the solution space that might not be reached through crossover alone. In timetabling, mutation can individually adjust certain schedules or room allocations, allowing the algorithm to escape local minima and explore solutions that might be radically different from those generated by the parents.

*2) Elitism in Genetic Operations:* By choosing only the chromosomes with the best fitness values for crossover and mutation, the algorithm ensures that desirable traits are maintained and possibly enhanced over generations. This aids in the algorithm's convergence, directing it more effectively toward optimal solutions. Using elitism in chromosome selection for reproduction allows for faster convergence, as good solutions are preserved and have the chance to be improved. Moreover, elitism helps prevent the loss of promising solutions due to random operations of crossover and mutation, which can be critical in complex solution spaces like timetabling. However, an excessive focus can lead to reduced genetic diversity, potentially causing the algorithm to concentrate on a niche of the solution space and overlook other potentially viable solutions. To mitigate this, it's important to balance the selection of highly fit chromosomes with the introduction of new chromosomes or sufficient mutation to properly explore the solution space.

*3) Fitness Function Coding:* The fitness function was coded to assess the viability and effectiveness of each schedule solution. The chromosome's fitness is calculated as the sum of the weighted fitnesses of teachers, classes, and classrooms. Specific time coefficients were used to calculate the level of satisfaction or dissatisfaction of teachers with their assigned schedules, considering their particular preferences and aversions.

*4) Implementation Results:* Using a genetic algorithm-based approach allowed for significant optimization of school schedules, resulting in a more balanced and satisfactory distribution of classes.

### B. Local Search Algorithm Approach

Local search algorithms have shown consistent effectiveness in solving educational timetabling problems [6]. This approach not only facilitates the optimization of timetable allocation, but also allows for precise adjustments that meet the multiple constraints and specific objectives of each educational institution.

This type of algorithm stands out for its ability to explore and adjust solutions in a local search space, seeking to continuously improve the quality of the solutions found. By focusing on incremental improvement, the local search algorithm offers a robust and efficient way to deal with complex optimization problems, moving from solution to solution in the space of candidate solutions and applying local changes until an optimal solution is found or a time limit is reached [7].

In this study, a basic local search algorithm was implemented to minimize conflicts in the allocation of school timetables. Following the principles of the local search algorithm [8], the process begins with the generation of a random initial solution, which in this case distributes teachers, classrooms and subjects over the days of the week and time periods available. Next, the algorithm evaluates the quality of the solution by counting conflicts, which are defined as situations where the same time slot or room is assigned to multiple classes simultaneously.

The local search algorithm then iteratively refines the solution by disturbing the current schedule through random changes in class times. Each modification is accepted if it reduces the number of conflicts compared to the best solution found so far [9]. This iterative process continues until there are no more significant improvements or a maximum number of iterations is reached. Finally, the optimized results are stored in a JSON file for practical analysis and applied use.

## III. Interactive Visualization

### A. Frontend

The front-end application was developed to provide an interactive and dynamic visualization of the results from executing algorithms for solving timetabling problems. This solution aims to facilitate the visualization of schedules and resource allocations. An intuitive interface allows users to explore different scenarios, adjust parameters, and visualize how these changes impact the planning and efficiency of school schedules.

The web interface offers several essential functionalities, including the selection of the timetabling algorithm to be executed and interactive visualization. This feature displays schedules generated by the chosen algorithm in a table format, allowing clear and organized viewing. Additionally, the

dropdown menu for selecting classes provides flexibility to change contexts and view schedules for different groups, such as Alpha, Zeta, and Shaw, as required.

This approach not only facilitates navigating and analyzing class schedules, ensuring a smooth and efficient user experience, but also allows users to easily switch between weekdays using intuitive tabs. This enables quick and convenient viewing of specific schedules for each day. It has added a dialog component that allows users to add or modify crucial parameters directly for algorithm execution, providing an interactive and accessible experience. Using the json-edit-react library [1], users can view and edit JSON data directly, offering flexibility in customizing parameters as needed. These features not only facilitate precise configuration of timetabling algorithms but also enhance an intuitive user experience.

For the interface development, several modern technologies were chosen, each playing crucial roles in the project. React was the primary choice due to its ability to create reactive and scalable user interfaces, leveraging its declarative model that simplifies the construction of interactive and dynamic components [2]. Vite was adopted for rapid and efficient project development, providing an extremely fast development environment optimized for JavaScript and TypeScript applications [3]. These tools significantly accelerated the development cycle, enabling immediate response to changes during development and testing. The shadcn/ui library played a crucial role by providing stylized and reusable user interface components [4], ensuring a consistent and professional appearance of the interface. TailwindCSS was then employed for responsive and efficient styling [5], utilizing its utility-first approach to simplify the creation and customization of styles, thereby ensuring a visually appealing experience for end users. These technologies combined not only facilitated interface development but also contributed to creating a modern and robust application aligned with current software development standards.

## IV. Comparative Analysis Methodology

To evaluate the effectiveness of various timetabling algorithms, we employed a systematic methodology focused on quantifying conflicts arising from unmet constraints. The primary constraints considered were ensuring that no two classes were scheduled in the same room at the same time, ensuring that no professor was scheduled to teach more than one class simultaneously, and ensuring that the number of students assigned to a classroom did not exceed its capacity. The performance of each timetabling model was assessed using several key metrics: time to converge, best conflicts, best elite fitness, best timetable, iterations, best iteration, average conflicts history, average elite fitness history, and time to evaluate.

Conflicts were calculated by tallying instances where these constraints were not met. Specifically, conflicts for classroom timeslots were counted each time two or more classes were scheduled in the same room at the same time. Conflicts for professor timeslots were counted each time a professor was scheduled to teach multiple classes simultaneously. Room capacity conflicts were counted each time the number of students assigned to a classroom exceeded its capacity.

By tracking these metrics and analyzing the patterns in conflict resolution and fitness scores, we performed a comparative analysis of three timetabling approaches: genetic algorithms, local search algorithms, and randomized approaches. This approach allowed us to identify the strengths and weaknesses of each model in producing optimal school schedules efficiently.

## V. Results

The results should be updates done by the time this article is presented.

## References


[1] CarlosNZ. "json-edit-react", v1.2.0. GitHub, San Francisco, CA, USA, Jan. 2023. Available at: https://github.com/CarlosNZ/json-edit-react.

[2] Facebook Inc., "React: A JavaScript library for building user interfaces," [Online]. Available: https://reactjs.org/.

[3] Evan You, "Vite: Next Generation Frontend Tooling," [Online]. Available: https://vitejs.dev/.

[4] Shadcn. Available: https://ui.shadcn.com/

[5] Tailwind Labs, "TailwindCSS: Rapidly build modern websites without ever leaving your HTML," [Online]. Available: https://tailwindcss.com/.

[6] Wikipedia contributors. (2024, March 19). Local search (optimization). In Wikipedia, The Free Encyclopedia. Retrieved 20:25, July 1, 2024, from

[7] Schaerf, Andrea & Di Gaspero, Luca. (2002). Local Search Techniques For Educational Timetabling Problems.

[8] Sneha Kothari "Local Search Algorithms in AI: A Comprehensive Guide". disponível em: https://www.simplilearn.com/local-search-algorithms-in-ai-article

[9] Autor desconhecido. "Local Search". Complexica, disponível em: https://www.complexica.com/narrow-ai-glossary/local-search