Blockchain Timetabling Scheduling System

Universidade Fernando Pessoa

Bruno Miguel Batista Pereira

Faculdade de Ciência e Tecnologia

Universidade Fernando Pessoa

A thesis submitted for the degree of

Master of Science

Supervised by

Prof. Dr. Ivo Pereira and Prof. Dr. Christophe Soares

2023

Abstract

Every year, universities face a challenge with schedule generation. This task is very complicated and takes a long time because of many limitations and different factors. Each schedule's organization depends on specific constraints. Each class has its unique restrictions due to specialized subject requirements. Certain subjects require specific rooms, such as labs, rooms with projectors, or computer outlets.

Managing university schedules can be complex, especially when instructors teach multiple subjects. These subjects cannot be scheduled concurrently, and there is a maximum limit on the daily teaching hours for each instructor.

In addition to these rules, some factors affect how well a schedule works, like reducing wait times. Creating an efficient schedule is a complex challenge because we need to meet all the constraints and metrics. However, a significant added complication is user satisfaction. A schedule should be organized and flexible to meet everyone's preferences, but it is hard to please everyone.

To solve the problem, this dissertation suggests a system that balances two important aspects. It allows schedule creation while adhering to university-imposed restrictions. This system is unique because users can vote on generated schedules to show their preferences. Votes are recorded securely on a blockchain. This guarantees better reliability, transparency, and security than other methods.

To handle the system's high computational demands, was used *Graphics Processing Unit* instead of *Central Processing Unit* for scheduling. This option helps with doing more tasks at the same time and making things faster because of the better abilities of the *Graphics Processing Unit*.

The findings of this thesis indicate the system's viability in real-world scenarios. Using the GPU helped improve performance and saved resources. Incorporating a blockchain-based voting system potentially enhances schedule satisfaction, showing promise in the system's practical application.

I wish to dedicate this dissertation to my parents, girlfriend, and family for their unwavering love and support

Acknowledgements

I wish to express my sincere acknowledgments to my esteemed supervisors, Prof. Dr. Ivo Pereira and Prof. Dr. Christophe Soares. Their invaluable guidance and counsel were vital in steering me through the course of this research. I would also like to extend my gratitude to the dedicated educators within the computer engineering program, whose teachings and insights have contributed significantly to the completion of this dissertation.

I am deeply grateful to my family and friends for their unwavering support and motivation

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACO *Ant Colony Optimization*

BB *Branch and Bound*

CD-GA *Constraint-Directed Genetic Algorithm*

COPs *Combinatorial Optimization Problems*

CPU *Central Processing Unit*

CSP *Constraint Satisfaction Programming*

CSPs *Constraint Satisfaction Problems*

GA *Genetic Algorithm*

GAS *Genetic Algorithm Service*

GPU *Graphics Processing Unit*

LP *Linear Programming*

PESP *Periodic Event Scheduling Problem*

PoA *Proof of Authority*

PoW *Proof of Work*

PSO *Particle Swarm Optimization*

SA *Simulated Annealing*

SC *Smart-Contracts*

TS *Tabu Search*

VBS *Voting Blockchain Service*

Chapter 1

Introduction

This chapter provides a contextual background to the problem that has been addressed, along with the objectives and contributions realized through this thesis. The chapter concludes by outlining the structure of the document, describing the division of the thesis and the particular subjects covered in each chapter.

1.1 Context

One of the crucial issues faced by administrators and university staff in today's higher education scene is the hard and often long work of schedule generation (Martin, 2022). The importance of good schedule management cannot be overlooked, since it has a direct impact on students' university experience, productivity, and the university's overall operational efficiency (Martin, 2022). Crafting optimal timetables becomes a complicated task with different limitations and ever-changing requirements, necessitating intricate balancing acts.

The research was motivated by the substantial impact of well-optimized timetables on the educational experience. A well-planned schedule can improve student happiness, learning results, and resource usage, resulting in a more efficient and effective educational institution (Sulong *et al.*, 2023). The complexities of organizing university calendars include a plethora of elements such as teacher preferences, classroom availability, course prerequisites, and student preferences, among others.

The study's objectives are dual. Firstly, it aims to develop a system that addresses the complexities involved in creating university timetables. This system will consider various constraints and optimize the use of resources, ensuring a smooth academic experience for both students and professors. Secondly, this study seeks to enhance the transparency, reliability, and fairness of the timetable selection process through the implementation of a blockchain-based voting mechanism. This integration aims to provide a voice to all parties involved in the scheduling process, namely those who will use the schedule. It allows

them to vote for the one that best suits their needs, thereby enhancing satisfaction with the implemented timetable. Subsequently, this can lead to improved academic performance.

The primary challenge this study tackles is the intricate task of optimizing university timetable creation while adhering to numerous constraints. This task becomes increasingly challenging as universities endeavor to accommodate the diverse needs and preferences of students within the limits of available resources. Moreover, ensuring fairness in the final schedule selection is essential for maintaining the trust of all parties. This research proposes a comprehensive solution that merges inventive scheduling techniques with a transparent and secure voting system, advancing the current practices in university timetable management.

By exploring the innovative solutions presented in this thesis, it aims to provide a novel and effective approach to address the intricacies of schedule generation in a university setting, benefiting educational institutions, students, and faculty alike.

1.2 Objectives

The primary objectives of our system, and consequently this thesis, are multifaceted. Firstly, the system aims to establish a comprehensive framework for resolving timetabling problems. The goal is to use resources efficiently and improve scheduling, especially in the university. Planning carefully helps align schedules with the institution's, student's needs, improving outcomes.

The system also wants to change how decisions are made when choosing schedules. It does this by adding a blockchain voting system. This integration helps create fair decision-making processes that are democratic and transparent. The timetable selection process is improved. This makes sure the outcomes meet the needs of students, faculty, and staff. By doing this, it helps people feel like they are all part of something and that they have a say. This makes them happier and more likely to work together.

This thesis examines the architecture, development, and evaluation of this innovative system. This system provides valuable information on its use in academic scheduling. It helps make scheduling better and fairer.

1.3 Contributions

This study sets out to overcome the complex issues connected with university timetable generation while also presenting a novel blockchain-based voting system. This thesis' contributions are complex, covering the following dimensions:

- **Improving daily life**: The suggested system has the potential to have a significant impact on the daily lives of students, staff, and university administrators. It will

considerably improve students' university experiences by improving and simplifying the schedule generation process. They might expect schedules that are more in line with their preferences, minimizing disputes and increasing overall happiness. Faculty members will benefit from more efficient allocation of teaching hours, allowing them to focus on providing high-quality education. Administrators will notice better operational efficiency due to optimized resource usage and more exact scheduling. Finally, our method aspires to make academic life more manageable and gratifying for all stakeholders.

- **Blockchain usage**: The use of blockchain in academic scheduling provides a new perspective on democratizing the scheduling process. Furthermore, the use of a blockchain solution makes a step in expanding the fields where the blockchains can be used.

In summary, this study provides a disruptive alternative that goes beyond the typical constraints of academic scheduling. It improves the daily lives of the people concerned, brings previously unknown perspectives to the forefront, and achieves crucial goals in the areas of scheduling optimization and decision-making transparency. These contributions demonstrate our system's potential to transform the way academic calendars are handled and selected.

It is also important to highlight the publication of a scientific paper entitled *"Blockchain-Based Electronic Voting: A Secure and Transparent Solution"*, published in MDPI Criptography journal (Pereira *et al.*, 2023).

1.4 Document Structure

This document is divided into three main chapters, each corresponding to a crucial step in the development of the proposed solution.

Chapter 2 is the Literature Review, which discusses the research conducted to outline a plan for the proposed solution. This phase includes an explanation of various approaches that could have been taken, a comparison between them, and an explanation of why the chosen approach was selected. More specifically for the schedule generation, it discusses numerous optimization algorithms, their implementations, strengths, and weaknesses, along with a comparison between them and the rationale behind the chosen algorithm. For the blockchain voting system, we explain the fundamental terms necessary for understanding it, followed by a presentation of various systems already implemented in the market. Contextualize these systems, highlight their differences, and include a comparison with the proposed voting system. This phase also includes a comparison between centralized and decentralized systems, as well as an explanation of the differences between systems implemented on CPUs and GPUs.

Chapter 3 involves the system development, the Blockchain Timetabling Scheduling System. This phase provides a detailed explanation of the proposed system, detailing all its components, implementations, and optimization techniques used in each component.

Chapter 4 is the Empirical Evaluation of the System. In this phase, it reveals the entire flow and message exchanges that occur in the developed system, and the results it generates. It analyzed the schedules created by the system and compared them with a real schedule implemented by one university in previous years. It also, tests how computational power is relevant to the performance of the blockchain voting system. Additionally, it compares the performance of the same system implemented on a CPU and a GPU, to determine which implementation is superior performance-wise.

Finally, some conclusions and future work are presented in chapter 5.

Chapter 2

Background and Literature Review

This chapter has a detailed discussion of the challenges involved in timetable optimization and decision-making and the potential solutions.

After that, will shift toward identifying and evaluating various strategies that promise to address these challenges effectively. This section is committed to exploring a range of possible approaches. Each method will be assessed based on its ability to produce practical and optimized timetables and to facilitate a clear, fair, and inclusive decision-making process among the available options.

2.1 Problem

The search for optimization and constraint satisfaction expands across various sectors and embodies a fundamental challenge in operational planning and decision-making processes (Li & McClelland, 2022). Industries ranging from logistics and manufacturing to healthcare and education find themselves entangled in the intricate web of identifying optimal solutions that conform to a multitude of constraints and requirements (Mavrotas *et al.*, 2007). Each area has its unique factors and conditions that require custom solutions. These solutions need to be precise, flexible, and adaptable to effectively handle their specific operational challenges.

In the presence of numerous optimal solutions created by computational algorithms, a secondary yet crucial problem surfaces: the dilemma of decision-making. The multitude of possible optimized outcomes demands an intricate process of selection that resonates with the collective preferences and priorities of the stakeholders involved (El Khatib *et al.*, 2023). This highlights the essential need for a mechanism capable of facilitating informed, democratic decisions that enhance the selection process's credibility and acceptance, ensuring that the chosen solution is reflective of a consensual agreement amongst the various parties involved.

Exploring the educational sector, especially in the area of timetable creation. Here,

creating an optimized timetable requires balancing various elements such as class schedules, teacher availability, and resource allocation, all of which come with their own set of constraints and preferences (Alghamdi *et al.*, 2020). Traditional algorithmic methods in this field aim to produce numerous optimized timetables that meet these conditions and limitations. However, the task doesn't end with creating these possible schedules. It progresses to the next stage of decision-making, where the best schedule must be chosen from the generated options. This choice should closely align with the combined preferences and needs of the educational community.

The focus of this thesis is on timetabling scheduling, which embodies the problem of optimization coupled with nuanced layers of stakeholder considerations and constraints. This problem, robust in its complexity, demands a meticulous solution that organizes and coordinates multifarious elements, ensuring the efficacy and smooth flow of a university.

- **Multidimensional Optimization**: The core challenge lies in the realm of optimization, where varying elements such as faculty schedules, classroom availabilities, and course timings interweave. Each element brings forth unique constraints and considerations, necessitating a solution that adeptly navigates these variables to yield a timetable that epitomizes efficiency and feasibility (Voget & Kolonko, 1998).

- **Diverse Stakeholder Preferences**: Inherent in this scenario is a tapestry of stakeholder preferences and needs. From faculty availabilities and preferences to student course selections and optimal learning periods, the system must gracefully assimilate these variables, ensuring that the resulting schedules echo the collective needs and preferences of the academic community (Mahlous & Mahlous, 2023).

- **Dynamic Constraint Integration**: The system must possess intrinsic flexibility, enabling the dynamic integration and modification of constraints and preferences. This adaptability ensures that the system remains resilient and responsive to evolving needs, ensuring sustained relevance and applicability (Mahlous & Mahlous, 2023).

- **Democratic Decision-Making**: One primordial point of this challenge is the necessity for a democratic approach to decision-making. Given the generation of multiple optimal timetables, a transparent and inclusive mechanism is essential, allowing the university community to actively participate in the selection process (El Khatib *et al.*, 2023).

- **Blockchain-Based Voting**: To boost the integrity and reliability of the democratic process, the incorporation of blockchain technology emerges as a vital requirement. This technology would safeguard the voting process, ensuring the immutability,

transparency, and authenticity of the votes cast, thus enhancing the credibility and trustworthiness of the decision-making process (Jafar *et al.*, 2021).

Even after all of these challenges, still exist constraints that can vary for each scenario, university, and context. Here are some constraints that were taking into account:

- **Course Timing:** Each course should be scheduled in such a way that it doesn't conflict with other courses. This is particularly important for students who have required courses in their curriculum.
- **Teacher Availability:** Lecturers may have specific times when they are unavailable due to other commitments, and these need to be taken into account.
- **Room Capacity and Availability:** The number of students in a course must not exceed the capacity of the assigned room. Additionally, rooms may have specific equipment or features needed for a particular course.
- **Balancing Student Workload:** The timetable should be organized in such a way that students do not have overly concentrated periods of activity.
- **Minimizing Idle Time:** Both for lecturers and students, the timetable should be optimized to minimize periods of inactivity within the day.

In summation, the problem presents a rich tapestry of challenges, each thread weaving through the intricate landscape of academic scheduling, stakeholder considerations, democratic participation, technological integration, and user experience.

2.2 Optimization Algorithms

This section will discuss potential algorithms that could be chosen for scheduling generation. Explain their respective implementations and the scenarios where each algorithm should be used. Comparing and evaluating these algorithms will help determine the most suitable one for the specific use case of this thesis.

2.2.1 *Branch and Bound*

The *Branch and Bound* (BB) algorithm is a powerful tool in computational optimization (Shinano *et al.*, 1995), renowned for delivering exact solutions to a wide array of NP-hard optimization problems. This algorithm, rooted in the pioneering work of Land and Doig, has flourished over the years, embodying a suite of algorithms that share a foundational core solution procedure (Clausen, 2016).

Central to the BB algorithm is a tree structure that houses partial solutions, known as subproblems. The algorithm navigates through the vast landscape of potential solutions, systematically branching to dissect the problem into more manageable subproblems and applying bounding rules to prune unequivocally suboptimal regions. This meticulous process of exploration and elimination is instrumental in steering the algorithm toward the optimal solution with enhanced precision and efficiency.

The performance of the BB algorithm is intricately woven with three pivotal components: the search strategy, branching strategy, and pruning rules. These components serve as the algorithm's navigational compass, guiding the exploration of subproblems, the partitioning of the solution space, and the exclusion of suboptimal regions from further consideration.

Algorithm 1 *Branch and Bound* (Shinano *et al.*, 1995)

Require:

```
bestSolution ← initial solution
Q ← priority queue with one node
while Q is not empty do
    currentProblem ← Q.dequeue()
    if currentProblem is better than bestSolution then
        bestSolution ← currentProblem
    end if
    if currentProblem is not pruned then
        children ← expand currentProblem
        Q.enqueueAll(children)
    end if
end while
return bestSolution                    ▷ Return the best solution found
```

The *Branch and Bound*(algorithm 1) starts with a root that represents an initial solution or partial solution. This root is then "branched" into subproblems, representing different paths or decisions in the solution space. Each subproblem or "node" is evaluated to determine its bound, which is an estimate of the best possible solution that can be obtained from that node. If a node's bound is worse than the best-known solution, it is "pruned" or discarded from further consideration, as it cannot lead to an optimal solution. If a node leads to a feasible solution that is better than the current best-known solution, it updates the best-known solution. The algorithm continues branching and pruning nodes until all nodes are either pruned or lead to feasible solutions, ensuring that the best solution found is the global optimum. The efficiency of the BB lies in its ability to eliminate large portions of the solution space, reducing the number of solutions that need to be explicitly evaluated (He *et al.*, 2014).

2.2.2 *Linear Programming*

Linear Programming (LP) is a powerful mathematical tool (Dantzig, 1990) used in optimizing objective functions subject to a set of linear constraints. In a recent study by Kosheleva *et al.* (2020), a novel perspective on solving LP problems is presented, focusing on the utilization of symmetries and derivative constraints in the algorithmic process.

In traditional LP algorithms, such as the Simplex method (Dantzig, 1990), the objective is to optimize a linear objective function by navigating through the vertices of the feasible region defined by the constraints. However, the approach discussed in the paper leverages symmetries and derivative constraints to guide the optimization process.

The utilization of symmetries and derivative constraints in the algorithm allows for a more intuitive and simplified process, making it a promising approach to tackling real-life optimization problems where linear relationships and constraints are involved

LP is a mathematical method used to find the best outcome in a mathematical model whose requirements are represented by linear relationships. Applying LP in the realm of timetable generation encapsulates an optimization approach, striving to create the most efficient schedules following various constraints and objectives.

Algorithm 2 Simplex Method (a LP approach) (Dantzig, 2002)

Require:

BasicVariables ← initialize basic variables
Tableau ← initialize tableau with A, b, c
while there exists a positive coefficient in the objective row **do**
    EnteringVariable ← choose entering variable using Bland's rule
    LeavingVariable ← choose leaving variable
    if no leaving variable found **then**
        return "Unbounded"
    end if
    $pivot(Tableau, EnteringVariable, LeavingVariable)$
    $update(BasicVariables, EnteringVariable, LeavingVariable)$
end while
return $solution(BasicVariables, Tableau)$ ▷ Return the solution found

The Simplex Method(algorithm 2) is a simple implementation of the simplex Method. It starts by initializing the basic variables and the tableau using the input matrix A, and vectors b and c. The algorithm iteratively modifies the tableau, aiming to find the optimal solution to the linear programming problem. In each iteration, an entering variable is chosen based on Bland's rule (Avis & Chvátal, 1978), ensuring that cycling does not occur. A leaving variable is also selected, determining which variable will exit the basis to maintain feasibility. If no leaving variable can be found, the algorithm concludes that the problem is unbounded. Pivoting operations are then performed to update the tableau

and basic variables, moving toward the optimal solution. This iterative process continues until all coefficients in the objective row are non-positive, indicating that the optimal solution has been found. The algorithm then returns the solution, extracting it from the current state of the basic variables and the tableau.

The *Periodic Event Scheduling Problem* (PESP) (Goerigk & Schöbel, 2011), is a notable discrete problem known for its complexity and practical relevance, particularly in finding periodic timetables, an economically crucial but mathematically challenging task. The PESP is identified as NP-hard, signifying the substantial computational effort required to find even a feasible solution. Despite its complexity, successful real-world applications, such as the computation of the Dutch railway system timetable, underscore the model's practical viability and effectiveness.

2.2.3 *Tabu Search*

The TS, initially introduced by Glover (1986), is a prominent metaheuristic used to tackle combinatorial optimization problems. This algorithm operates in a systematic, iterative manner, continuously navigating through the solution space to find an optimal or near-optimal solution to a given problem.

In Mirsad Buljubasic Ph.D. thesis "Efficient local search for several combinatorial optimization problems" (Buljubasic, 2015) elucidates the application of *Tabu Search* (TS) in the realm of *Combinatorial Optimization Problems* (COPs), demonstrating its versatility and efficacy. The algorithm operates by exploring the neighborhood of solutions, making local changes, and progressively moving toward optimality. A distinctive feature of TS is its utilization of memory structures, primarily the tabu list. This list records certain attributes of recently visited solutions, preventing the algorithm from revisiting them immediately. This strategy is instrumental in enabling the algorithm to escape local optima and facilitating a more extensive and diverse exploration of the solution space (Luke, 2013).

Buljubasic (2015) work emphasizes the adaptability of TS, showcasing its applicability across a spectrum of problems, ranging from classical, well-known optimization problems to more complex, real-world industrial challenges. The algorithm's design, characterized by its dynamic search capabilities and strategic use of memory, allows it to navigate through the complexities and constraints inherent in COPs effectively. This makes TS a robust and reliable approach for finding high-quality solutions in a variety of optimization contexts, underscoring its practical relevance and utility in solving intricate optimization problems (Talbi, 2009).

The *Tabu Search* (algorithm 3) starts by selecting an arbitrary solution as the current and best-known solution. It also initializes a Tabu list, which helps remember recent solu-

Algorithm 3 *Tabu Search* (Glover, 1986)

Require:

max_iterations ▷ he maximum iterations that the algorithm can do
tabu_list_size ▷ The length of the tabu list
neighborhood_size ▷ The number of neighbors to consider

```
current_solution ← initial_solution
best_solution ← initial_solution
tabu_list ← initialize empty list
iteration ← 0
while iteration < max_iterations do
    neighbors ← generate neighbors(current_solution, neighborhood_size)
    next_solution ← select best neighbor(neighbors, tabu_list)
    if cost(next_solution) < cost(best_solution) then
        best_solution ← next_solution
    end if
    update Tabu List(tabu_list, tabu_list_size, next_solution)
    current_solution ← next_solution
    iteration++
end while
```

Ensure: *best_solution* ▷ Return the best solution found

tions and moves. It stops the algorithm from revisiting them and encourages exploration of new areas. This tabu list usually maintains a fixed dimension, and upon reaching its capacity, the oldest record is systematically removed (Glover & Kochenberger, 2003).

In each iteration, the algorithm 3 generates a neighborhood around the current solution. This neighborhood comprises solutions reachable by making minor modifications or moves from the current solution. The algorithm then selects the best move from this neighborhood, even if it doesn't improve the current solution, to promote a broader exploration of the solution space.

The Tabu list plays a crucial role in guiding the algorithm. It avoids cycling back to recently explored solutions. However, under certain conditions, moves listed as Tabu can be accepted if they meet specific aspiration criteria, such as leading to a solution better than any previously found.

The iterative process continues until specific stopping criteria are met, such as reaching a maximum number of iterations or achieving a convergence condition. Ultimately, the algorithm returns the best-found solution as the output (Glover & Kochenberger, 2003; Madureira, 2010).

An algorithm like this is widely used for timetabling problems. For instance, a study conducted at the Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak (UNIMAS), explored the heuristic orderings approach to solve the university course timetabling problem. The study aimed to reduce the number of timeslots required to allocate courses each day. The results demonstrated that the combination

of the largest degree and largest enrolment in descending order or the largest degree in descending order outperformed other heuristic orderings in generating a feasible initial solution. Moreover, the heuristic orderings approach proposed in this study required fewer timeslots than the existing software in generating the timetable (Chen *et al.*, 2022).

2.2.4 *Ant Colony Optimization*

The *Ant Colony Optimization* (ACO) algorithm has emerged as a robust technique in solving *Constraint Satisfaction Problems* (CSPs), where the objective is to find an assignment of values to variables that satisfies a set of constraints (Talbi, 2009). In the study conducted by Boxin Guan, Yuhai Zhao, and Yuan Li (Guan *et al.*, 2021), an enhanced version of the ACO algorithm is proposed, which incorporates an automatic updating mechanism, aiming to improve the solution quality and convergence speed of traditional ACO-based algorithms in solving CSPs.

In the conventional ACO algorithm, artificial ants traverse a solution space, constructing solutions and depositing pheromone trails that guide the search process. The proposed improved ACO algorithm, termed AU-ACO, innovates this process by introducing an automatic updating mechanism. This mechanism optimizes assignments by retaining excellent variable-value pairs and only optimizing non-excellent ones. This nuanced approach allows the algorithm to maintain beneficial components of the solution while enhancing others, increasing the likelihood of finding superior solutions.

The automatic updating mechanism in AU-ACO allows for a more focused and efficient search. Instead of optimizing all variable-value pairs in an assignment, the algorithm selectively optimizes those that are not excellent, thereby improving the convergence speed and enabling the algorithm to find better solutions more efficiently.

Applying this algorithm to the problem of timetable generation brings a novel approach that mimics the natural world to find optimal solutions in the complex landscape of educational scheduling. Here are some insights into the pros and cons of employing the ACO algorithm for our timetable generation issue.

The *Ant Colony Optimization* (algorithm 4) was first introduced by Marco Dorigo in 1991 in Ant System (Dorigo *et al.*, 1991). It starts by initializing essential components such as the best solution variable, the population of ants, and the pheromone matrix. The algorithm runs in iterations, where each iteration involves each ant in the population constructing a solution path. This construction is guided by the pheromone matrix, which influences the probability of choosing a particular path. An ant incrementally constructs a solution, and if the quality of the solution found by an ant is better than the current best solution, the best solution is updated. After all ants have constructed their paths for an iteration, the pheromone matrix is updated. Pheromones evaporate to simulate the natural evaporation of pheromones, and new pheromones are deposited on the paths that

Algorithm 4 *Ant Colony Optimization* (Talbi, 2009)

Require:

```
1: best_solution
2: population ← generatePopulation(NumAnts)        ▷ Generates a population of ants
3: initialize(pheromone_matrix)                    ▷ Initializes the pheromone matrix
4: iteration ← 0
5: while iteration++ < stop_citeria do
6:     for each ant in population do               ▷ Goes through all ants in the population
7:         construct_path(ant, pheromone_matrix)   ▷ Incrementally constructs a path for the ant
8:         if quality(ant) > quality(best_solution) then
9:             best_solution ← ant                 ▷ Updates the best solution found
10:        end if
11:    end for
12:    update_pheromone(pheromone_matrix, rate)    ▷ Updates the pheromone
13: end while
14: return best_solution                           ▷ Returns the best solution found
```

the ants traveled, with the amount being inversely proportional to the solution's quality. The algorithm continues iterating until a stopping criterion, such as a maximum number of iterations, is met. Finally, the algorithm returns the best solution found during the iterations, representing the approximate solution to the optimization problem (Glover & Kochenberger, 2003).

In the study Nothegger *et al.* (2012), the authors address the Post Enrolment Course Timetabling problem. They introduce an algorithm based on ACO, where artificial ants incrementally build solutions using pheromones and local information. The algorithm employs two simplified pheromone matrices to enhance convergence and maintain flexibility in guiding the solution process. When parallelized, the algorithm significantly improves solution quality. Applied to ITC2007 instances, the algorithm consistently found optimal solutions, showcasing its effectiveness and efficiency in solving this timetabling problem.

2.2.5 *Simulated Annealing*

Simulated Annealing is a probabilistic optimization algorithm (Thyer *et al.*, 1999) inspired by the annealing process in metallurgy. Annealing is a physical process used to reduce defects in a material, involving heating and controlled cooling. Similarly, *Simulated Annealing* is used to find an approximation to the global optimum of an objective function.

In the context of optimization (Kirkpatrick *et al.*, 1983), the algorithm starts with an initial solution and seeks to improve it by making small, random changes. At each step, the algorithm assesses whether the new solution is better or worse than the current

one. Unlike gradient descent methods, *Simulated Annealing* allows worse solutions to be accepted with a certain probability. This probability is influenced by a parameter known as temperature, which gradually decreases over time, mimicking the cooling process in annealing.

The temperature parameter is crucial in controlling the exploration-exploitation trade-off (Xu & Zhang, 2014). In the early stages, when the temperature is high, the algorithm is more likely to accept worse solutions, allowing it to explore the solution space more broadly and escape local minima. As the temperature decreases, the algorithm becomes more conservative, accepting only solutions that offer a significant improvement, thus exploiting the areas of the solution space that seem promising.

Simulated Annealing is versatile and can be applied to various optimization problems, including those for which the objective function is not differentiable or contains many local minima. Its ability to escape local minima makes it particularly useful for complex, high-dimensional optimization problems where traditional algorithms might struggle.

Algorithm 5 *Simulated Annealing* (Thyer *et al.*, 1999)

$current\_solution \leftarrow initial\_solution$
$current\_temperature \leftarrow initial\_temperature$
while $current\_temperature > 1$ **do**
    $new\_solution \leftarrow \text{getNeighbor}(current\_solution)$
    $cost\_difference \leftarrow cost(new\_solution) - cost(current\_solution)$
    if $cost\_difference < 0$ or $\text{random}() < e^{(-cost\_difference/current\_temperature)}$ **then**
        $current\_solution \leftarrow new\_solution$
    end if
    $current\_temperature \leftarrow current\_temperature \times cooling\_rate$
end while
return $current\_solution$

The *Simulated Annealing* (algorithm 5) starts with an initial solution and an initial temperature. It iteratively makes small random modifications to the current solution, generating neighboring solutions. For each new solution, the algorithm calculates the cost difference between the new and the current solution. If the new solution is better or meets a certain probability criterion based on the cost difference and the current temperature, it is accepted as the current solution. This acceptance probability allows the algorithm to escape local minima by occasionally accepting worse solutions. The temperature is gradually reduced in each iteration, controlling the likelihood of accepting worse solutions—high temperatures allow more exploration, while lower temperatures focus on exploitation around promising areas. The algorithm continues until the temperature reaches a predefined threshold, and the last accepted solution is returned as the approximate optimal solution (Delahaye *et al.*, 2019).

In school timetabling problem used a *Simulated Annealing* (SA) approach to resolve it (Abramson *et al.*, 1997). The reheating schedules, particularly, are nuanced, with some

utilizing a geometric approach and others applying a calculated temperature for the reheating point. Through comprehensive computational experiments, the paper Abramson *et al.* (1997) unveils that the fourth reheating scheme consistently outperforms others, delivering superior quality solutions in a more time-efficient manner for the school timetabling problem. This illustrates the algorithm's adaptability and effectiveness in navigating the solution space and overcoming local optima to find satisfactory timetabling arrangements.

2.2.6 *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995) is a heuristic that has been effectively applied to various optimization problems, including CSPs, as discussed by Lin (2005). PSO is a population-based optimization technique inspired by the social behavior of birds flocking or fish schooling. In the context of CSPs, which are generally NP-hard and applicable to many practical problems, PSO has been modified and extended to solve n-ary CSPs effectively.

Lin's research introduces new particle swarm algorithms that are tested on practical configuration problems and traditional n-queens problems. These algorithms incorporate modifications that enhance their effectiveness and efficiency in solving CSPs. For instance, the algorithms combine zigzagging particles and repair-based CSP-solving methods, which have been found to perform best among the studied algorithms.

The application of PSO in CSPs, as presented in (Lin, 2005), demonstrates a promising approach to finding satisfactory solutions to these problems. The adaptability and robustness of the PSO-based algorithms allow them to navigate the solution space comprehensively, increasing the likelihood of finding optimal or near-optimal solutions to the CSPs.

The implementation of the *Particle Swarm Optimization* (algorithm 6). Starts by initializing a swarm of particles with random positions and velocities within the search space. Each particle represents a potential solution to the optimization problem. Throughout the algorithm's iterations, each particle continuously updates its position and velocity based on its own best-known position (pBest) and the best-known position in the entire swarm (gBest). The velocity update considers the current velocity, the distance to the particle's pBest, and the distance to the swarm's gBest, ensuring a balance between exploration and exploitation in the search space. If a particle finds a better position that improves the objective function value, it updates its pBest. Similarly, if a particle's pBest is better than the current gBest, the gBest is updated. The algorithm iterates a predefined number of times, and the final gBest represents the approximate optimal solution to the problem, reflecting the collective exploration and learning of the swarm (Jain *et al.*, 2018).

Algorithm 6 *Particle Swarm Optimization* (Kennedy & Eberhart, 1995)

Require:

$particles \leftarrow$ initializeParticles($numParticles$)
$gBest \leftarrow$ getGlobalBest($particles$)
for $i = 1$ to $numIterations$ **do**
    for each $particle$ in $particles$ **do**
        $pBest \leftarrow$ getPersonalBest($particle$)
        $velocity \leftarrow$ updateVelocity($particle, pBest, gBest$)
        $position \leftarrow$ updatePosition($particle, velocity$)
        if fitness($position$) $<$ fitness($pBest$) **then**
            $pBest \leftarrow position$
        end if
        if fitness($pBest$) $<$ fitness($gBest$) **then**
            $gBest \leftarrow pBest$
        end if
    end for
end for
return $gBest$

2.2.7 *Genetic Algorithm*

Genetic Algorithm (GA) (Holland, 1992) is a family of computational models inspired by natural evolution. This algorithm encodes a potential solution to a specific problem as a chromosome-like data structure and applies operators such as mutation and crossover to these structures to guide the search toward the best solution.

GA are particularly well-suited for optimization and search problems (Wright, 1991) where the solution space is vast, complex, and poorly understood. They operate on a population of potential solutions applying the principle of survival of the fittest to produce better approximations to a solution. At each generation, the fitness of every individual in the population is evaluated, and multiple individuals are stochastically selected based on their fitness, modified (recombined and possibly mutated) to form a new population, which becomes current in the next iteration of the algorithm (Lee & Kim, 2005).

The GA is descibred in algorithm 7 and the various steps are (Holland, 1992):

1. **Initialization**: A population of random individuals is generated.

2. **Fitness evaluation**: Each individual in the population is evaluated based on its fitness function, which considers various constraints and preferences.

3. **Selection**: A set of individual is selected based on their fitness.

4. **Crossover**: The selected individuals are combined to generate a new set of timetables.

5. **Mutation**: The new set of individuals is subjected to mutation, where certain genes (i.e., classes, instructors, or classrooms) are randomly changed.

6. **Elitism**: The best individuals from the previous generation are preserved in the current generation to ensure convergence toward the optimal solution (Pandey *et al.*, 2014).

7. **Termination**: The algorithm terminates when a stopping criterion is met, such as a maximum number of generations or a satisfactory fitness level.

Algorithm 7 *Genetic Algorithm* (Holland, 1992)

Require: $Generations$ ▷ The maximum iterations that the algorithm can do
Require: $Population$ ▷ The number of optimal solutions per generation
Require: $MinimumScore$ ▷ The minimum score to the schedule by approved
Require: $Nsolutions$ ▷ The number of approved schedules to the program stopped
  $currentGeneration = 0$
  $appreovedSchedules = 0$
  while $currentGeneration < Generations || appreovedSchedules < Nsolutions$ **do**
    function INITIALIZATION(constrains)
    end function
    function FITNESS EVALUATION(generated schedules)
    end function
    function SELECTION(generated schedules, Minimum Score)
    end function
    $approvedSchedules = getAprovedSchedules(generatedschedules)$
    if $approvedSchedules \geq Nsolutions$ **then**
      $EndAlgorithm$
    end if
    function CROSSOVER(generated schedules)
    end function
    function MUTATION(generated schedules)
    end function
    function ELITISM(generated schedules)
    end function
  end while

2.2.8 Comparation between optimization algorithm

The task of university timetable generation is a multifaceted problem that encompasses a labyrinth of optimization and constraint satisfaction challenges. In exploring the pantheon of algorithms available to tackle this issue, several contenders emerge, each with its unique strategies and subtleties.

The comparative Table 2.1 presents an evaluation of the discussed optimization algorithms, each analyzed in the context of university timetable generation through a set of defined criteria. The exploration ability of an algorithm delineates its efficacy in navigating

through the solution space, emphasizing its capability to evade local optima and gravitate towards a global optimum. Concurrently, solution precision underscores the algorithm's ability to generate accurate and high-quality solutions, embodying the reliability and validity of its outputs. Adaptability to changes is a crucial metric, reflecting an algorithm's resilience and flexibility in adjusting to modifications and uncertainties within the problem's parameters or constraints. In parallel, the criterion of parameter tuning delves into the algorithm's susceptibility to variations in its internal parameters, indicating the level of tuning requisite for optimized performance. Scalability emerges as a pivotal factor, representing the algorithm's proficiency in maintaining performance integrity in the face of escalating problem sizes and complexities. Lastly, suitability for timetabling provides a comprehensive assessment, encapsulating an algorithm's overall effectiveness and applicability in addressing the specific challenges intrinsic to university timetable generation. Through a synthesis of these evaluative measures, the table facilitates a nuanced selection process, enabling the identification of an algorithm that resonates most harmoniously with the multifaceted demands of timetable optimization.

Table 2.1: Comparative analysis of the algorithms.

Algorithm	Exploration Ability	Solution Precision	Adaptability to Changes	Parameter Tuning	Scalability	Suitability for Timetabling
GA	High	Medium-High	High	Medium	High	High
PSO	Medium-High	Medium	Medium	High	Medium	Medium
SA	Medium	Medium	Medium	Medium-High	Medium	Medium
LP	Low	High	Low	Low	Medium	Low-Medium
ACO	Medium	Medium	Medium-High	High	Medium	Medium-High
BB	Low-Medium	High	Low	Low	Low	Low-Medium
TS	Medium-High	Medium	High	Medium-High	Medium-High	Medium-High

Table 2.1 columns are:

- **Exploration Ability:** How well the algorithm explores the solution space and escapes local optima.
- **Solution Precision:** The accuracy and quality of the solutions produced.
- **Adaptability to Changes:** How well the algorithm can handle dynamic changes and uncertainties in the problem.
- **Parameter Tuning:** The sensitivity of the algorithm to its parameters and the level of tuning required.
- **Scalability:** How well the algorithm performs as the problem size or complexity increases.
- **Suitability for Timetabling:** The overall appropriateness of the algorithm for university timetable generation, considering all factors.

GA are very effective because they are flexible and can work through a wide range of solutions. They create a changing environment that helps improve the solutions over time, which is useful for managing the many restrictions common in university scheduling. GAs keep a variety of solutions available, preventing them from settling too quickly on a less-than-best option, a common problem in other optimization methods like PSO and TS.

PSO brings to the table a collaborative approach, where potential solutions, represented as particles, navigate the search space collectively. However, PSO tends to be sensitive to parameter selection and might succumb to premature convergence, limiting its applicability in the volatile domain of timetable generation.

SA and other metaheuristic algorithms like ACO and TS offer robust mechanisms to escape local optima. SA, for instance, employs a parallel search strategy improving the algorithm's exploration capabilities. However, managing the complexity of parallelization and communication overhead could pose significant challenges.

LP, and specifically the Simplex Method, is fundamentally powerful for problems with well-defined linear constraints and objectives. Yet, the highly discrete and non-linear nature of timetable generation nuances somewhat blunts LP's effectiveness in this realm.

BB exhibits prowess in providing exact solutions through a systematic exploration of the solution space. Nonetheless, its exhaustive nature could lead to computational inefficiencies, making it less suitable for the dynamic and constraint-ridden landscape of timetable generation.

Upon careful consideration, GAs conspicuously resonates as the most promising avenue for our university timetable generation problem. Their evolutionary nature, combined with a robust mechanism to navigate through a multitude of solutions, aligns seamlessly with the intricate requirements of our optimization problem. GA's adaptability to a fluctuating landscape of constraints and objectives further solidifies its position as the aptest algorithmic choice for this endeavor.

2.3 Blockchain concepts

This section, will explain some fundamental concepts that are essential for a comprehensive understanding of this thesis. It's crucial to establish a solid foundation by grasping these basic principles. This groundwork will ensure a more informed exploration of the topic as it progresses further into the discussion.

Blockchain

Blockchain technology is a distributed, decentralized digital ledger that allows for secure data storage and exchange. It is an innovative breakthrough that has grown in promi-

nence since the appearance of Bitcoin, the first decentralized digital currency, and has the potential to disrupt numerous businesses (Nakamoto, 2009).

> *"A blockchain is essentially a distributed database of records or public ledger of all transactions or digital events that have been executed and shared among participating parties"* (Puranik, 2021).

A Blockchain is a network of interconnected blocks, each containing a cryptographic hash of the preceding block and a set of transactions. The cryptographic hash function is an essential part of blockchain technology because it ensures that any changes made to a block are discovered and rejected by the network. This makes the blockchain tamper-proof, as any modifications to one block require changes in all following blocks (J Michael, 2018).

Blockchain technology is decentralized, no single entity controls the network, allowing for greater transparency and security. The network is usually maintained by a distributed network of nodes, each with a copy of the blockchain. Mining is a process that validates and adds transactions to the blockchain by solving challenging mathematical puzzles to verify the transaction and add it to the blockchain (Wang *et al.*, 2019). In cryptocurrency, since it uses a blockchain that is decentralized and distributed, there is no need for intermediaries such as banks or financial institutions to facilitate transactions. This has the potential to significantly cut transaction costs while increasing efficiency, especially in businesses such as banking (Wang *et al.*, 2020).

Smart-contracts

Smart-Contracts, which are self-executing programs running on blockChain platforms, have gained significant attention due to their potential to automate contractual agreements (Almakhour *et al.*, 2020; Krichen *et al.*, 2022). However, *Smart-Contracts*s are also prone to errors and vulnerabilities that can result in significant financial losses. Therefore, formal methods have been proposed as a solution for verifying the correctness and security of *Smart-Contracts* (SC)s. By mathematically proving the correctness of SCs, formal methods can increase confidence in their performance and ensure that they function as intended. The use of formal methods for SS verification is an active research area with promising results, and it is expected to become a crucial component of blockchain technology.

Transactions

Transactions refer to the exchange of assets or information between parties, which can be recorded in a blockchain. As described in the article "Towards Anonymous, Unlinkable, and Confidential Transactions in Blockchain" (Singh *et al.*, 2018), transactions in a

blockchain can be designed to be anonymous, unlinkable, and confidential. This means that the identities of the parties involved in the transaction can be hidden, the transaction cannot be linked to other transactions, and the details of the transaction can be kept private. Transactions are a fundamental component of blockchain technology, allowing for secure and transparent exchanges without the need for intermediaries such as banks.

2.3.1 Blockchain Systems

Voting systems have been an integral part of democratic societies for centuries. The emergence of blockchain technology has created new prospects for developing safe and transparent voting systems. Many blockchain-based voting systems have arisen in recent years, each with its own set of advantages and drawbacks. This section will go through some of the most well-known blockchain-based voting systems on the market

Horizon State" (Skella & Naamani, 2017) is a blockchain-based voting system that uses blockchain technology to enable safe, transparent, and efficient voting. The method enables voters to vote in elections using their cell phones or laptops, making it simple and accessible to all. Horizon State's underlying blockchain technology is Ethereum, which provides immutable and transparent evidence of all transactions. The system is planned to be simple to use, with a straightforward user interface that takes voters through the voting process.

Another blockchain-based voting system that aspires to improve the way that people vote is FollowMyVote (Ernest & Hourt, 2013) . The system is supposed to be safe, transparent, and accessible to all users. Follow My Vote employs blockchain technology to verify that each vote is correctly saved and cannot be manipulated. Voters may vote from anywhere using their cell phones or laptops, making it simple and accessible for everyone. Follow My Vote is based on the BitShares blockchain and provides a quick and efficient voting mechanism.

Voatz (Sawhney, 2015) is a blockchain-based voting system that ensures voter identity through biometric authentication. The system is meant to be safe, transparent, and accessible to all users. Voatz employs blockchain technology and biometric verification to ensure that each vote is correctly saved and cannot be tampered with. Voters may vote from anywhere using their cell phones or laptops, making it simple and reachable for everyone. Voatz is a scalable and efficient voting platform built on the Hyperledger Fabric blockchain.

Blockchain-based voting systems have the potential to change the way that people vote forever. These mechanisms are meant to be safe, transparent, and accessible to anyone. They use blockchain technology to verify that each vote is correctly saved and cannot be tampered with. While each of the initiatives mentioned above has its own set of advantages and disadvantages, they all represent an important step forward in the development

of safe and transparent voting systems.

2.3.2 Blockchain Voting Approaches

Blockchain technology has grown in popularity in a variety of industries, including voting systems (Al-Jaroodi & Mohamed, 2019). The blockchain is a distributed ledger that enables the construction of tamper-resistant, transparent, and unchangeable records. The blockchain can establish a system that is very resistant to manipulation and fraud by employing cryptographic algorithms to safeguard transactions. Numerous researchers have proposed blockchain-based voting systems to improve election security and transparency. The potential for openness and security is one of the primary benefits of blockchain-based voting systems. According to (Khan *et al.*, 2018a), blockchain technology has the potential to improve voting security and transparency by enabling tamper-proof recordings of each vote.

Similarly, (Jadhav, 2020) contends that blockchain-based voting systems can eliminate traditional voting systems concerns such as hacking, tampering, and fraud.

The Avalanche blockchain protocol (Tanana, 2019), is projected to provide fast and secure transactions for large-scale distributed computing systems. It introduces the consensus algorithm called Avalanche, which enables nodes in the network to quickly reach a consensus on the state of the ledger. The algorithm relies on a random sampling mechanism to select a small subset of nodes for the consensus process. The Avalanche protocol is planned to be highly scalable and can handle complex SC efficiently. The SC concepts will be further discussed and reviewed in Section 2.3.

(Hegadekatti, 2016) proposed "Democracy 3.0" which which stored votes on a public blockchain and was supposed to be transparent, anonymous, and verifiable. The author suggested a proof-of-stake consensus algorithm that allowed for low-cost transactions and high throughput, making it suited for use in large-scale elections.

Jsang and Endresen, who devised a mechanism dubbed "Agora Voting" made a more recent suggestion in 2018 (Agora, n.d.). This system employs a permissioned blockchain, which enables secure and transparent voting. The authors proposed a consensus technique based on proof of authority that is both secure and scalable. While these approaches have shown potential, deploying blockchain-based voting systems still presents hurdles.

Scalability is one of the most difficult issues. Blockchain-based solutions can be slow and costly, making them difficult to scale up for large-scale elections. Another difficulty is the issue of voter privacy. Chauhan et al. argue that while blockchain-based systems can provide openness and immutability, they can potentially jeopardize voter anonymity if not appropriately handled (Chauhan *et al.*, 2018).

Regarding these difficulties, several academics, like (Sontakke, 2017) and (Cao *et al.*, 2019), have proposed hybrid systems that combine the advantages of blockchain tech-

nology. For example, both recommended using blockchain technology for vote counting while employing traditional techniques for voter authentication and verification. For instance, (Moubarak *et al.*, 2018) suggest that blockchain-based voting systems are vulnerable to attacks such as 51%. These can jeopardize the system's security. This happens when a single entity or group of entities controls more than half of the network's computing or hashing power. This means that they can potentially manipulate transactions.

Finally, blockchain-based voting systems have the potential to improve vote integrity and transparency. Yet, there are still issues to address, such as scalability and voter privacy. Further research and development are required to produce a strong and reliable blockchain-based voting system suitable for large-scale elections.

2.3.3 Comparing Blockchain projects

This subsection compares all the previously introduced blockchain projects. Showing their similarities and differences, drawing comparisons among themselves and with the developed system.

Table 2.2: Comparison of blockchains project.

Feature	**Horizon State (Skella & Naamani, 2017)**	**FollowMyVote (Ernest & Hourt, 2013)**	**Voatz (Sawhney, 2015)**
Voter anonymity	✓	✓	✓
End-to-end verifiability	✓	✓	✓
Accessibility	✓		✓
Security	✓	✓	✓
Scalability			✓
Transparency	✓	✓	✓
Decentralization			✓
Hybrid consensus mechanism			✓
Scheduler generation			
Register customized constraints			

Table 2.2 describes the comparison between three existing blockchain-based voting systems: Horizon State (Skella & Naamani, 2017), FollowMyVote (Ernest & Hourt, 2013), and Voatz (Sawhney, 2015). These blockchain-based voting systems have similar aims of assuring safe and transparent voting procedures. Each system has distinguishing qualities that set it distinct from the others:

- **Voter anonymity**: refers to the principle that a voter's identity should be kept secret and not be revealed to anyone, including the election officials. In other words, it means that no one should be able to link a particular vote to a specific voter (Fusco *et al.*, 2018).

- **End-to-end verifiability**: is a property of e-voting systems that allows voters to verify that their votes have been correctly recorded and counted. It means that voters can check that their votes have not been altered or tampered with during

the voting process and that they have been included in the final tally (Khan *et al.*, 2018b).

- **Accessibility**: is the ease with which one person was to access and use one system, regardless of their abilities. That means how easily can the voters vote and understand what is happening in each state of the system (Jafar *et al.*, 2021).
- **Security**: refers to the measures or procedures taken to protect something from threats or unauthorized access. That is how hard it is for someone to violate our system. For example, how hard it is to voter manipulate the election. The harder the work, the more secure the system is (Khan *et al.*, 2018b).
- **Scalability**: It means the ability of a system to handle increasing amounts of data or workload without compromising its performance. This means that the bigger the poll, the more scalable the system needs to be. This measure copes with the number of voters the system handles at the same time (Abuidris *et al.*, 2019).
- **Transparency**: represents the quality of the system state being easily visible or understood. The processes and transactions carried out within the system are easily traceable, and the information related to them is readily available to all authorized voters (Abuidris *et al.*, 2019).
- **Decentralization**: refers to the process of distributing power or control away from a central authority or entity, and instead distributing it among multiple nodes or participants in a network. Thus, multiple nodes have to validate a transaction (voters vote), before being approved and registered (Abuidris *et al.*, 2019).
- **Hybrid consensus mechanism**: is a combination of two or more methods used to achieve consensus in a blockchain network. It is planned to mitigate some weaknesses of a specific consensus algorithm, by changing that part of the algorithm with another algorithm, to make it as secure and efficient as possible. The consensus in a blockchain is the process of achieving agreement among all the nodes, for example, approving a transaction (Abuidris *et al.*, 2019).
- **Scheduler generation**: relates to the process of creating an optimal schedule, taking into account all the constraints created.
- **Register customized constraints**: concerns the ability of customization that a user has to customize the schedules and the configuration of the blockchain itself.

In terms of voter anonymity, FollowMyVote (Ernest & Hourt, 2013) and Voatz (Sawhney, 2015) provide end-to-end encryption and blockChain anonymity, while Horizon

State offers encrypted and private voting but not necessarily anonymity. End-to-end verifiability is a vital feature provided by all three systems, allowing voters to prove that their vote was correctly counted. Accessibility is also a goal for these systems, with Voatz having mobile voting capabilities and Horizon State and FollowMyVote providing accessible voting alternatives for those with impairments. The security of any voting system is critical, and all three blockchain-based systems use powerful encryption technology to protect against hackers and manipulation. Horizon State (Skella & Naamani, 2017) and Voatz(Sawhney, 2015) have an advantage in terms of scalability, with Horizon State's (Skella & Naamani, 2017) hybrid consensus method and Voatz's (Sawhney, 2015) proprietary mobile voting technology both enabling more efficient and quicker vote processing.

Transparency is a feature shared by all three of these systems, with each giving immutable recordings of the voting process. Another crucial consideration is decentralization, with FollowMyVote (Ernest & Hourt, 2013) and Voatz (Sawhney, 2015)employing a completely decentralized method, while Horizon State (Skella & Naamani, 2017) employs a more centralized architecture with distributed components.

In terms of security, both sides employ encryption and digital signatures, but the proposed solution adds a layer of security through its hybrid consensus mechanism. This makes it less prone to single-point-of-failure attacks and increases its overall security.

In terms of cost, the State-of-the-Art Solutions are expensive due to their high energy consumption.

Finally, accessibility is limited in the first system due to its technical complexity, while the proposed solution uses a user-friendly interface to increase accessibility.

2.4 Centralized and Decentralized approaches

Centralized and decentralized approaches stand as two distinct models, each bearing its unique attributes and consequences. Centralized systems Control is concentrated in one authority, which makes all decisions and operations. Centralizing decision-making can lead to quicker and more consistent decisions. It can also improve efficiency by reducing redundancy. If something goes wrong or someone attacks the central entity, the whole system could stop working. When the system grows, it can be hard to handle. It might need big changes to keep up (Pasupulati & Shropshire, 2016).

Decentralized approaches allow groups to work separately, promoting creativity and flexibility. Decentralization makes the system stronger. If one part fails, the whole system can still work. Decentralized systems can be more flexible and scalable. You can make changes or add more to certain parts without affecting the whole operation. But without one central authority, coordination and decision-making become more complex and difficult (Pasupulati & Shropshire, 2016).

On the Figure 2.1, the one on the left, Centralized, shows a system where all users directly connect to a singular central application. This application is directly linked to a database. In this model, all data and transactions flow through the central core. This can offer advantages in terms of data consistency and control but also introduces single points of failure and potential bottlenecks, as everything relies on the central system (Alfain *et al.*, 2022).

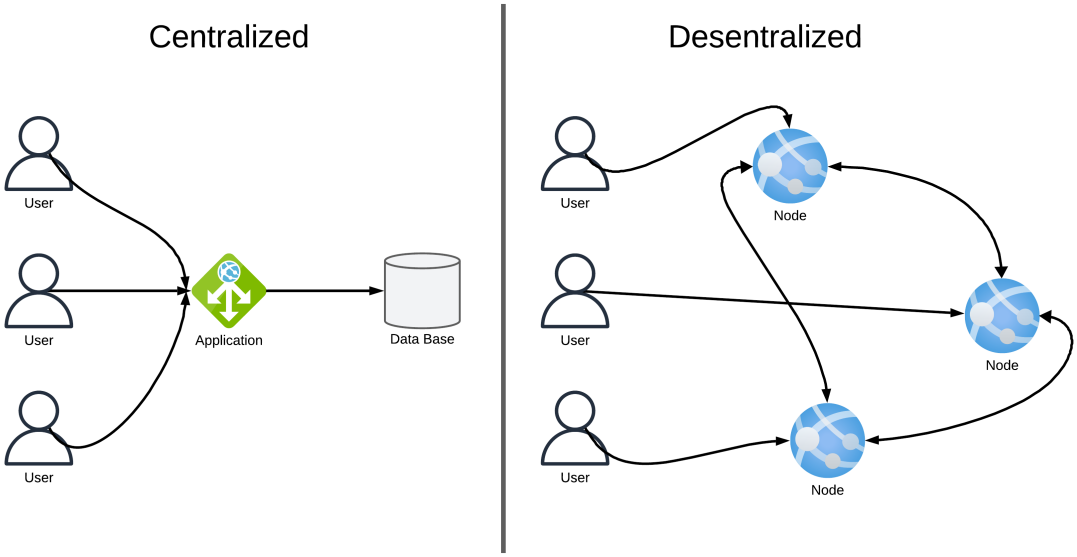


Figure 2.1: Centralized (left) VS Decentralized (right)

On the Decentralized side (Figure 2.1), the one on the right, rather than having a single central application, users connect to multiple nodes. These nodes are interconnected, allowing data and transactions to flow through multiple routes and access points. There's no single point of control or failure, which can offer increased resilience and redundancy to the system. This architecture can be more robust against failures and attacks since information is spread across various nodes (Alfain *et al.*, 2022).

To summarize, the centralized system has control and access at one point. The decentralized system spreads control and access to many points. This reduces the risk of failure.

Chapter 3

Blockchain Timetabling Scheduling System

This chapter discusses both the functional and non-functional requirements. It will provide a detailed explanation of the implementation of each component, specifying all the optimization techniques, the logic behind the timetable generation, and the blockchain.

3.1 Functional and Non-Function Requirements

The essence of developing a robust, efficient, and user-friendly system lies fundamentally in the identification and clear articulation of its requirements. This chapter aims to elaborate on the various functionalities that the proposed system seeks to fulfill. These requirements arise from the need to tackle the intrinsic challenges associated with timetable generation and to incorporate a democratic, transparent decision-making process through voting.

The functional requirements are:

- **FR01- Timetable Generation**: The system must competently generate multiple optimized timetables, taking into consideration the constraints and preferences intrinsic to the context that is involved.
- **FR02- Constraint Handling**: The system should allow for the seamless integration of a multitude of constraints such as room availability, teacher schedules, and specific time slots, ensuring flexibility and adaptability, for this example of the university context.
- **FR03- Voting System**: A secure and user-friendly voting system should be integrated, allowing stakeholders to participate in selecting the most suitable timetable from the generated options.

- **FR04- Blockchain Integration**: To enhance the security and integrity of the voting process, the system must incorporate blockchain technology, safeguarding the transparency and credibility of the voting outcomes.
- **FR05- Messaging Flow**: The system should be equipped with a communication mechanism ensuring efficient interaction between its various components and the users, thereby ensuring that information flows seamlessly.
- **FR06- Adaptability**: The system must be able to adapt itself to the most various contexts, it must exhibit adaptability, allowing for convenient modifications and adjustments to the timetables as necessary.
- **FR07- User Authentication**: A robust user authentication mechanism must be in place, ensuring that only authorized individuals can engage in the voting and timetable modification processes.
- **FR08- Voting**: The user must be able to vote, but only in the votes that are allowed.
- **FR09- Access voting State**: The user should be capable of seeing the status of the votes, but shouldn't be able to see the other users' votes, if anonymous voting is on.
- **FR10- Flow creation**: The user must be able to parameterize how the timetables are created, and what constraints should be applied.

The non-functional requirements are:

- **NFR01- Reliability**: The system should be reliable, ensuring consistent performance and availability to meet the users' needs effectively.
- **NFR02- Scalability**: Designed to handle a varying number of users and data, the system should exhibit scalability, ensuring it performs optimally under different loads.
- **NFR03- Security**: Essential security measures must be integrated into the system's design, safeguarding sensitive data and ensuring the overall integrity of the system's operations.
- **NFR04- Performance**: The system should be optimized to ensure swift responses and efficient performance, enhancing the user experience and ensuring timely execution of tasks.

The specified requirements are crucial in guiding the development process of the system, guaranteeing that it is carefully customized to address the distinct demands and issues related to university timetable generation. By fulfilling these criteria, the system will be ready to provide an improved, safe, and user-friendly solution, greatly boosting the processes of generating and selecting timetables in a university environment.

3.2 Architecture of the System

This section provides a more comprehensive analysis of the development system. Analyzes each individual component, providing detailed explanations of its purpose and how it is implemented. Furthermore, it examines the methodologies employed to enhance the efficiency of the system, clarifying the interdependence of its components and providing a comprehensive analysis of the information flow.

When delving into the development system, it consists of three crucial components (Figure 3.1): the API, the Genetic Algorithm service, and the Voting Blockchain service. Every element is carefully designed to carry out separate but interconnected tasks, coordinating an integrated sequence of actions resulting in a robust functional system customized for the most efficient production and secure voting procedures.

The API functions as the primary access point, providing a user-friendly interface through which users can interact with the system. Its responsibilities include collecting schedule limitations and voting criteria, optimizing the voting procedure, and presenting poll outcomes. The API provides users with a means to actively participate in voting, interact with scheduling, and directly interface with the underlying blockchain, ensuring that user experiences are user-friendly and straightforward.

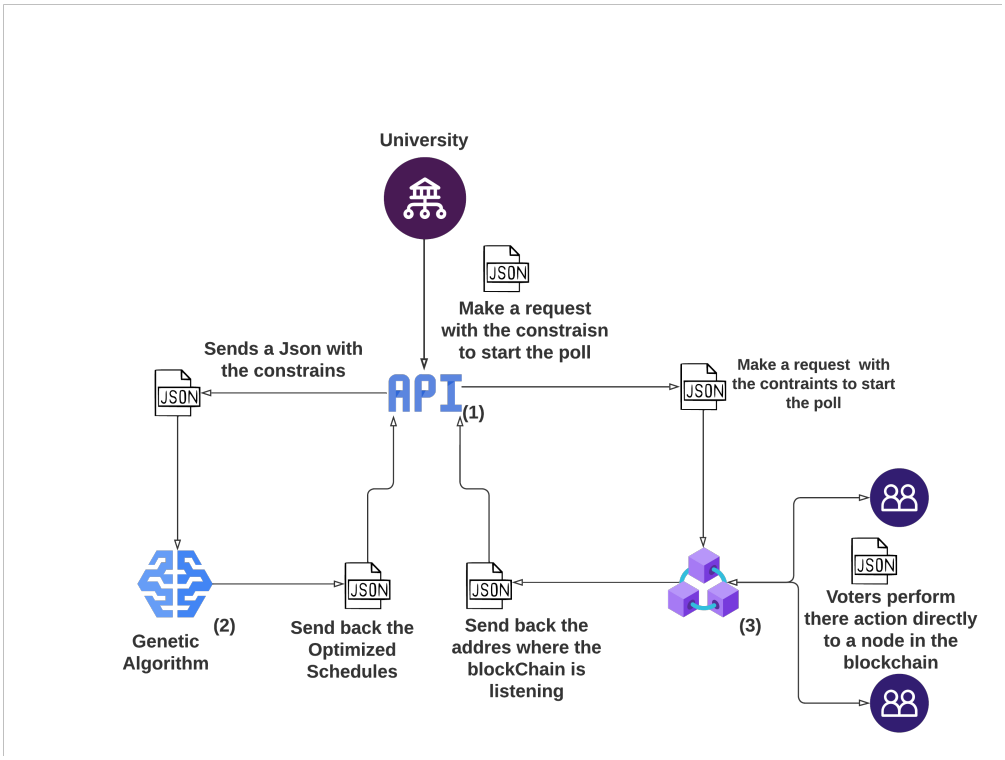


Figure 3.1: System Overview

(1)- API (2)-Genetic Algorithm Service (GAS) (3)- Voting Blockchain Service (VBS)

The *Genetic Algorithm Service* is the central point where the complex operations of schedule generation occur. Equipped with a *Genetic Algorithm Service*, this component carefully analyzes constraints obtained via the API, efficiently navigating through the complexity of optimization. The *Genetic Algorithm Service* acts heuristically, systematically searching extensive solution spaces to build schedules that respect to specified restrictions and preferences, demonstrating adaptability and precision.

The *Voting Blockchain Service* stands as the guardian of integrity and security within the system. It embodies a decentralized network of nodes, each operating in isolated environments to boost security. This architecture guarantees that the voting process is carried out with the highest level of integrity and protects the logic and data of the voting system from unauthorized access and tampering. As a result, it maintains the integrity of the voting procedures and ensures the accuracy and reliability of the results.

Together, these elements form a complex and innovative system, with each one having a crucial function in coordinating a resilient framework that efficiently manages scheduling and voting processes while ensuring the security and reliability of operations.

3.2.1 Module 1: Connect the User Interface to the Schedule and Voting System through an API

The API serves as an intermediary between users and the system, ensuring seamless and effortless operation. It is specifically engineered to be easily understandable and convenient, facilitating individuals in casting their votes and accessing timetables without any difficulty or inconvenience. The API collects important information such as voting rules and constraints, making sure that everything is organized and accurate. It also shows the poll results, allowing users to see the outcomes quickly and clearly. Users can vote and interact with the blockchain through the API, making it a key part of the system. This makes the whole process more approachable and easy to use, ensuring that everyone can participate and use the system's features without difficulty.

3.2.2 Module 2: Find an optimal Schedule using a Genetic Scheduling System

The GAS plays a crucial role as the main computational engine in creating near-optimal schedules. The system operates in perfect synchronization with the API, creating a harmonious interface that improves the overall functioning and efficiency of the system. The API serves as a medium, transmitting crucial information, such as limitations and voting parameters, to the GAS. The sharing of information is made easier by using well defined protocols and data structures, which guarantee the accuracy and usability of the data for computer operations.

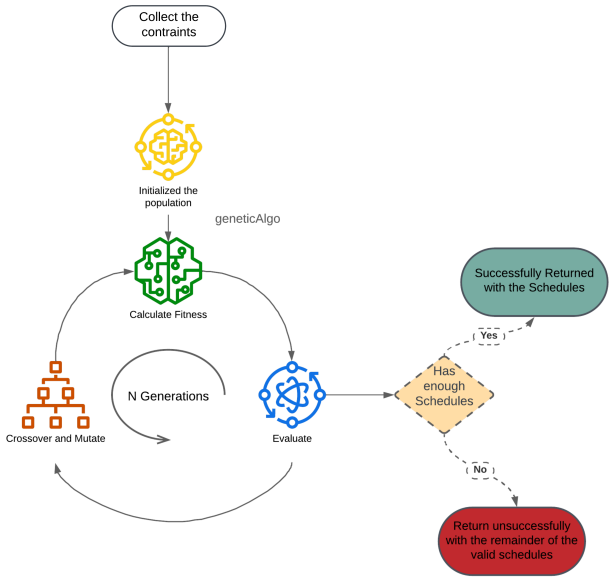


Figure 3.2: Genetic Algorithm

The second component operates as defined in Figure 3.2. Initially, it starts by loading the data of classes, rooms, schedules, professors, and constraints. Subsequently, the population is initialized, essentially an array of phenotypes, where each position of this array represents a generation. Phenotypes are then initialized, which, in our context, represent all university schedules. These are depicted by a three-dimensional matrix of irregular rows, columns, and depths, where columns represent the years of the course (1st, 2nd, and 3rd years), rows represent the semesters of each year and the depth represents the classes. Each position in this matrix represents a Gene, which encapsulates a regular matrix where rows correspond to possible scheduling hours (9h,...,23h) across potential days (Monday,..., Friday), and columns depict available rooms. Beyond initializing the schedule matrix in the gene, an array of free positions is also initialized, representing all potential slots available for scheduling (Figure 3.3).

Subsequently, data validation occurs within the gene, verifying, for instance, the feasibility of arranging all participants with their particular lengths, among other possible validations. These validations vary depending on the constraints specific to each scenario. Subsequently, it is necessary to prioritize the themes by considering their restrictions and durations. The purpose of this prioritizing is to minimize the occurrence of idle periods during the gene initialization procedure.

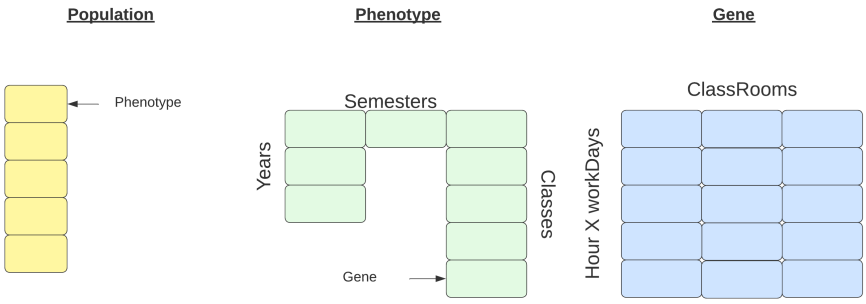


Figure 3.3: Data Structures

With this ordering, gene population commences. A random position from the array of free positions is selected, and availability is validated considering the subject's duration since a subject might require multiple consecutive slots. If space is available, the subject is positioned in the gene matrix, and the corresponding positions are marked as occupied, this process continues until all subjects are allocated. Post-allocation and reordering strategies are applied.

The first strategy involves managing unplaced subjects. If there are subjects left unplaced, the days with sufficient free spaces are identified, and subjects are rearranged to accommodate the unplaced ones. This process might involve moving subjects around to make enough consecutive free slots available, as shown in Figure 3.4.

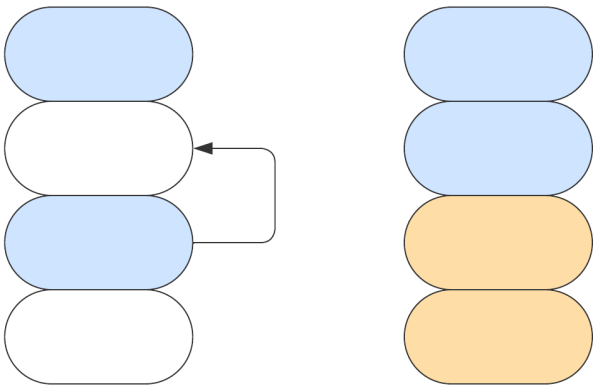


Figure 3.4: Reorganize Subjects

Subsequent strategies focus on improving fitness. The second strategy involves reorganizing subjects to minimize dead time. It involves repositioning subjects to either let students leave earlier or to create more extended lunch hours, always respecting constraints. This strategy can vary based on the specific constraints involved.

The third strategy is room reorganization. This strategy aims to minimize room changes for classes and involves coordination among all genes of a phenotype to prevent

scheduling two classes in the same room simultaneously. A matrix of rooms and available times is created, and genes 'rent' rooms as per the subject specifications, ensuring no overlaps occur.

It is essential to note that the initialization process of each gene occurs in parallel, requiring any shared data structures to be thread-safe. After initialization, the fitness calculation process starts, where each gene's validity and quality are assessed. Genes begin with a perfect score, decrementing as they fail to meet soft constraints, ensuring that all meet at least the hard constraints. The fitness scores of all genes are summed up, giving the phenotype's fitness score.

The population's first evaluation then takes place, considering variables like a maximum number of generations, the minimum score for a phenotype to be considered valid, and the number of valid phenotypes required for the program to terminate successfully. These variables are set by the user, and based on these, the program decides whether to continue to the next generation or terminate, saving the valid schedules found so far.

In subsequent generations, genes are subjected to modifications such as crossover and mutation. In the crossover process, positions in the matrix of two phenotypes are mixed, creating a new phenotype. This process is done randomly to avoid getting stuck in local optima and to ensure diverse solutions. Mutation involves randomly swapping subjects or sets of subjects with equal durations within the same gene, always ensuring that the resulting schedules remain valid. These processes repeat, moving through generations until either a satisfactory solution is found or the maximum number of generations is reached, concluding the program execution.

PickOne, the selection technique

Within the domain of GA, the "pickOne" procedure is crucial in guiding the evolution towards an optimal or nearly optimal solution. The primary aim of this strategy is to choose individuals from the existing population, favoring those with higher fitness scores, to take part in crossover and mutation processes, thus creating the next generation.

1. **Assignment of Fitness-Proportionate Probabilities**: In this initial stage, individual entities within the population are attributed specific probabilities of selection. These probabilities are meticulously calibrated in alignment with each individual's respective fitness scores, thereby ensuring that individuals demonstrating superior fitness characteristics are accorded heightened probabilities of selection.

2. **Implementation of Roulette Wheel Selection**: The "pickOne" process frequently employs the "roulette wheel selection" strategy, also recognized as "fitness proportionate selection." Within this framework, a hypothetical roulette wheel is conceptualized, wherein each individual occupies a distinct segment of the wheel. The dimension of each segment is precisely proportional to the individual's fitness score,

thus enhancing the likelihood of selection for individuals exhibiting superior fitness attributes.

3. **Application of Stochastic Universal Sampling**: An alternative methodology for implementing the "pickOne" process is through Stochastic Universal Sampling (SUS). This technique fosters a more equitable and comprehensive representation of the population by facilitating the simultaneous selection of multiple individuals through a singular spinning process, thereby preserving the population's intrinsic diversity.

4. **Manipulation of Selection Pressure**: The "pickOne" procedure can be adjusted to optimize the intensity of selection pressure applied. These adjustments provide precise control over the algorithm's convergence path, allowing for a balanced combination of fast convergence and the reduction of premature convergence dangers caused by decreased variety.

GPU Implementation: Parallel Processing and Adaptation

Implementing the GA on a GPU requires a thoughtful and meticulous adaptation of the algorithm's components to fully leverage the GPU's parallel processing capabilities. Chromosomes must be designed and represented in a manner handler to parallel processing, ensuring compatibility with the GPU's architectural nuances.

Fitness evaluations, a computationally intensive aspect of the algorithm, can be executed concurrently on multiple threads, allowing for the simultaneous evaluation of multiple individuals, thus enhancing computational efficiency. Similarly, selection, crossover, and mutation operations can be adapted for parallel execution, where multiple pairs of chromosomes undergo genetic operations concurrently, further leveraging the parallel processing capabilities of the GPU.

Synchronization strategies must be effectively implemented to ensure data consistency and coherence during parallel operations. Additionally, data transfer between the CPU and GPU must be optimized to prevent bottlenecks and ensure the algorithm operates efficiently.

Kernel functions, fundamental to the GPU's operation, must be meticulously designed to be optimized for the GPU architecture, ensuring effective utilization of the GPU's resources and memory hierarchies. This involves tailoring the kernel functions to align with the GPU's processing paradigms, ensuring that they are optimized for performance and efficiency.

Through a detailed and thoughtful implementation and optimization of the GA on the GPU, the GAS is enhanced, ensuring it operates with a high level of computational efficiency, scalability, and overall performance.

3.2.3 Module 3: Voting using a BlockChain

The voting system is projected to provide secure, reliable, and transparent voting. It is built on a blockchain platform, which ensures that every vote is saved in a tamper-proof manner. The voting system consists of several components:

1. **Voter Registration**: To participate in the voting process, users need to register and authenticate themselves using a secure and user-friendly interface. This component includes identity verification and digital signatures to ensure the authenticity of each voter.

2. **Block Creation**: The block creation component is responsible for creating the block and ensuring that it contains all the necessary information required for the voting process. This includes candidate lists, referendums, and voting rules.

3. **Voting**: The voting component is responsible for ensuring that each vote is recorded and stored in the blockchain securely and transparently. The component ensures that the vote is valid and that the voter has not voted twice.

4. **Vote Counting**: The vote counting component is responsible for tallying the votes and declaring the winner. This component ensures that the vote count is accurate, transparent, and tamper-proof.

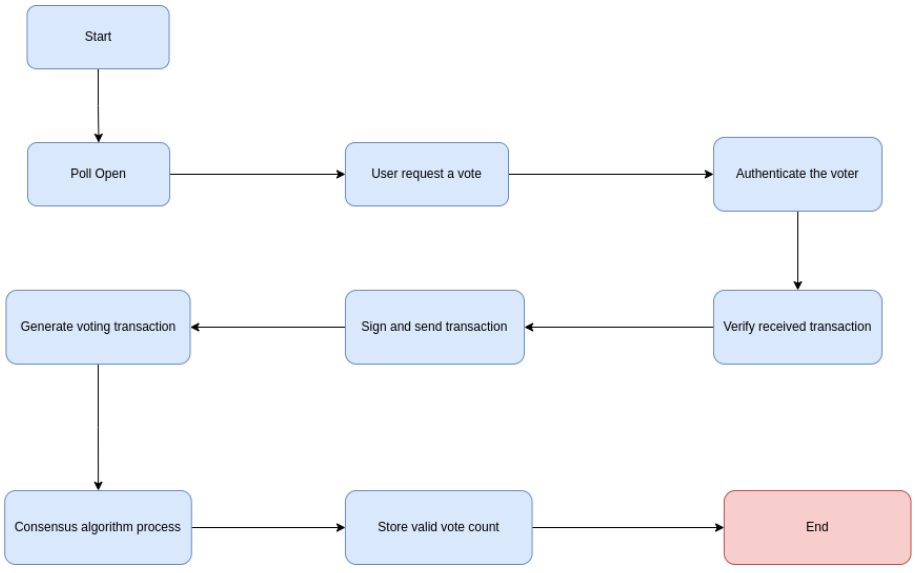


Figure 3.5: BlockChain operation flowchart

Figure 3.5 represents all the steps that a vote needs to pass before being inserted in the blockchain. The proposed system is a voting platform that utilizes blockchain technology for enhanced security and encryption. It enables organizations to create and customize their own system for conducting different polls. The primary objective of this system is

to establish a voting mechanism that is both transparent and resistant to tampering, hence eliminating the risk of hacking, while still ensuring the anonymity of voters.

The system makes use of a customized blockchain, which eliminates the requirement for institutions to pay feeds to publish on blockchains such as Ethereum or Bitcoin. The system may be built as Docker (Merkel, 2014) images that are deployed with Kubernetes (Dikaleh *et al.*, 2017) to produce numerous instances per server on as many servers as the institution requires, making it cost-effective. Kubernetes is critical to ensure the system is always highly available and responsive. Nonetheless, the system's deployment component may be tailored to the institution's present system. The system validates itself via proof of work, which is more expensive on the computer but a safer alternative for tiny blockchains.

The online interface for establishing and editing polls allows the pool's developer to assign a height to each person's vote, which is essential in administration board polls where each person's vote has a varied amount of importance. Its height can be described as the proportion of the institution that the individual owns, such as in an administrative pool. To ensure transparency, the system offers anonymous voting, where no one knows who owns the wallets, but everyone can see where each wallet voted. Additionally, the blockchain is encrypted to guarantee that data is safe and that only authorized parties have access to it.

The system assures that the voting process is visible, tamper-proof, and anonymous by utilizing blockchain technology, making it appropriate for a variety of applications where trust and transparency are crucial. The construction of a smart contract on a blockchain containing a list of voters and their votes, all encrypted using homomorphic encryption. Homomorphic encryption is a type of encryption that allows calculations to be conducted on the ciphertext. This leads to an encrypted result that may be decoded to get the same result (Ogburn *et al.*, 2013). Each voter would be given a one-of-a-kind private key that would be used to encrypt their vote. The encrypted votes would be transferred to a mixer, which would use the mixing process to shuffle the votes. Depending on the application, the mixer might be centralized or decentralized. The use of mixing increases privacy and helps to avoid vote rigging.

Figure 3.6 illustrates the sequential operations performed by the mixer to guarantee that the votes saved on the blockchain are not preserved in their original order. This flowchart depicts the process of a centralized mixer. In a decentralized mixer, the processes remain the same, but there are additional steps that distribute the votes equally across each node. Finally, there is a merging process at the end to verify that all nodes have the same block order. It is advisable to use a decentralized mixer when there are numerous votes to exit.

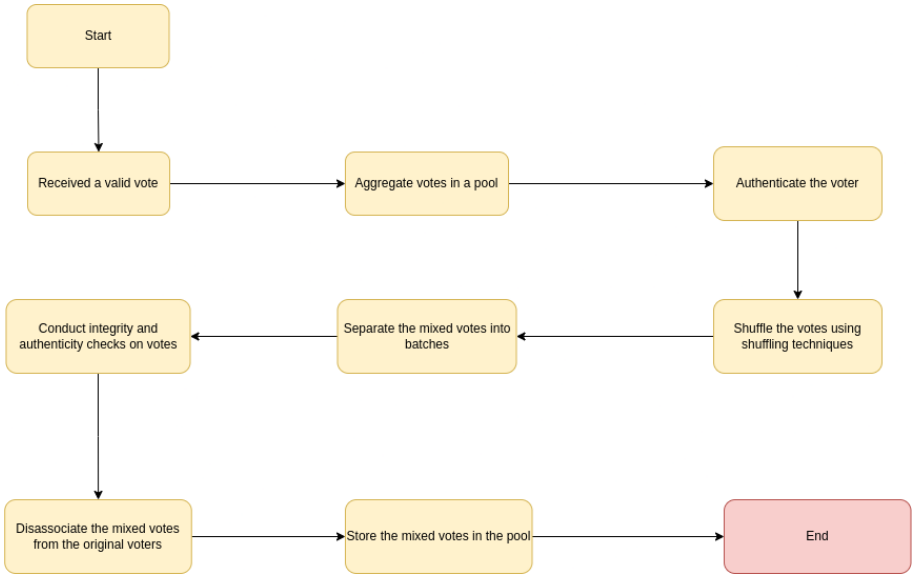


Figure 3.6: Mixer flowchart

The smart contract would be set up to employ Zero-Knowledge Proofs (ZKPs) to allow each voter to verify their vote without exposing their decision. A ZKP is a cryptographic procedure that enables one party to demonstrate knowledge of a secret without disclosing the secret itself (Zitnik, 2013). The Schnorr protocol, for example, is a ZKP that may be used to establish knowledge of a discrete logarithm without exposing it (Goldreich *et al.*, 1986).

The smart contract would also be set up to employ ring signatures, which are made by several voters but only one of them is the real signer. This makes determining which voter signed the message challenging, as all members of the group might have been responsible for the signature. Rivest, Shamir, and Tauman (Rivest *et al.*, 2001) pioneered the use of ring signatures in different privacy-preserving protocols, including anonymous electronic payment systems. Ring signatures are a form of digital signature system that allows a group of users to sign a message without exposing which member of the group signed it (Chen, 2012). Upon the completion of the voting, the smart contract with the list of encrypted votes will be used to count the votes using homomorphic encryption. This permits the votes to be tabulated without exposing each voter's preference. This technique allows each voter to know who they voted for but not where their fellow voters voted, preserving anonymity. It also enables votes to be tabulated without disclosing each voter's decision, respecting their privacy and security. The use of zero-knowledge proofs and ring signatures adds a degree of security and anonymity, making it even more difficult for third parties to learn the identities of voters or alter votes.

3.3 Technical In-depth

This chapter continues discussing the modules of the developed system, with a greater emphasis on the technical aspects. Delves into how the system was implemented, detailing the entire logic and information management that occurs while the system is running.

3.3.1 Genetic Algorithm

In this chapter, we will look into a more technical explanation of the four phases of the genetic algorithm. Here, it will thoroughly explain the logic behind the developed algorithm, covering each phase in detail.

Initialization phase

After all information and constraints have been initialized and allocated in the data structures mentioned in the previous section 3.2.2. This phase occurs at the gene level and is executed asynchronously for all genes, where each gene represents a unique schedule. In it, a matrix of atoms is allocated. Each atom represents a time slot containing information such as whether the slot is free or occupied, and if occupied, details like the room to be used and the specific class. An array of positions is also allocated, where each position represents a time slot with information like the day and hour. It's important that in this array to maintain only the time slots that are still free. Next, all the classes in the gene are sorted. The sorting criteria can vary based on restrictions, such as the type of class. An optimization is to use the duration of the class as a criterion as well, as this will provide more free slots for longer classes, which will avoid the need to reposition shorter classes later. A good practice is to validate minimum requirements at this stage, such as whether there are enough teachers for all classes and sufficient time slots for all classes.

With everything allocated, it will proceed to fill the previously allocated atom matrix. By Iterating through all the classes, and for each class, it's searched for a space. Then we validate if there are enough slots for the duration of the class, as some classes occupy more than one slot. If the slot is not validated, it searches for another position in our array. If none of the available slots in our array has enough space, we will need to reorder the classes already existing in the matrix, eliminating dead spaces, as indicated and explained in the previous section, always being careful to update the array of free spaces to avoid overlapping classes in future iterations. Having identified our time slot, it's need to find a room for the class. For the rooms, since the genes are being filled asynchronously, a map of free rooms and an array with all available rooms in the phenotype, representing the entire university schedule, were allocated. The map is used to control the overlap of classes in the same room, where the key is a combination of the day and hour, and the value is the class taking place in that slot. A map was chosen to maximize performance,

as this operation is a blocking operation for other genes since it is a critical space in our application because this operation can only be executed by one gene at a time. The search for our room is as follows, it's look for a room in the array of rooms and validate if it is available on the map. If available, the room can be used, remembering that when fetching the room from the array, it must choose a room that meets the class's restrictions.

Having obtained the room, the time slot, and the class, the only thing left is to place it in our matrix. For this, it needs to allocate an atom with the information of the room and the class and place it in the matrix in the intended time slot, considering that the matrix was allocated with the specification that the rows are days and the columns are hours.

Fitness phase

In this phase, the fitness calculation takes place. The fitness calculation represents the quality of the schedule; the higher the fitness score, the better the quality of the schedule in meeting specific requirements. For calculating this fitness, a subtraction technique was chosen. Each gene starts with a perfect score, meaning the maximum possible value for fitness. Then, for every failure in the schedule, points are deducted for each constraint that is not met or is incomplete. It is in this phase that the soft constraints are evaluated. The failure to meet soft constraints leads to the loss of points for the gene. Due to the validations in the previous phase, all hard constraints should be already met. Once the soft constraints are validated and the fitness value is calculated, the only step left is to update the phenotype by adding the fitness value to it. The phenotype will have a variable representing the fitness of the university schedule, which is the sum of the fitness of all genes. For this variable, it is good practice to use an atomic value, thus avoiding creating a critical zone in our code, remembering that all genes are calculated asynchronously.

Evaluate phase

The evaluation phase is the easiest to implement, given the way the other phases are implemented. This phase essentially involves finding the phenotypes with the highest fitness. In the initial configurations, two parameters are set that are used in this phase. One is the number of generated timetables desired, and the other is the minimum fitness required for a timetable to be considered valid. With these two parameters, iterate through the phenotypes and collect the top x with the best fitness, storing them in an array of phenotypes. This array is configured in the population, which is a level above the phenotype and represents all the university schedules generated for a given generation. However, these are only stored if the phenotype has a higher fitness than those already stored, this array is passed from generation to generation. If all these collected schedules have a fitness above the configured acceptable level, the program terminates and returns the collected phenotypes. If not, it checks whether the maximum number of generations, a parameter initially

set, has been reached. If it has, the program ends and returns the stored phenotypes; if not, it continues to the next phase.

Crossover and mutate phase

This phase is the transition phase, where one generation is dismantled to create another. It is divided into two stages crossover and mutation. The implementation of crossover is quite straightforward. Two phenotypes are chosen using the previously explained 'pick one' technique (section 3.2.2). A fusion is then performed between these two phenotypes. This fusion simply involves mixing the genes of both phenotypes. It is done by iterating through the genes and randomly selecting a gene from one of the phenotypes, thus creating a new generation of genes for a phenotype. It is important to note that the gene with the highest fitness is not chosen from the phenotypes to avoid a local optimum zone. Each pair of phenotypes creates one new phenotype, and a phenotype can be chosen more than once.

The mutation step occurs after crossover, once all the phenotypes have been generated. Mutation is a step carried out asynchronously for each gene, involving the reconstruction of its atom matrix, or its timetable. This involves iterating through the matrix and, for each atom, attempting to find another atom to swap with. The swap is executed by simply exchanging the atoms in the matrix. This search for a new atom is conducted with soft constraints in mind, aiming to increase fitness as much as possible.

The fitness, evaluate, crossover, and mutation phases are done in a loop for the number of times specified in one configuration variable. The initialization phase is the one that is made only in the creation of the first population.

3.3.2 Blockchain

The structure of a block consists of two main parts, the header and the body. The block header includes essential elements such as a timestamp that represents its creation time, and two cryptographic hashes, one referring to the previous block and another representing the hash of the current block. The block body predominantly contains the list of transactions, which in a voting context, would consist of information like voter identification and vote choice.

The process of creating a block starts with the accumulation of pending transactions. These transactions, after being verified for authenticity and compliance with established rules, are processed. The system then performs hashing operations, essential for ensuring the security and integrity of the blockchain. The hash of the previous block is included in the new block, maintaining the chain's continuity, and a new hash is generated for the current block from its data. This process creates an immutable and sequential record of

blocks, where altering one block would require changing all subsequent blocks, a task nearly impossible due to consensus mechanisms like Proof of Work used to validate and add blocks to the chain.

Regarding the storage of transactions within blocks, each transaction is represented by a data structure containing key information and is digitally signed. The immutability, a central attribute of the blockchain, is ensured by this architecture, where modifying any information in a block is computationally expensive. Furthermore, establishing a peer-to-peer network is crucial for the communication and synchronization of the blockchain across different nodes, ensuring its decentralization.

Chapter 4

Empirical Evaluation of the System

This chapter systematically unfolds the outcomes of the empirical experimentation. Initially, the chapter elaborates on the problem (section 4.1), where a meticulous portrayal of the comprehensive testing environment, alongside the operational conditions, is elucidated. This foundational section is imperative, furnishing the essential backdrop necessary for an informed interpretation and appreciation of the empirical findings subsequently presented. After the scenario description, the chapter evolves towards an in-depth exploration of the messaging flow (section 4.2), where it will describe all the flow of messages and data around the system. To finalize, a results discussion concerning the algorithm's operational performance is conducted, encapsulating an exhaustive presentation of crucial performance indices and characteristics pertinent to evaluating the algorithm's overarching efficacy and dependability (section 4.3).

4.1 Evaluation Setup and Proposed Scenario

In this part of the study, it was carefully set various testing scenarios to see how well our algorithm performs, utilizing a Lenovo Yoga 720, equipped with an Intel Core i7 processor, 16GB of RAM, and an NVIDIA GeForce GTX 1050 graphics card. Despite the computer managing other processes simultaneously, which may influence the outcomes, it hosted our experimental trials effectively.

The experiment hinges on authentic, historical data from a university, presented in a structured file format, acting as pivotal input for our algorithm. The data layout encapsulates diverse attributes like Discipline Order, Class, Course, Academic Year, Semester, Discipline Type, Duration, and Lecturer, amongst others. For instance, consider a line: "ESM2, 1, RDR, FDG, ENE, 3, 2, Enfermagem de Saude Mental e Psiquiatrica II - Ensino Clinico, pra, 2, Prof 1, Prof 2. This represents a precise class session, detailing that it's a practical class of 'Enfermagem de Saude Mental e Psiquiatrica II - Ensino Clinico' for third-year students, lasting two hours. This format, although rich in information, is

meticulously processed by our algorithm, ensuring each data point is adequately respected in the scheduling algorithm.

Table 4.1: Input data example.

Disc	Order	Acronym	Class	Course	Year	Semester(Per.)	Subject	Type	Duration	Teacher(s)
ESM2	1	RDR	FDG	ENE	3	2	Enfermagem de Saude Mental e Psiquiatrica II - Ensino Clinico	pra	2	Prof 1, Prof 2
OPC4	2	RDR	FDH	ENE	3	2	Economia Internacional (Opção 4)	teo	1	Prof 3, Prof 4
MUL1	1	JDS	FDI	ENE	3	2	Multimédia I	teo	2	Prof 5, Prof 2
EUE1	1	PPO	FDA	ENE	3	2	Ética e Deontologia Profissional	teo	1	Prof 3

Table 4.1 delineates a segment of our data, meticulously crafted to represent a scenario encompassing five days, each comprising ten available hours for scheduling classes. This carefully curated dataset embodies a robust compilation of approximately 3366 classes, distributed across 133 distinct rooms, complemented by the presence of 390 additional rooms, thereby offering a comprehensive landscape for our scheduling endeavors.

In the heart of our experimental scenarios, hard and soft constraints were diligently observed. Professors, subjects, and room specifications, amongst various considerations, navigated the algorithm's functioning, striving for a schedule that mirrors practical feasibility and optimal organization.

A vivid aspect of our scenario was the inclusion of a dynamic voting mechanism. Simulated students engaged in voting on different schedules, enabling us to gauge the robustness and responsiveness of our blockchain and voting system amidst high-frequency, simultaneous user interactions.

The spectrum of our scenarios, from stringent rule adherence to the vibrancy of user interactions, crafts a multifaceted testing environment. This design allows for a complete evaluation of what the algorithm can do, making it easier to make improvements for practical use and better performance.

Moving forward, the array of scenarios curated for this study has been meticulously crafted to evaluate the adaptability and proficiency of the algorithm in various contexts. Initial scenarios were manifested from a concoction of diversified yet hypothetical conditions to assess the algorithm's resilience and flexibility. The latter scenarios were meticulously molded closer to reality, ensuring a profound analysis grounded in practicality, where genuine data and constraints became the cornerstones of the testing environment.

In a crucial scenario, the algorithm was bestowed with a tapestry of real-world data, marinated with the intricate nuances and unpredictabilities inherent in a university's operational landscape. The data, a meticulous compilation of various academic elements such as courses, professors, and specific classroom constraints, became the crucible where the algorithm's mettle was tested. Hard constraints like professor availability, classroom specifications, and disciplinary restrictions, were non-negotiable, ensuring the algorithm's output echoed the realms of feasibility and orderliness.

The algorithm adhered to soft constraints, focusing on practical desirability. Objectives included minimizing idle times, optimizing room changes, and structuring the timetable for pedagogical coherence and student convenience.

Connected with these algorithmic processes was an important voting phase. In this part, a group of simulated students interacted with the system, giving their feedback by voting, helping to test the system's ability to handle many users at once. This step was very important because it showed how well the algorithm could work in a busy, changing environment, highlighting its strength and dependability.

In conclusion, each scenario, a unique theatre of constraints, interactions, and objectives, becomes a vital act in our algorithm's performance. They collectively weave the narrative of its adaptability, efficiency, and readiness to champion the complexities and demands of real-world academic scheduling. Through this orchestration of multifaceted scenarios, the study aims to unfurl the algorithm's true potential and the promising symphony of technological innovation and practical applicability it heralds.

4.2 Messaging Flow Process

In the scheduling and voting system, the messaging flow process is essential. It connects different parts and ensures that everything runs smoothly and effectively, guaranteeing that important information is communicated clearly and reliably throughout the system. It's important to note that, in the explanation of these flows, the example of a single class in one day instead of the complete sample. Has chosen to allow to focus on the flow of messages and set aside the complexity of data generation and volume, providing a clearer and more focused understanding of the process.

The process starts with the registration phase, where all the necessary constraints and requirements are collected and organized carefully in a JSON object (shown in Figure 4.1). Each piece of information, such as requirements, availability, and priorities, is carefully considered and placed, creating a detailed and well-organized overview that guides the scheduling within the academic environment.

As the scheduling process begins, the genetic algorithm plays a crucial role, representing intelligence and flexibility. It creates various possible schedules, each one aiming for the best organization, compatibility with restrictions, and overall practicality (an example of this is displayed in Table 4.2).

In the arena of decision-making, the voting system unveils itself, embracing blockchain's transparency and security. Wallets, unique identifiers of participants, open the gates to a realm where votes cascade into the system, each finding its immutable place within the blockchain (illustrated vividly in Figure 4.2).

```
{
  "Teachers": [
    {
      "name": "Prof 1",
      "subject": "Enfermagem de Saude Mental e Psiquiatrica II - Ensino Clinico",
      "hour_per_day": 10
    },
    {
      "name": "Prof 3",
      "subject": "Economia Internacional (Opção 4)",
      "hour_per_day": 12
    }
  ],
  "Schedule": [
    {
      "Disc": "ESM2",
      "Order": "1",
      "Acronym": "RDR",
      "Class": "FDG",
      "Course": "ENE",
      "Year": "3",
      "Semester": "2",
      "Subject": "Enfermagem de Saude Mental e Psiquiatrica II - Ensino Clinico",
      "Type": "pra",
      "Duration": "2",
      "Teacher": [
        "Prof 1",
        "Prof 2"
      ],
      "Capacity": 15,
      "Projector": false,
      "Sockets": true
    }
  ]
}
```


Figure 4.1: Teacher, Schedule Constraints

Table 4.2: Example of a schedule generated by the system

Timetable	Room 101	Room 102
Monday 9h	None	MUL1
Monday 10h	OPC4	MUL1
Monday 11h	EUE1	None
Monday 12h	None	None
Monday 13h	None	ESM2
Monday 14h	None	ESM2

```
List Address:
1DWskLPPtZSkwmzLUsMNpvU2UWpNVeiSG
1H56zRgCAE9L7azu3EHtFAJxHzzFQ1vknR
1CTQY3TokZdYm6epD23tahETeFRPVRepoS
1FUzHVHxFpQfXx9nXRjvPwCJmgtQfUQMx7
```

Figure 4.2: List of the public addresses of the wallet of the voters

Transactions and blocks are created using blockchain technology, acting like strong containers for participants' choices. Each block is a secure and unchangeable record, ensuring that every decision is protected from changes and dishonesty, as illustrated in Figure 4.3.

```
Previous Hash: 00000157a2b5c04a3f4eed716d5c8eba68ee345a007b0c2343441112315767bb
Hash: 00013412232d1018be11689cbfbc7dbcddd8ed2191278dc36464fa26eaeb9e01
Pow: true

Previous Hash: 00009c94a5df539809bbfa706ee9c7a8f40f1f094c43745290f2748feaa7a027
Hash: 00000157a2b5c04a3f4eed716d5c8eba68ee345a007b0c2343441112315767bb
Pow: true

Previous Hash: 0000376aa3195fe59ced8f93250ffb036158cb1ec37afd95ec96755413db25bd
Hash: 00009c94a5df539809bbfa706ee9c7a8f40f1f094c43745290f2748feaa7a027
Pow: true

Previous Hash: 00014c3e43bdb7bd0bb26112861a31ce74e6a1badfbc8cc3006a73e6c097e5de
Hash: 0000376aa3195fe59ced8f93250ffb036158cb1ec37afd95ec96755413db25bd
Pow: true

Previous Hash: 000298a5a4100d5e706687f90a84575268b6d05d58cef0732a4fa43ef15c4868
Hash: 00014c3e43bdb7bd0bb26112861a31ce74e6a1badfbc8cc3006a73e6c097e5de
Pow: true
```

Figure 4.3: Blockchain Transactions

The results are finally revealed, representing the combined choices and wishes of all participants. This concludes a process of active and flexible decision-making (the final results are shown in Figure 4.4).

```
Parsed data:
Total votes for the "Schedule 1": 1
Total votes for the "Schedule 2": 3
```

Figure 4.4: Result of the Pool

The system works like a well-designed painting where rules, calculations, choices, and outcomes come together to tell a strong story of smart technology and creative planning.

It is where the practical needs of school operations are met using technology and logical design, and where people's decisions are included in making choices. This creates a system that is both logical and creative, meeting the real-world needs of education.

4.3 Results Discussion

This chapter outlined all the tests conducted on the developed system. It's divided into two sections for clearer and more specificity. These two parts correspond to the main components of the system, the genetic algorithm and the blockchain voting system. Quality and performance tests were carried out for both parts. Real data from a university was used to generate schedules and compare them with the schedules used by the university in one year.

4.3.1 Schedule Analysis

This chapter, delves into the heart of our analysis, exploring the wide range of results garnered from extensive testing of the scheduling system. The significance of a timetable generation system isn't merely in its ability to produce outcomes, but also in the quality, practicality, and real-world applicability of these results. With this outlook, it had been conducted thorough evaluations.

Will begin with an in-depth comparison of the generated schedules, assessing their structure, cohesion, and convenience. These tests will allow us to discern how well these timetables align with students' genuine needs and the dynamics of educational institutions.

Beyond the qualitative analysis of the schedules, this chapter will also realize performance tests. These tests are pivotal in understanding the system's efficiency, response time, and capability to handle varying data sizes and complexities.

This chapter will explore other pertinent tests that supplement our grasp of the system, underscoring its strengths and areas for improvement. This holistic approach aims to provide a clear and comprehensive insight into the system's efficacy and efficiency.

Figure 4.5 unveils a meticulously generated schedule, made by our algorithm for the CRE class, which is a class of criminology, embarking on its first academic year at the university. At first glance, one can appreciate the strategic balance between academic rigor and student well-being, which our system prioritizes.

First and foremost, the importance of a well-deserved break is undeniably recognized by our algorithm. It consistently earmarks the 13:00 to 14:00 timeslot across every weekday for lunch. Such continuity not only caters to students' nutritional needs but also ensures mental rejuvenation, preparing them for the academic challenges that follow.

GENE CRE1					
#	Monday	Tuesday	Wednesday	Thursday	Friday
09:00		HPCR TP 56	EDPR TP 55	MCSO TP 84	
10:00		HPCR TP 56	EDPR TP 55	MCSO TP 84	HPCR PL 29
11:00				ING1 TP 84	PSCR TP 14
12:00				ING1 TP 84	PSCR TP 14
13:00					
14:00		DPEN TP 24		GRC TP 50	
15:00		DPEN TP 24		GRC TP 50	
16:00		DPEN PL 24		PSSO TP 50	
17:00		CPUE TP 24	PBPC TP 45	PSSO TP 50	
18:00		CPUE TP 24	HPCR PL 18		
19:00		CPUE PL 24			

Figure 4.5: Generated Schedule for CRE1

Moving on to the classroom arrangements, our system demonstrates a penchant for stability. It endeavors to minimize the often disruptive classroom transitions. This is evident in the way the schedule mostly keeps students in a singular classroom for extended class sequences. There are, however, a couple of deviations from this pattern, notably on Wednesday afternoons and Friday mornings. Yet, these could be attributed to specific classroom availability or resource constraints, which the system has intelligently navigated.

Furthermore, the schedule embodies a pedagogical philosophy: laying the theoretical groundwork before delving into practical applications. This is perceptible in the deliberate sequencing of theoretical sessions preceding their practical counterparts, aligning with the age-old educational principle of "understanding before application."

The schedule has been rated at 81 out of 100, indicating its efficiency. However, there's an area for improvement. It was been observed that the HPCR PL sessions are dispersed across different days, which may pose continuity challenges for students. Addressing this could enhance the schedule's coherence. Additionally, considering a shift of Wednesday afternoon classes to the morning might offer students an additional free period in the afternoon, beneficial for individual or collaborative study. Furthermore, streamlining room assignments to limit transitions could enhance the daily academic experience.

In sum, the presented schedule stands as a testament to our algorithm's prowess, offering a blend of academic structure and student-centric flexibility. While it showcases commendable efficiency, the quest for perfection continues, inviting further refinements to elevate the student experience.

GENE CRE1					
#	Monday	Tuesday	Wednesday	Thursday	Friday
09:00	PSSO TP 56	EDPR TP 20			
10:00	PSSO TP 56	EDPR TP 20		PBPC TP 54	DPEN PL 102
11:00	DPEN TP 13	ING1 TP 20		GRC TP 54	HPCR PL 102
12:00	DPEN TP 13	ING1 TP 20		GRC TP 54	HPCR PL 102
13:00					
14:00				HPCR TP 68	CPUE TP 102
15:00				HPCR TP 68	CPUE TP 102
16:00				MCSO TP 68	PSCR TP 102
17:00				MCSO TP 68	PSCR TP 102
18:00					
19:00					

Figure 4.6: Second generation for CRE1

Upon a closer look at this second iteration of the schedule (Figure 4.6), we see a stark contrast to its predecessor, emphasizing the strides taken in refining the algorithm. With a performance score now soaring to a commendable 91, this rendition showcases its capacity to adeptly address many of the initial design's pitfalls.

A significant triumph here is the eradication of the issue of splitting practical classes a notable grievance in the prior version. Furthermore, the utilization of available hours has been optimized, reducing unproductive gaps that students might have found inconvenient. While the initial schedule had evident areas where simple room switches would significantly benefit its practicality, this revised version has minimized such obvious shortcomings.

Yet, as with all designs, room for refinement always exists. A recurring theme from the earlier schedule, the frequent classroom transitions, remains a challenge. The model set on Friday, where students occupy the same classroom for the entirety of the day, stands out as the gold standard. Replicating this streamlined approach across all weekdays would

elevate the schedule's convenience and functionality even further.

The enhancements observed in this iteration can be traced back to meticulous adjustments made to the algorithm. By opting for a larger population and expanding the number of generations, the system was granted a broader exploratory scope. This allowed for better avoidance of local optima, yielding schedules of heightened fitness. However, it's crucial to note that such augmentations come with their own set of demands. Increasing the population and generations not only requires more intricate tuning but also demands a surge in computational resources, necessitating a delicate balance between optimization and resource allocation.

4.3.2 Real Schedule Analysis

This section compares the university's actual timetables with those produced by the system. To analyze how well the system works and find areas that need improvement. This analysis will test the system's adaptability to real-world schedule variations and needs.

Table 4.3: Schedule of the class AQT

Disc	Order	Acronym	Class	Course	Year	Semestre(Per.)	Subject	Type	Duration
DES1	145	PIP	DGP	AQT	1	1	Desenho I	PL	"seg (15:00 - 17:00) 101 (2014-09-22 - 2015-01-16) qua (15:00 - 17:00) 101 (2014-09-22 - 2015-01-16)"
DES1	145	PIP	DOZ	AQT	1	1	Desenho I	TP	"seg (14:00 - 15:00) 101 (2014-09-22 - 2015-01-16)"
GETO	145	FRF	ENU	AQT	1	1	Geometria e Topologia	TP	"seg (13:00 - 14:00) 101 (2014-09-22 - 2015-01-16) qui (14:00 - 16:00) 101 (2014-09-22 - 2015-01-16)"
HART	145	NHP	DGR	AQT	1	1	História da Arte	TP	"qua (13:00 - 15:00) 202 (2014-09-22 - 2015-01-16) qui (13:00 - 14:00) 202 (2014-09-22 - 2015-01-16)"
MTC	145	PAR	DGS	AQT	1	1	Métodos e Técnicas de Comunicação	PL	"ter (15:30 - 17:00) 105 (2014-09-22 - 2015-01-16)"
MTC	145	RAE	DQI	AQT	1	1	Métodos e Técnicas de Comunicação	TP	"ter (13:00 - 15:30) 0.1 (2014-09-22 - 2015-01-16)"
PJ01	145	ATA	DGT	AQT	1	1	Projecto I	OT	
PJ01	145	ATA	DQE	AQT	1	1	Projecto I	TC	
PJ01	145	ATA	DGV	AQT	1	1	Projecto I	PL	"ter (09:00 - 12:00) 101 (2014-09-22 - 2015-01-16) qui (09:00 - 12:00) 101 (2014-09-22 - 2015-01-16)"
PJ01	145	ATA	DQG	AQT	1	1	Projecto I	T	"seg (10:00 - 12:00) 101 (2014-09-22 - 2015-01-16)"

Table 4.3 displays the schedule for class AQT, a first-year, first-semester class. This was the actual schedule of that year without any changes or modifications. Figure 4.7 shows the same schedule in our system's format for easier viewing and analysis.

This schedule (Figure 4.7) was selected due to its unique characteristics. Subjects like PJ01 of type OT and TC do not have specific assigned times. They are project submissions and work review classes, so they do not have weekly classes. Another peculiarity is the presence of classes with varying durations. For instance, MTC has both practical and theoretical sessions that differ in length.

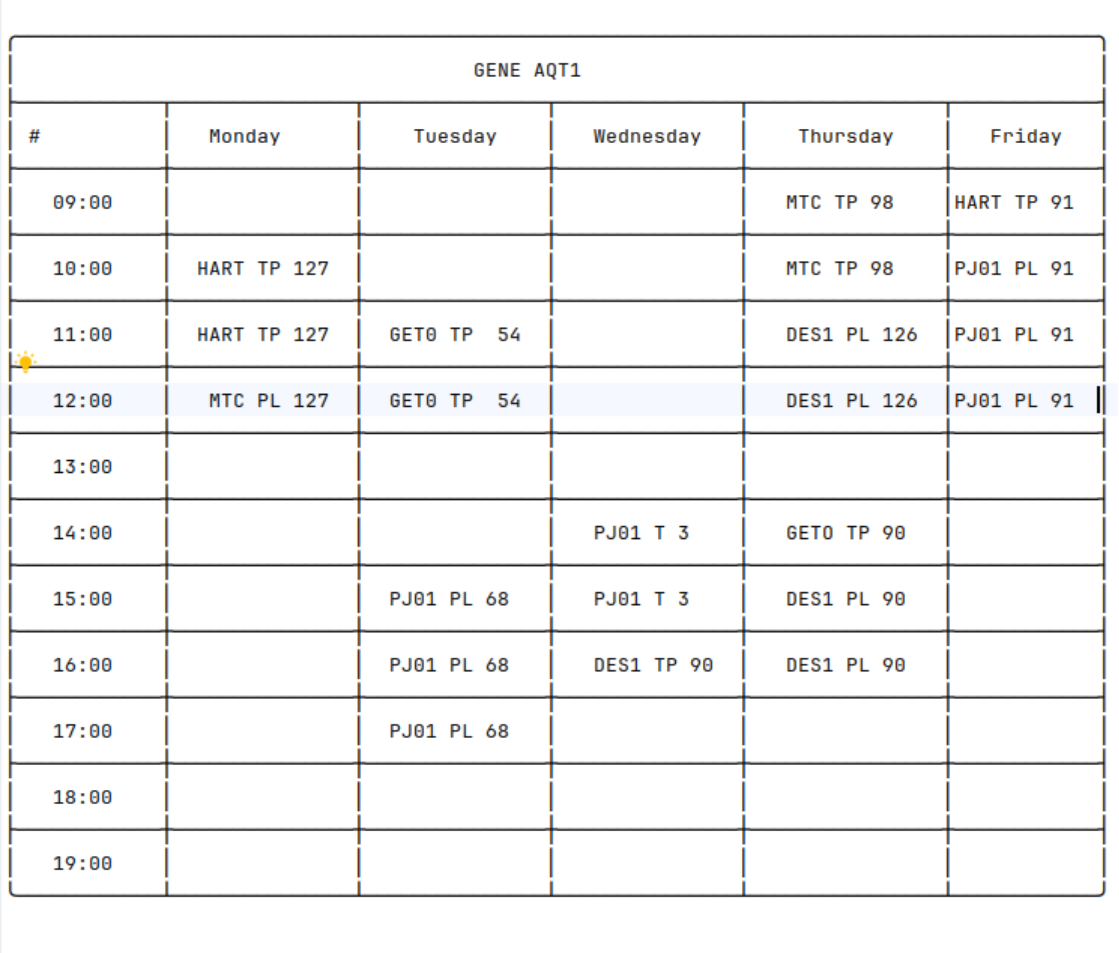
Moreover, the schedule is well-structured and comprehensive. The way it's organized grants students free periods on Friday and Wednesday mornings. Lunch is always at noon. Classes are primarily condensed between 9 a.m. and 5 p.m., with students finishing at 5 p.m. on three days and at 4 p.m. on another. This means that on class days, students have a packed schedule, which might be seen as a downside because the consecutive hours of instruction can be tiring for them.

GENE AQT1					
#	Monday	Tuesday	Wednesday	Thursday	Friday
09:00		PJ01 PL 101		PJ01 PL 101	
10:00	PJ01 T 101	PJ01 PL 101		PJ01 PL 101	
11:00	PJ01 T 101	PJ01 PL 101		PJ01 PL 101	
12:00					
13:00	GET0 TP 101	MTC TP 0.1	HART TP 202	HART TP 202	
14:00	DES1 TP 101	MTC TP 0.1	HART TP 202	GET0 TP 101	
15:00	DES1 PL 101	MTC TP 0.1	DES1 PL 101	GET0 TP 101	
15:30	DES1 PL 101	MTC PL 105	DES1 PL 101	GET0 TP 101	
16:00	DES1 PL 101	MTC PL 105	DES1 PL 101		
17:00					
18:00					
19:00					

Figure 4.7: Schedule AQT system view

The schedule from Figure 4.8, while distinct from the actual one, is also well-structured for students. This schedule runs from 9 a.m. to 5 p.m. without any full days off. However, this design offers two clear afternoons on Monday and Friday and a free morning on Wednesday. Although this might initially seem less ideal, this approach helps students by balancing classes and avoiding overwhelming schedules. This schedule was able to have all the theoretical classes before the practical ones, which can enhance learning. Consistent lunch breaks were maintained, and there was a noticeable effort to minimize room changes. For the majority of extended class sessions, students stay in the same room, with Thursdays being the exception.

Upon comparing the two schedules, both are well-structured and organized. Each maintains a consistent lunch break throughout the week, with the real schedule in Figure 4.7 setting it at noon and the generated one in Figure 4.8 at 1 p.m. Predominantly, both schedules have positioned theoretical classes before practical sessions. Efforts to minimize room changes are evident in both, with the generated schedule excelling in this aspect. It only necessitates a room change exception on Thursdays, whereas the actual schedule has changed on Tuesday, Wednesday, and Thursday afternoons. A limitation is the system's inability to handle variable time slots.



GENE AQT1

#	Monday	Tuesday	Wednesday	Thursday	Friday
09:00				MTC TP 98	HART TP 91
10:00	HART TP 127			MTC TP 98	PJ01 PL 91
11:00	HART TP 127	GET0 TP 54		DES1 PL 126	PJ01 PL 91
12:00	MTC PL 127	GET0 TP 54		DES1 PL 126	PJ01 PL 91
13:00					
14:00			PJ01 T 3	GET0 TP 90	
15:00		PJ01 PL 68	PJ01 T 3	DES1 PL 90	
16:00		PJ01 PL 68	DES1 TP 90	DES1 PL 90	
17:00		PJ01 PL 68			
18:00					
19:00					

Figure 4.8: Schedule AQT Generated

In conclusion, the schedule in Figure 4.7 appears more compact, affording students an entire day off. Conversely, the generated schedule in Figure 4.8 offers a less dense layout with a more even distribution of consecutive class periods. Based on this analysis, it can be inferred that the system is successfully capable of producing schedules suitable for real-world applications.

4.3.3 Genetic Algorithm Performance

This section thoroughly analyzes the performance indicators related to the genetic algorithm. The real-world relevance and usability of every algorithmic approach depend on its performance capabilities, and genetic algorithms are no different. A complicated set of tests, designed to test and clarify the algorithm's intricacy, seeks to offer a thorough assessment of its capacity to respond, scale, and remain strong. It is important to mention that these assessments were carried out on a personal computer, which may not accurately represent the algorithm's complete capabilities. An ideal setting for conducting such testing would unquestionably involve a specialized server, designed to minimize any fluctuations in the system. However, the goal is for this thorough investigation to

not only enhance the scholarly comprehension of the genetic algorithm's capabilities and constraints, but also to establish a path for its improved application in future initiatives, particularly in the domain of intricate optimization difficulties.

Generation evolution

Figure 4.9 illustrates the relationship between the number of generations and the time required for the algorithm to process each stage. The progression of the algorithm is evident through the ascending curve on the graph. This curve demonstrates that as the number of generations increases, the time required to process them correspondingly grows, though not in a strictly linear manner.

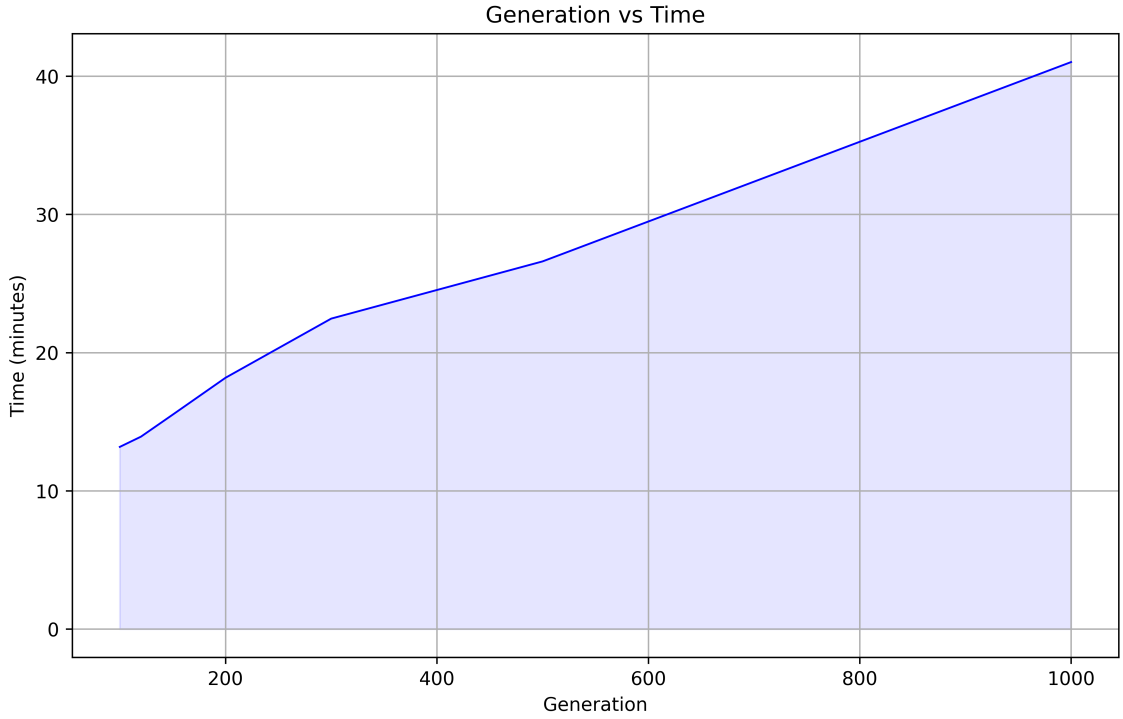


Figure 4.9: Generation Evolution

In the initial stages, is observed a more moderate growth pace in terms of processing time. The first few generations, up to about 200, show a gentler rise in time, suggesting that the algorithm was efficiently adapting and learning. This period can be perceived as the algorithm's "learning phase," where it was beginning to grasp and optimize its functionality.

However, as it moves towards higher generations, from 300 up to 1000, the curve's slope becomes steeper. This indicates a more pronounced increase in the time needed to process each generation. It can be inferred that as the algorithm evolves, the complexity of its solutions and adaptations also increases, leading to greater computational effort and, consequently, extended processing time.

The constant maximum population set at 100 suggests that the variation in time is primarily due to the increasing generations rather than population size.

In summary, the graph offers valuable insights into the algorithm's evolution and behavior. It reveals that as the algorithm advances and seeks more optimized solutions, it also demands more resources and time. Analyzing these trends is crucial for future optimizations and understanding the balance between efficiency and accuracy.

Population evolution

The progression of the algorithm, as illustrated in Figure 4.10, clearly demonstrates a direct correlation between the size of the population and the computational time required to process it. Specifically, as the population increases, the time required for its processing follows, where in this scenario the number of generations is the same,100 generations.

The graph's trend suggests an almost linear relationship between population and time. This can be interpreted as an indication that the algorithm is optimized to handle incremental population sizes; however, the temporal cost of this expansion is inherent to the computational process. Each incremental addition to the population appears to demand a proportionally incremental amount of time.

In comparison with Figure 4.9, there are analogous tendencies. In both scenarios, an increase in the independent variable, be it generation or population, has a directly proportional increase in execution time. However, it's paramount to note that the current focus is on population dynamics, rather than generational progression.

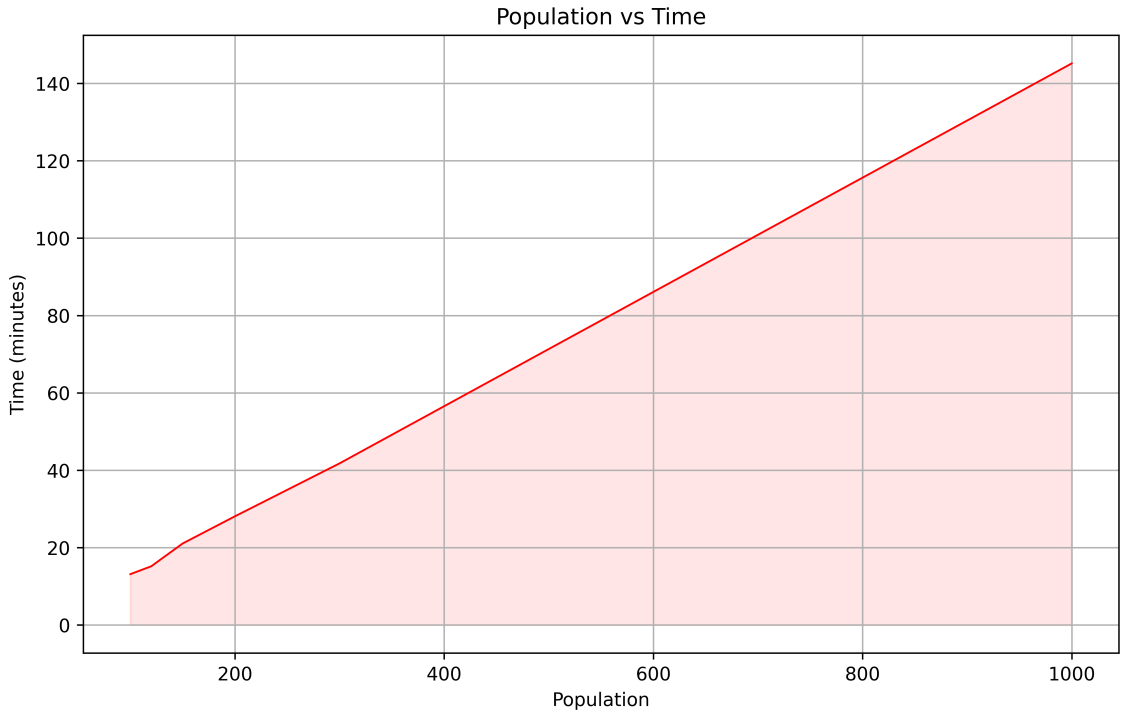


Figure 4.10: Population Evolution

In summary, when deciding to increase the test population, it needs to be aware of the extra time it will take. Understanding this is key to making the algorithm better in the future. This way can find the best balance between how fast it works and how well it performs.

Fitness evolution

The algorithm's fitness is a metric employed to measure the schedule's quality. The higher the number of constraints it respects and implements, the greater its fitness, indicating that the schedule's quality is directly proportional to its fitness.

In Figure 4.11, the evolution of the algorithm's fitness is displayed. In general, it is yielding the expected results, as the fitness increases with progressing generations. This implies that the quality of generated schedules is also on the rise, affirming the algorithm's ability to adapt to imposed constraints.

However, upon closer examination, it becomes evident that this evolution is not flawless. A perfect evolution would entail continuous progress from one generation to the next, which is not the case. Several stagnation zones can be identified, occurring when the number of generations increases but the fitness remains constant. These areas are called local optima (Barbulescu *et al.*, 2000) and can occur in two scenarios. Firstly, if no schedule with higher fitness than the best schedule from the previous generation is generated in the current generation, the current generation is disregarded, and the schedule from the previous generation prevails. For this reason, the fitness never decreases from generation to generation. Secondly, if the schedule generated in the current generation has the same fitness, it replaces the older schedule to increase the chances of the next generation avoiding a local optima.

In summary, the algorithm is exhibiting the expected behavior by evolving, but it frequently gets stuck in areas of local optima. An improvement to be considered involves implementing strategies to avoid these areas.

Population vs Fitness evolution

Now, the analysis will focus on determining the optimal factor for achieving the best results in a scenario with limited resources. The analyses will demonstrate the impacts of modifying the population size and the number of generations on the final fitness.

In Figure 4.12, one can observe the impact that the number of generations has on fitness. For this scenario, a constant population was used. As expected, there is a natural trend of increasing fitness with the rise in generations. Additionally, a gradual evolution of fitness is evident, where from the 100th to the 150th generation, there is only a two-point increase in fitness. Notably, the fitness of the 100th generation was 84, while at the 150th generation, it reached 86.

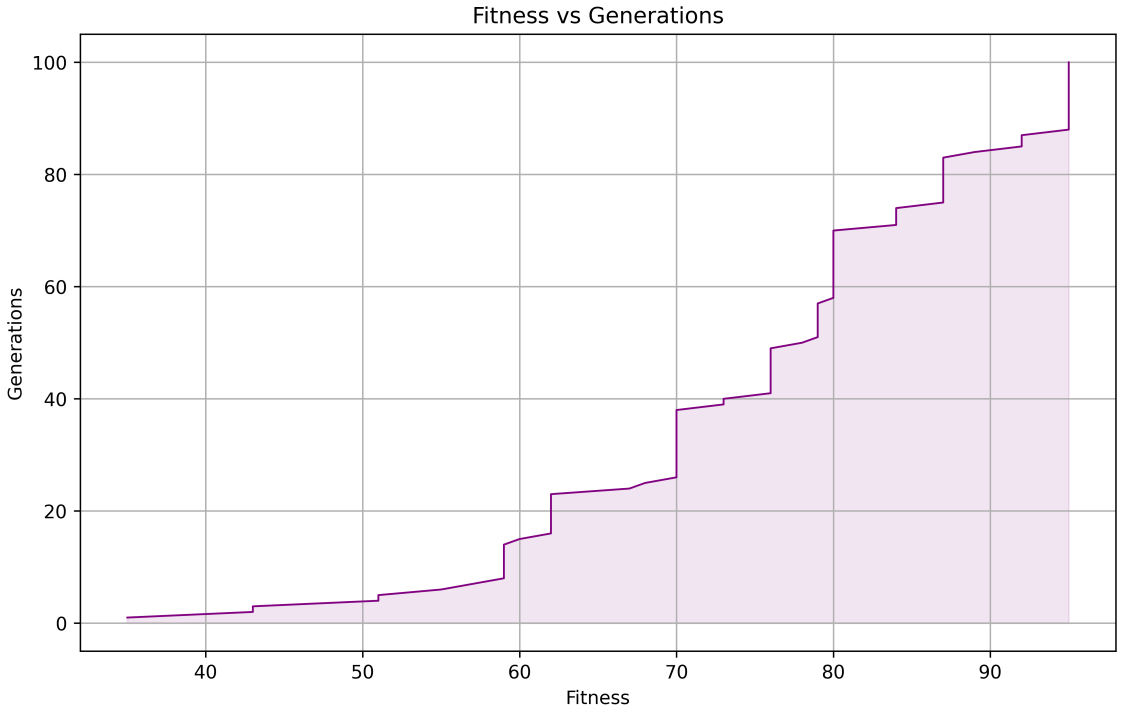


Figure 4.11: Fitness Evolution

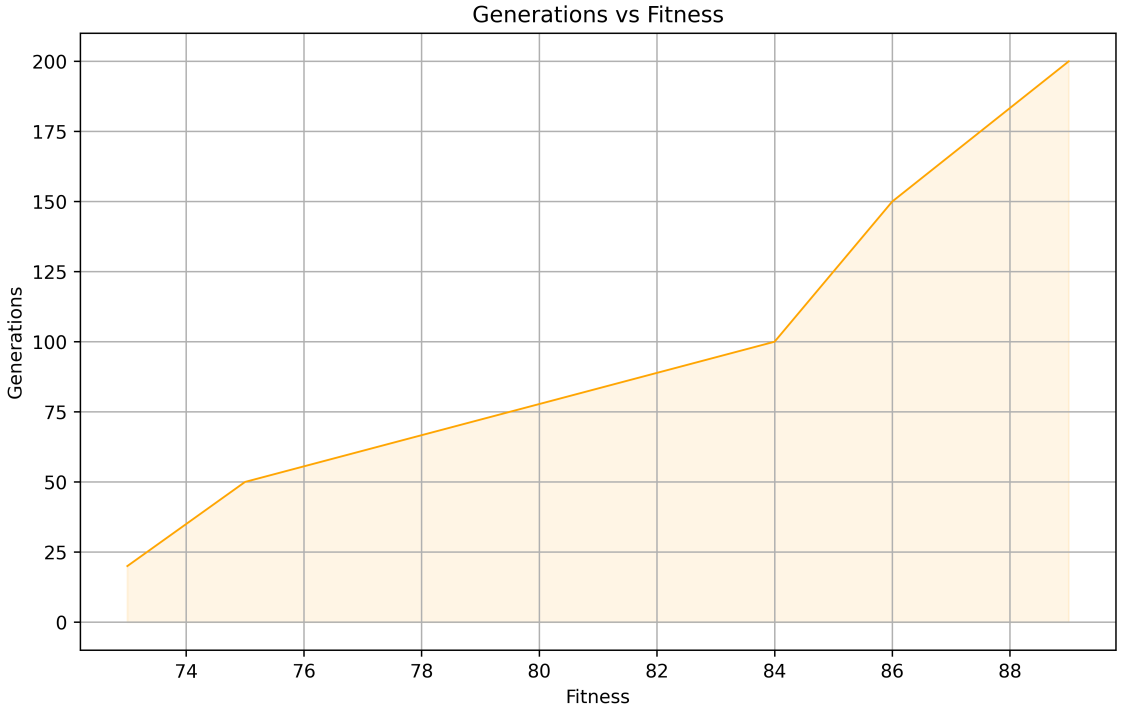


Figure 4.12: Generations vs Fitness

In Figure 4.13, one can observe the impact that the variation in population has on the final fitness. As expected, a notable occurrence is that with the increase in population, there is an improvement in the final fitness.

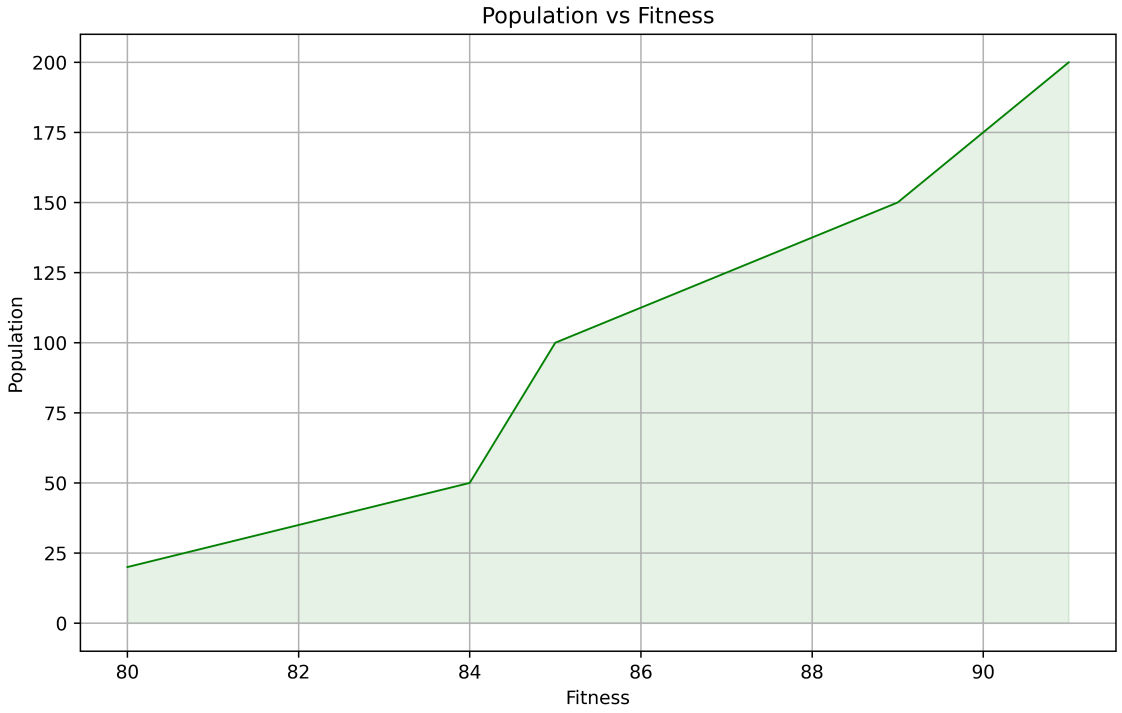


Figure 4.13: Population vs Fitness

Comparing the two figures above, it is possible to analyze that the increase in population exhibits a faster and more significant growth than the rise in generations. This occurrence is because, with the population increase, the algorithm finds it easier to escape optimal zones, reducing stagnation and promoting evolution. However, it's crucial to recall from previous tests (section 4.3.3) that this substantial population increase may not always be preferable. This is because, along with the significant rise in population, there comes a temporal increase, meaning the algorithm becomes slower.

Thus, it can be concluded that the modification of the population may be the most impactful factor on the final fitness. It achieves better fitness in the initial generations and a more effective evolution from generation to generation. However, this improvement comes at the cost of increased execution time. In some cases, it might be more advantageous to modify the number of generations if a solution is needed as quickly as possible, since both approaches tend towards positive evolution, and the execution time does not increase as exponentially as it does with population growth.

4.3.4 BlockChain Performance

In this subsection, the performance of the blockchain will be tested by altering its parameters to gain insights into how to optimize it for the hardware on which it will run. It is worth noting that the blockchain will operate on a personal computer with the previously mentioned specifications in section 4.1.

Difficulty variability

One of the initial parameters to be modified and tested was the mining difficulty of the blockchain. The "difficulty" in the proof-of-work refers to the complexity of the mathematical problem that the miner needs to solve to add a new block to the blockchain. The objective is to maintain a relatively constant rate of creating new blocks, regardless of the increase or decrease in processing power in the network (Fullmer & Morse, 2018).

Table 4.4: Difficulty vs Time

Difficulty	Time in minutes
10	0.0007182395
20	2.38
30	165

As evident from the Table 4.4, this parameter can drastically alter the performance of the blockchain. For a difficulty of 10, the equipment managed to create a block in less than a second, but for a difficulty of 30, it took 165 minutes for the equipment to create a single block.

Latency

For the latency test, the following scenario was set up: the difficulty remained constant throughout the process, but writes to the blockchain were performed with clients simultaneously. The number of clients making requests at the same time varied.

The first test was conducted with just one user, and the write took approximately 1.47 seconds to be validated and recorded. Now, in a scenario with 5 users, the write took an average of about 2.79 seconds, indicating that this slight increase in users almost doubled the time for the write to be registered. If, instead of 5, 20 users were performing a simultaneous write, the time increased to 11.70 seconds. With 100 users, each write took an average of nearly 1 minute to occur (58.11 seconds).

Thus, with these data and the Figure 4.14, it can be concluded that there is a direct correlation between the number of users registering simultaneously and the registration time for each vote on the blockchain.

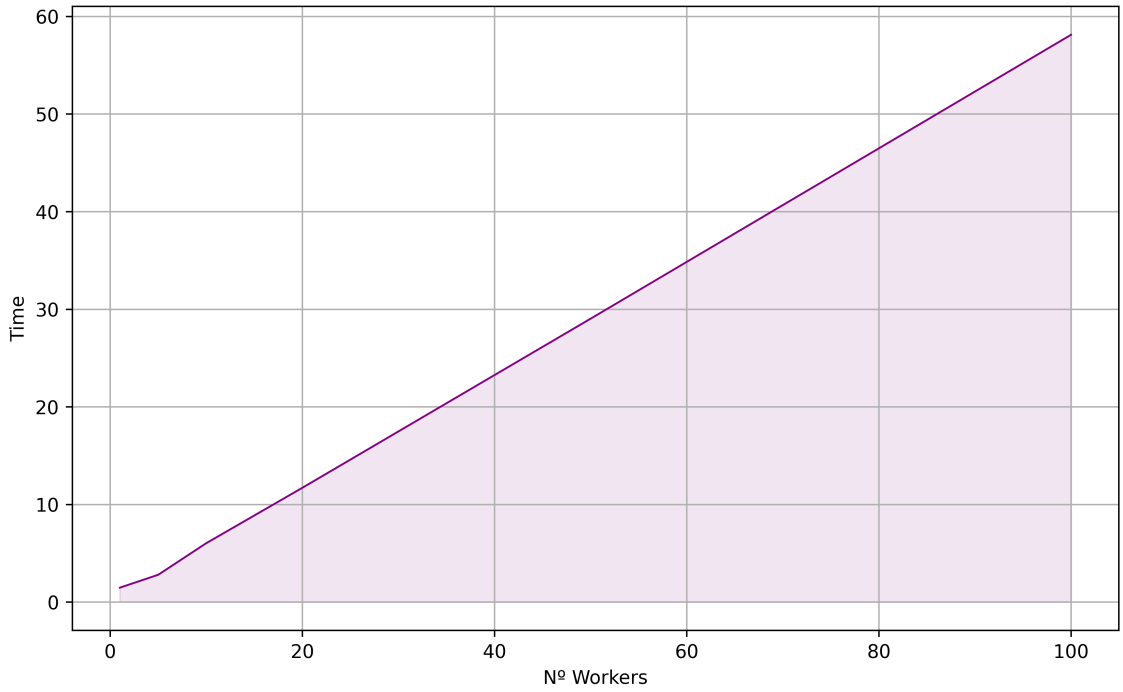


Figure 4.14: Latency

Difficulty and Latency

For a final test on blockchain performance, a combination of the two factors discussed earlier was examined to determine the true impact of a real-world scenario on write speed. The test was conducted by increasing the number of users registering simultaneously while also raising the blockchain difficulty level.

Table 4.5: Difficulty and Latency variation

	Number of Users					
Difficulty	1	5	10	20	50	100
10	15.49 ms	20.26 ms	28.86 ms	48.08 ms	167.74 ms	341.37 ms
15	662.32 ms	1.20 s	1.99 s	3.98 s	10.97 s	24.79 s
19	1.47 s	2.79 s	6.03 s	11.70 s	29.03 s	58.12 s
20	12.72 s	24.85 s	48.05 s	1 min 36.02 s	4 min 34.42 s	8 min 43.80 s

The results are presented in the Table 4.5, with users ranging from 1 to 100 and difficulty varying between 10 and 20. The difficulty remained relatively low due to computational power limitations.

Through this test, a confirmation of findings from previous tests emerged. Latency and difficulty directly influence blockchain to write time. With a low number of users, the blockchain is relatively fast for all tested difficulties. However, as the user count increases, the speed can reach almost 9 minutes for a difficulty of 20.

One conclusion drawn is that, for this computational power, the most suitable difficulty is 19. It proves to be relatively fast, maintaining a write time of about 1 minute with 100 users. This is beneficial for maintaining control over registrations without causing excessive delays. Any difficulty beyond that becomes too costly for the equipment in question.

4.3.5 Comparing Performance GPU vs CPU

Now will delve into a comprehensive analysis of the performance metrics of genetic algorithms on both CPU and GPU platforms. Using illustrative graphs as reference points aims to shed light on the stark differences and nuances between the two computing environments. This comparison will provide invaluable insights into where each platform excels and where potential bottlenecks may arise, serving as a guide for those looking to optimize their algorithmic executions.

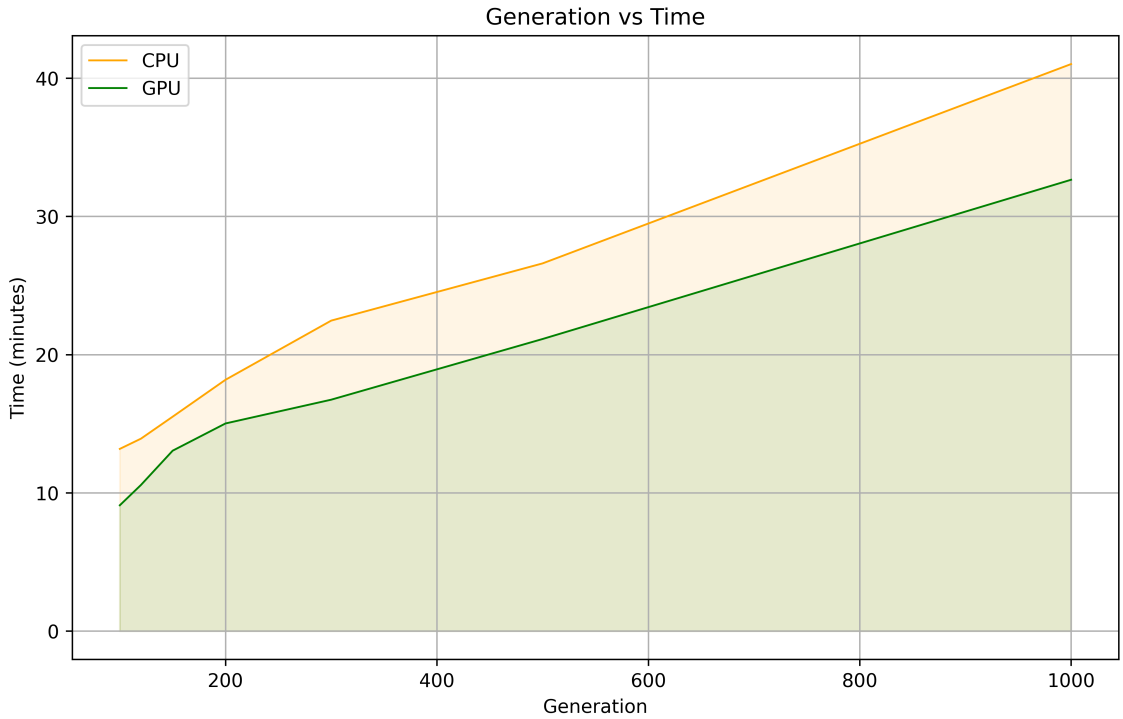


Figure 4.15: Generation CPU vs GPU

Figure 4.15 provides a detailed comparison of the performance metrics of genetic algorithms when executed on two distinct computing platforms: CPU and GPU. The x-axis represents the number of generations, ranging from 0 to 1000, while the y-axis indicates the time taken in minutes. The orange line, representing the CPU, demonstrates a more pronounced incline in comparison to the green GPU line. This suggests that as the number of generations increases, the CPU takes progressively more time to compute, whereas GPU maintains a steadier, more linear growth in time consumption. Specifically, around

the 400-generation mark, the gap between CPU and GPU times begins to widen, illustrating the superior efficiency of GPU computations for larger-scale tasks. In essence, while both platforms show an increase in computation time as the number of generations grows, the GPU consistently outperforms the CPU, especially as the tasks become more demanding. This disparity underscores the potential advantages of leveraging GPU acceleration for intensive genetic algorithm computations.

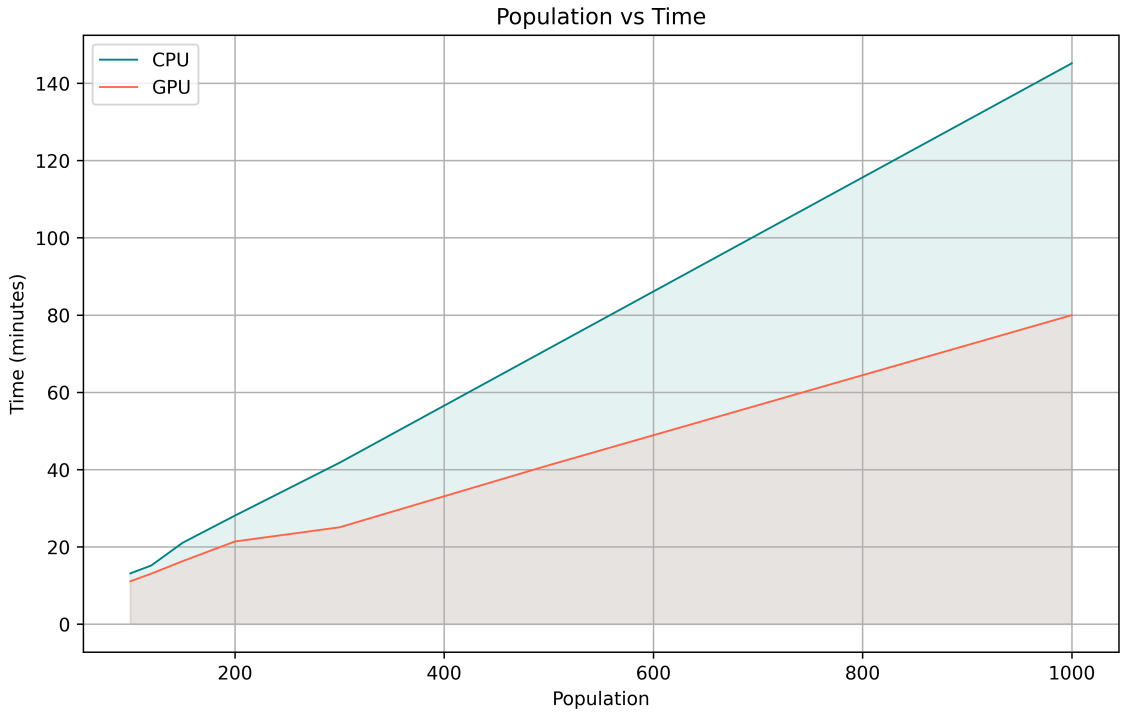


Figure 4.16: Population CPU vs GPU

Figure 4.16 presents an analysis of performance distinctions between CPU and GPU in the context of genetic algorithms. As it examines the x-axis, which represents the population size from 0 to 1000, the y-axis reveals the computational time in minutes. The cyan line associated with the CPU shows a steeper rise, indicating increased time consumption as the population size enlarges. Contrastingly, the GPU, represented by the coralline, maintains a more consistent and less steep trajectory. This behavior can be attributed to the inherent architectural differences between CPUs and GPUs. CPUs, being designed for general-purpose tasks and sequential processing, may struggle with the parallelism required for handling larger populations in genetic algorithms. On the other hand, GPUs, characterized by their multitude of cores and proficiency in parallel processing, are inherently suited to handle large datasets and perform simultaneous computations, making them more efficient for tasks that demand scalable parallelism, such as large population genetic algorithms.

4.3.6 Computational power

Blockchain technology, renowned for its high level of security and transparency, was trialed in a controlled environment to gauge its performance under specific conditions. The chosen consensus mechanism for this blockchain was "proof of work." Throughout the testing phase, the block validation difficulty was intentionally set low. This decision stemmed from the fact that the tests were being run on a personal computer, which isn't the ideal setting for large-scale blockchain operations. In an optimal scenario, dedicated servers would be utilized, with each server housing a single instance of the blockchain, thereby ensuring maximal scalability.

Figure 4.17 offers a clear perspective on the mining capacities of two distinct nodes: the personal computer and the Raspberry. As depicted, the personal computer mined a significantly higher number of blocks compared to the Raspberry. This stark mining disparity can be primarily attributed to two factors. The first is computational power; the personal computer, equipped with more potent hardware, greatly outperforms the Raspberry. The second factor is internet connectivity. The personal computer, boasting a faster and more stable connection, has a definitive advantage over the Raspberry, which may have a more constrained connection.

Considering these insights, the blockchain's performance was good. In all tampering attempts, the blockchain consistently self-corrected, ensuring that the voting process was correct. This resilience underscores the robust design of blockchain and its capability to uphold data integrity, even in challenging situations.

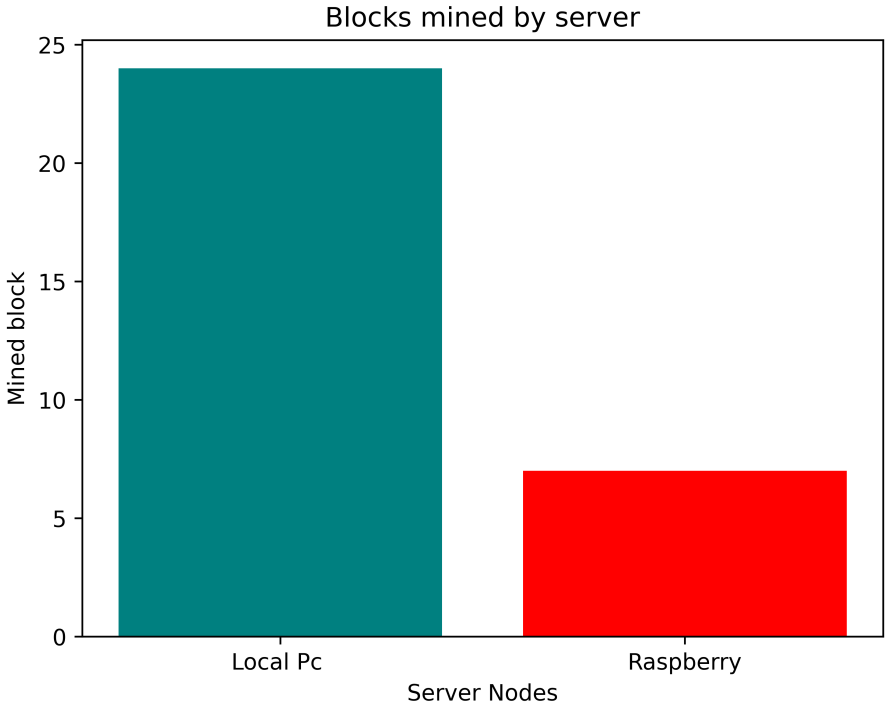


Figure 4.17: Blocks mined by the server

This exercise provided invaluable insights into operating a blockchain under varying hardware and connectivity conditions. The study accentuates the importance of proper hardware and connectivity selection to optimize blockchain performance and highlights the innate resilience of this technology when faced with compromise attempts.

4.3.7 Simulation

In this subsection, a simulation will be conducted to analyze how the system would behave in a real-world usage scenario. Multiple schedules with actual constraints from a year of college will be generated, and users will vote for their preferred schedule. The system's performance in response to this scenario will be assessed.

For this simulation, the previous data from a college will be used. To maintain simplicity, only the data from a specific class will be analyzed, preventing redundant analysis and ensuring the section does not become overly extensive. For simplicity, the genetic algorithm will be configured to generate only two schedules, with a maximum of 100 generations for a population of 100. Regarding the decision-making process will have five valid and eligible users, one valid but ineligible user, and one invalid user. This arrangement allows testing all blockchain validations.

Starting the simulation, the schedules will be configured and generated. The setup procedure was already previously explained in section 4.3.2, so it passes to the result demonstration of the schedules. The class selected for this demonstration was Software Engineering.

The figures in the appendix A 5 represent the two best generations generated by the first module of the system, the genetic algorithm. Both schedules have very close global fitness values, with the first schedule having a fitness of 495 and the second one with 504, this global fitness is the sum of all the fitness of every schedule. Despite their close fitness values, they differ from each other. The first option schedules most of its classes in the morning, whereas the second one has a higher proportion in the afternoon.

Having the generated schedules, the process moves on to the voting decision. In the first instance, each user must have a public address, serving as the key to access their wallet and subsequently the blockchain. Figure 4.18 is a representation of addresses for valid users who can access the blockchain.

```
List Address:
13qQzho77uNk5XYCJJg2GXpNWHpCobqMy5
16QZF3qJvyrmG9peXtigMw1sgf6upm1QpW
1K23ZPRs11oRuYXwVguNXFSsfidD66PNE9
1Lhx7sF8CpbZoo2cpF3vWCVNq9jQYHnug1
1JkKP88j9x5fjR2xNC66cbjogHV5L2ucyr
1AuNhPhmpQrQWqLT47fGrF98osvypGo7Jf
```

Figure 4.18: Public addresses List

With the valid addresses in place, the blockchain initiation phase begins, and this is where the configuration occurs.

Figure 4.19: Init pool config

Figure 4.19 provides an example used for configuring this blockchain. In this example, the "address" field represents the pool host or the user initiating the process. This field is crucial for validating whether the user is legitimate and has permission to initiate a pool. The second field, "voters," represents all users authorized to vote in the initiated poll. It's noteworthy that, as seen in this example, only 5 out of the 6 valid users are authorized for this vote. The third field, "title", serves as a title for the vote to facilitate identification. The "options" field represents the generated schedules. Initiating this request initializes the blockchain, and it returns, the hash of the Genesis block, the first block of the blockchain (figure 4.20).

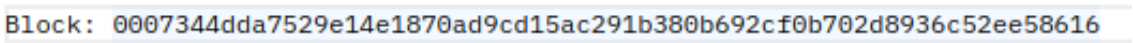


Figure 4.20: Genesis hash

With the blockchain initialized, the registration of votes can be initiated.

Figure 4.21: Add vote configuration

The vote registration proceeds as follows: three fields need to be sent in the request. The "address" represents the user's public key, the "block" represents the hash identifying the vote in which the user wants to participate, and the "option" represents the schedule they want to vote for. At this step, numerous validations take place:

- If the user is valid and eligible to vote, the vote is recorded, and the registration hash is returned (figure 4.22).

```
Parsed data: 169ef98d572372a19e79fcda361d94f9eec37b896e8ccf0fcf455e2aff0ca72e
```

Figure 4.22: Vote successfully added

- If the user is valid but ineligible to vote, the vote is rejected (figure 4.23).

```
Parsed data: User can't vote in this pool
```

Figure 4.23: User with no permissions

- If the user is not valid, the vote is immediately rejected (figure 4.24).

```
Parsed data: Address is not Valid
```

Figure 4.24: User doesn't exist

- If the user is valid and eligible but has already voted, their vote is rejected (figure 4.25).

```
Parsed data: User already vote in this pool
```

Figure 4.25: User already vote

Upon completion of the vote, all valid and eligible users can check the voting status by making the following request (figure 4.26).

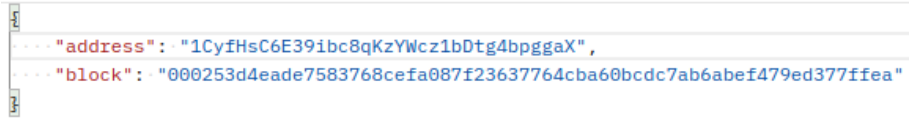


Figure 4.26: Total votes request

In this request, they need to send their public key information and the hash identifying the pool. If valid, they receive the voting status in return (figure 4.27).

```
Total votes for the "First schedule generated": 3
Total votes for the "Second schedule generated": 2
```

Figure 4.27: Pool state

To summarise this simulation, the winning schedule was the first one generated, which had a lower fitness, indicating that the schedule considered with better quality by our system is not necessarily the one preferred by the users. This serves as an example of why the existence of the second module is crucial. Sometimes, a schedule with "lower" quality may be more well-received than one with "higher" quality. The polling concluded with a total of 5 votes, signifying that the blockchain is accurately validating users, as neither the valid but ineligible user nor the invalid user was able to vote.

Chapter 5

Conclusion

The final part of this thesis has taken a closer look at the system, focusing on its innovative approaches to solving the complex problem of university timetable generation. The system is flexible and accurate, carefully coordinating various components to produce optimized timetables that meet the specific needs and constraints of universities.

The system can generate multiple timetable options, each carefully considering and balancing different requirements and preferences. This provides a variety of choices, ensuring that the resulting timetables are both practical and well-suited to the university's needs, making the whole process more efficient and effective.

The system innovation goes beyond just creating optimized timetables. It also introduces a way for people to participate in choosing the best timetable through a secure and transparent voting process, made possible by blockchain technology. This approach ensures that everyone involved can make informed decisions, adding credibility and a democratic spirit to the selection process.

Looking back at the results shared in section 4.3, the system has shown impressive skills in managing and working through various challenges, creating several suitable timetable options. An essential part of the system is the democratic voting feature, which successfully brings together the thoughts and needs of everyone involved, ensuring that the chosen timetables truly meet the community's needs and preferences. This approach encourages a sense of shared responsibility and ownership among the users. It also increases the likelihood that the timetables will be practical, well-received, and widely accepted by the community, as they have a say in the decision-making process.

Additionally, a scientific article was published in MDPI Cryptography journal (Pereira *et al.*, 2023).

Limitations and Future Work

Despite the robust performance and versatility exhibited by the developed system, it is essential to acknowledge its limitations and to contemplate the horizons of potential enhancements and broader applicability that future iterations could explore.

A limitation of our current approach is related to the variety of test cases that have been used to evaluate the system. Although the system has been rigorously tested, there are several areas where it could be further examined. Firstly, there are still various complicated scenarios and unusual constraints that have not been fully explored. Conducting tests in these unexplored areas will help us better understand the system's adaptability and accuracy.

Moreover, to better assess the system's performance and scalability, it would be beneficial to conduct tests on a dedicated server. This approach would allow us to gain valuable insights into how well the system operates under different loads and demands, ensuring that it can handle real-world usage effectively.

The voting system should be tested in scenarios where many people are voting simultaneously. This would help ensure that the system can handle high traffic and that the voting process remains smooth and efficient, even when faced with a large number of users.

There is significant potential for further improvement in the system in the future. Efforts can be made to enhance the planning tools by introducing changeable slot time and expanding the capacity of the blockchain voting system to accommodate a larger number of users. By addressing these factors, the system can enhance its speed, user-friendliness, and overall user experience.

Considering the broader scope, the method has the potential to be applied in several domains beyond the exclusive purpose of generating university timetables. Through implementing modifications and adaptations, it possesses the capability to yield significant impacts in various domains. The concept of developing effective solutions while also allowing individuals to have a voice is universally valuable. By making appropriate adjustments, the system could be beneficial in diverse scenarios such as logistics, healthcare planning, and resource management across different industries, demonstrating its versatility.

In summary, the future path is filled with prospects for ongoing enhancement, growth, and inventive utilization of the system. Future efforts could concentrate on investigating these possibilities, with the goal of strengthening the system's resilience, expanding its usefulness, and improving its ability to optimize operations and facilitate informed decision-making in diverse fields.

Appendix A

Generated schedules in the simulation section 4.3.7.

First Generation

GENE INN11					FIT: 87
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00	SI TP 1 101	IAPG TP 1 87			
10:00	SI TP 1 101				
11:00					
12:00					
13:00			SI PL 1 121		GRC TP 1 13
14:00			SI PL 1 121		GRC TP 1 13
15:00					FIS PL 1 111
16:00					FIS PL 1 111
17:00	ING TP 1 37				
18:00	ING TP 1 37			MAT1 TP 1 120	
19:00	FIS TP 1 1		IAPG PL 1 87	MAT1 TP 1 120	

Figure 1: INN 1 1

GENE INN12					FIT: 71
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00	OPC1 T 2 70	OPC1 T 2 40	ELAP PL 2 87		OPC1 TP 2 54
10:00	ELMG 2 127	OPC1 T 2 40	ELAP PL 2 87	ELAP TP 2 3	OPC1 TP 2 54
11:00		OPC1 TP 2 116	OPC3 TP 2 55	ELAP TP 2 3	OPC1 TP 2 20
12:00		OPC1 TP 2 116			
13:00					
14:00				ANSI TP 2 101	OPC1 TP 2 58
15:00				OPC1 TP 2 7	OPC1 TP 2 58
16:00					
17:00		ESTA TP 2 1		MAT2 TP 2 55	
18:00	OPC3 TP 2 89	ESTA TP 2 1	ANSI PL 2 62	MAT2 TP 2 55	
19:00			ANSI PL 2 62	OPC1 TP 2 38	

Figure 2: INN 1 2

GENE INN21					FIT: 90
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00			LPR1 PL 1 39		
10:00	AED1 PL 1 50		LPR1 PL 1 39		
11:00	AED1 PL 1 50		SDIG PL 1 102		
12:00			SDIG PL 1 102	LPR1 TP 1 51	
13:00					
14:00					
15:00	ACPT PL 1 41				
16:00			SDIG TP 1 26		
17:00					ANUM TP 1 60
18:00	AED1 TP 1 40	ACPT TP 1 23			ANUM TP 1 60
19:00	AED1 TP 1 40				

Figure 3: INN 2 1

GENE INN22					FIT: 91
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00	SOPE TP 2 37	SOPE PL 2 52			
10:00	SOPE TP 2 37	SOPE PL 2 52		AED2 PL 2 3	
11:00	LPR2 PL 2 97			AED2 PL 2 3	
12:00					
13:00	LPR2 PL 2 97	AED2 TP 2 128			
14:00					HRDS PL 2 127
15:00					HRDS PL 2 127
16:00			IO PL 2 105		
17:00	HRDS TP 2 68				IO TP 2 101
18:00	HRDS TP 2 68				
19:00					

Figure 4: INN 2 2

GENE INN31					FIT: 80
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00		BDAD PL 1 84	OPC3 TP 1 48		MUL1 TP 1 118
10:00		BDAD PL 1 84	OPC3 TP 1 48	OPC3 TP 1 91	MUL1 TP 1 118
11:00				OPC3 TP 1 87	
12:00				OPC3 TP 1 87	
13:00					
14:00			RCO1 TP 1 8		
15:00	ESOF PL 1 103		RCO1 TP 1 8		
16:00					ESOF TP 1 40
17:00				BDAD TP 1 14	ESOF TP 1 40
18:00	OPC3 TP 1 59		RCO1 PL 1 58	BDAD TP 1 14	
19:00	OPC3 TP 1 59			MUL1 PL 1 56	

Figure 5: INN 3 1

GENE INN32					FIT: 76
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00			OPC2 TP 2 79	RCO2 TP 2 130	
10:00		OPC2 TP 2 99	OPC2 TP 2 79	RCO2 TP 2 130	ANSI PL 2 107
11:00		OPC4 TP 2 57			
12:00					
13:00		OPC4 TP 2 57			OPC2 TP 2 73
14:00	RCO2 PL 2 99	OPC2 TP 2 114	OPC2 TP 2 86		OPC2 TP 2 73
15:00	RCO2 PL 2 99		OPC2 TP 2 86		
16:00		OPUE T 2 105			OGE TP 2 11
17:00	OPC2 TP 2 79	OPUE T 2 105			MUL2 PL 2 52
18:00	OPC2 TP 2 41	ANSI TP 2 97		OPC2 TP 2 20	MUL2 TP 2 86
19:00		ANSI TP 2 97			

Figure 6: INN 3 2

Second Generation

GENE INN11					FIT: 86
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00	FIS TP 1 77	FIS PL 1 59			
10:00		FIS PL 1 59			GRC TP 1 85
11:00					GRC TP 1 85
12:00					
13:00					
14:00		IAPG TP 1 7		SI TP 1 18	
15:00				SI TP 1 18	
16:00	MAT1 TP 1 97				ING TP 1 4
17:00	MAT1 TP 1 97		IAPG PL 1 80		ING TP 1 4
18:00					SI PL 1 68
19:00					SI PL 1 68

Figure 7: INN 1 1

GENE INN12					FIT: 80
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00					
10:00	ANSI TP 2 113	ELAP PL 2 107			MAT2 TP 2 87
11:00		ELAP PL 2 107	OPC1 TP 2 115		MAT2 TP 2 87
12:00		OPC1 T 2 124	OPC1 TP 2 115	OPC1 TP 2 99	
12:00					
14:00	OPC1 T 2 6	OPC1 T 2 124		OPC3 TP 2 56	ESTA TP 2 78
15:00		OPC1 TP 2 80	ANSI PL 2 14	OPC1 TP 2 114	ESTA TP 2 78
16:00		OPC1 TP 2 80	ANSI PL 2 14	OPC1 TP 2 114	OPC1 TP 2 46
17:00	ELAP TP 2 41				
18:00	ELAP TP 2 41		OPC1 TP 2 31	OPC3 TP 2 131	
19:00			ELMG 2 97		

Figure 8: INN 1 2

GENE INN21					FIT: 87
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00					
10:00					AED1 TP 1 69
11:00	SDIG TP 1 31			LPR1 PL 1 9	AED1 TP 1 69
12:00					
13:00	AED1 PL 1 108			LPR1 PL 1 9	
14:00	AED1 PL 1 108				
15:00				LPR1 TP 1 15	
16:00		ANUM TP 1 73			
17:00	ACPT TP 1 122	ANUM TP 1 73			SDIG PL 1 81
18:00					SDIG PL 1 81
19:00	ACPT PL 1 105				

Figure 9: INN 2 1

GENE INN22					FIT: 89
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00	LPR2 TP 2 74	SOPE TP 2 43			
10:00	LPR2 TP 2 74	SOPE TP 2 43		SOPE PL 2 24	
11:00				SOPE PL 2 24	LPR2 PL 2 51
12:00					
13:00				HRDS TP 2 113	LPR2 PL 2 51
14:00	HRDS PL 2 80			HRDS TP 2 113	
15:00	HRDS PL 2 80	IO PL 2 84	AED2 PL 2 27		
16:00		IO TP 2 66	AED2 PL 2 27	AED2 TP 2 34	
17:00					
18:00					
19:00					

Figure 10: INN 2 2

GENE INN31					FIT: 86
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00					MUL1 TP 1 52
10:00		ESOF PL 1 48	RCO1 TP 1 63		MUL1 TP 1 52
11:00	OPC3 TP 1 128		RCO1 TP 1 63	BDAD PL 1 96	
12:00					
13:00	OPC3 TP 1 128			BDAD PL 1 96	
14:00	BDAD TP 1 33	OPC3 TP 1 52			
15:00	BDAD TP 1 33	OPC3 TP 1 52	OPC3 TP 1 52	ESOF TP 1 104	
16:00				ESOF TP 1 104	
17:00				RCO1 PL 1 128	
18:00			MUL1 PL 1 9	OPC3 TP 1 67	
19:00				OPC3 TP 1 67	

Figure 11: INN 3 1

GENE INN32					FIT: 76
#	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00	OPC2 TP 2 29				
10:00	MUL2 TP 2 45			OPC2 TP 2 39	OPC2 TP 2 102
11:00		MUL2 PL 2 98	OPC2 TP 2 5	OGE TP 2 95	OPC2 TP 2 102
12:00					
13:00					
14:00	OPC2 TP 2 26	OPC2 TP 2 131			OPC4 TP 2 100
15:00		OPC2 TP 2 131		RCO2 TP 2 97	OPC4 TP 2 100
16:00	ANSI TP 2 29		OPUE T 2 48	RCO2 TP 2 97	
17:00	ANSI TP 2 29		OPUE T 2 48		RCO2 PL 2 100
18:00	OPC2 TP 2 27		OPC2 TP 2 10		RCO2 PL 2 100
19:00	OPC2 TP 2 27			ANSI PL 2 11	

Figure 12: INN 3 2

Bibliography

ABRAMSON, D., KRISHNAMOORTHY, M. & DANG, H. (1997). Simulated annealing cooling schedules for the school timetabling problem. *School of Computing and Information Technology, Griffith University*, first Submitted: 15 July 1996, Revised: 6 November 1997. 14, 15

ABUIDRIS, Y., KUMAR, R., YANG, T. & ONGINJO, J. (2019). Secure large-scale e-voting system based on blockchain contract using a hybrid consensus model combined with sharding. *ETRI Journal*. 24

AGORA (n.d.). Agora: Bringing voting systems into the digital age. Accessed: 2023-10-29. 22

AL-JAROODI, J. & MOHAMED, N. (2019). Blockchain in industries: A survey. *IEEE Access*, **7**, 36500–36515. 22

ALFAIN, Z.W., SETIAWAN, H. & BUANA, I.K.S. (2022). Analysis of centralized vs decentralized electronic voting. In *2022 IEEE 8th Information Technology International Seminar (ITIS)*, 173–177. 26

ALGHAMDI, H., ALSUBAIT, T., ALHAKAMI, H. & BAZ, A. (2020). A review of optimization algorithms for university timetable scheduling. *Engineering, Technology & Applied Science Research (ETASR)*, **10**, 6410–6417. 6

ALMAKHOUR, M., SLIMAN, L., SAMHAT, A.E. & MELLOUK, A. (2020). Verification of smart contracts: A survey. *Pervasive and Mobile Computing*, **67**, 101227. 20

AVIS, D. & CHVÁTAL, V. (1978). Notes on bland's pivoting rule. In M.L. Balinski & A.J. Hoffman, eds., *Polyhedral Combinatorics: Dedicated to the memory of D.R. Fulkerson*, 24–34, Springer Berlin Heidelberg, Berlin, Heidelberg. 9

BARBULESCU, L., WATSON, J.P. & WHITLEY, L.D. (2000). Dynamic representations and escaping local optima: Improving genetic algorithms and local search. *AAAI/IAAI*, **2000**, 879–884. 55

BULJUBASIC, M. (2015). *Efficient local search for several combinatorial optimization problems. (Recherche locale performante pour la résolution de plusieurs problèmes combinatoires).*. Ph.D. thesis, University of Montpellier, France. 10

CAO, Y., ZHANG, J., YUAN, X., GUO, T., LU, C., CHEN, G., KANG, J., YAN, X., ZHANG, X. & HUANG, Y. (2019). A hybrid blockchain system based on parallel distributed architecture for central bank digital currency. In A.J. Tallón-Ballesteros, ed., *FSDM*, vol. 320 of *Frontiers in Artificial Intelligence and Applications*, 1138–1145, IOS Press. 22

CHAUHAN, A., MALVIYA, O.P., VERMA, M. & MOR, T.S. (2018). Blockchain and scalability. In *QRS Companion*, 122–128, IEEE. 22

CHEN, L. (2012). Ring group signatures. In G. Min, Y. Wu, L.C. Liu, X. Jin, S.A. Jarvis & A.Y. Al-Dubai, eds., *TrustCom*, 409–418, IEEE Computer Society. 37

CHEN, M., SZE, S., GOH, S.L. & LAU, S. (2022). Investigation of heuristic orderings with a perturbation for finding feasibility in solving real-world university course timetabling problem. In *IEEE*. 12

CLAUSEN, J. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Applied Mathematics*, **123**, 27–60. 7

DANTZIG, G.B. (1990). Origins of the simplex method. *"A history of scientific computing"*. 9

DANTZIG, G.B. (2002). Linear programming. *Operations Research*, **50**, 42–47. x, 9

DELAHAYE, D., CHAIMATANAN, S. & MONGEAU, M. (2019). Simulated annealing: From basics to applications. In M. Gendreau & J.Y. Potvin, eds., *Handbook of Metaheuristics*, 1–35, Springer International Publishing, Cham. 14

DIKALEH, S.G., SHEIKH, O. & FELIX, C. (2017). Introduction to kubernetes. In M. Mindel, K. Lyons & J. Wigglesworth, eds., *CASCON*, 310, IBM / ACM. 36

DORIGO, M., MANIEZZO, V. & COLORNI, A. (1991). Ant system: An autocatalytic optimizing process. In *IEEE*, 91-016. 12

EL KHATIB, M., AL MULLA, A. & AL KETBI, W. (2023). The role of blockchain in e-governance and decision-making in project and program management. *Scientific Research Publishing (SCIRP)*. 5, 6

ERNEST, A. & HOURT, N. (2013). Follow my vote - a transparent and secure voting system. `https://followmyvote.com/`, [Online; accessed March 08, 2023]. 21, 23, 24, 25

FULLMER, D. & MORSE, A.S. (2018). Analysis of difficulty control in bitcoin and proof-of-work blockchains. In *2018 IEEE Conference on Decision and Control (CDC)*, 5988–5992. 58

FUSCO, F., LUNESU, M.I., PANI, F.E. & PINNA, A. (2018). Crypto-voting, a blockchain based e-voting system. *SCITEPRESS*. 23

GLOVER, F.W. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, **13**, 533–549. x, 10, 11

GLOVER, F.W. & KOCHENBERGER, G.A. (2003). Handbook of metaheuristics. Hardcover. 11, 13

GOERIGK, M. & SCHÖBEL, A. (2011). Engineering the modulo network simplex heuristic for the periodic timetabling problem. In P.M. Pardalos & S. Rebennack, eds., *Experimental Algorithms*, 181–192, Springer Berlin Heidelberg, Berlin, Heidelberg. 10

GOLDREICH, O., MICALI, S. & WIGDERSON, A. (1986). How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In A.M. Odlyzko, ed., *CRYPTO*, vol. 263 of *Lecture Notes in Computer Science*, 171–185, Springer. 37

GUAN, B., ZHAO, Y. & LI, Y. (2021). An improved ant colony optimization with an automatic updating mechanism for constraint satisfaction problems. *Expert Systems with Applications*, **165**, 114021. 12

HE, H., III, H.D. & EISNER, J. (2014). Learning to search in branch and bound algorithms. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence & K.Q. Weinberger, eds., *NIPS*, 3293–3301. 8

HEGADEKATTI, K. (2016). Democracy 3.0: Voting through the blockchain. *Available at SSRN 2889291*. 22

HOLLAND, J.H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press. x, 16, 17

J MICHAEL, J.B., A COHN (2018). Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities. *Litigation*, **44**, 1–8. 20

JADHAV, R.A.P..S.R.R..V.M.J..P.S.P. (2020). Maintaining security of electronic voting data through block chains. *International Journal of Trend in Scientific Research and Development*, **4**, 245–250. 22

JAFAR, U., AZIZ, M.J.A. & SHUKUR, Z. (2021). Blockchain for electronic voting system—review and open research challenges. *Sensors*, **21**, 5874. 7, 24

JAIN, N.K., NANGIA, U. & JAIN, J. (2018). A review of particle swarm optimization. *Journal of The Institution of Engineers (India): Series B*, **99**, 407–411. 15

KENNEDY, J. & EBERHART, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1942–1948 vol.4. x, 15, 16

KHAN, K.M., ARSHAD, J. & KHAN, M.M. (2018a). Secure digital voting system based on blockchain technology. *Int. J. Electron. Gov. Res.*, **14**, 53–62. 22

KHAN, K.M., ARSHAD, J. & KHAN, M.M. (2018b). Secure digital voting system based on blockchain technology. *IGI Global*. 24

KIRKPATRICK, S., GELATT, C.D. & VECCHI, M.P. (1983). Optimization by simulated annealing. *Science*, **220**, 671–680. 13

KOSHELEVA, O., KREINOVICH, V. & DUMRONGPOKAPHAN, T. (2020). A symmetry-based explanation of the main idea behind chubanov's linear programming algorithm. *Algebraic Techniques and Their Use in Describing and Processing Uncertainty: To the Memory of Professor Elbert A. Walker*, 55–64. 9

KRICHEN, M., LAHAMI, M. & AL–HAIJA, Q.A. (2022). Formal methods for the verification of smart contracts: A review. In *2022 15th International Conference on Security of Information and Networks (SIN)*, 01–08. 20

LEE, W. & KIM, H.Y. (2005). Genetic algorithm implementation in python. In *Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, 8–11. 16

LI, Y. & MCCLELLAND, J.L. (2022). A weighted constraint satisfaction approach to human goal-directed decision making. *PLOS COmputacion Biology*. 5

LIN, I. (2005). Particle swarm optimization for solving constraint satisfaction problems (msc. thesis), simon fraser univerisity. 15

LUKE, S. (2013). *Essentials of Metaheuristics*. Lulu, 2nd edn., available for free at https://cs.gmu.edu/ sean/book/metaheuristics/Essentials.pdf. 10

MADUREIRA, A. (2010). *Técnicas Emergentes de Optimização no Suporte à Tomada de Decisão*. Ph.D. thesis, Instituto Superior de Engenharia do Porto. 11

MAHLOUS, A.R. & MAHLOUS, H. (2023). Student timetabling genetic algorithm accounting for student preferences. *PeerJ Computer Science*. 6

MARTIN, Q. (2022). From faculty to administration: preparing the next generation of academic leaders. *Perspectives: Policy and Practice in Higher Education*, **26**, 109–114. 1

MAVROTAS, G., GEORGOPOULOU, E., MIRASGEDIS, S., SARAFIDIS, Y., LALAS, D., HONTOU, V. & GAKIS, N. (2007). An integrated approach for the selection of best available techniques (bat) for the industries in the greater athens area using multi-objective combinatorial optimization. *Energy Economics*, **29**, 953–973. 5

MERKEL, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, **2014**. 36

MOUBARAK, J., FILIOL, E. & CHAMOUN, M. (2018). On blockchain security and relevant attacks. In *MENACOMM*, 1–6, IEEE. 23

NAKAMOTO, S. (2009). Bitcoin: A peer-to-peer electronic cash system. 20

NOTHEGGER, C., MAYER, A., CHWATAL, A. & RAIDL, G.R. (2012). Solving the post enrolment course timetabling problem by ant colony optimization. *Annals of Operations Research*. 13

OGBURN, M., TURNER, C. & DAHAL, P. (2013). Homomorphic encryption. In C.H. Dagli, ed., *Complex Adaptive Systems*, vol. 20 of *Procedia Computer Science*, 502–509, Elsevier. 36

PANDEY, H.M., CHAUDHARY, A. & MEHROTRA, D. (2014). A comparative review of approaches to prevent premature convergence in ga. *Applied Soft Computing*, **24**, 1047–1077. 17

PASUPULATI, R.P. & SHROPSHIRE, J. (2016). Analysis of centralized and decentralized cloud architectures. In *SoutheastCon 2016*, 1–7. 25

PEREIRA, B.M.B., TORRES, J.M., SOBRAL, P.M., MOREIRA, R.S., SOARES, C.P.D.A. & PEREIRA, I. (2023). Blockchain-based electronic voting: A secure and transparent solution. *Cryptography*, **7**. 3, 67

PURANIK, N.C..Y. (2021). Blockchain and beyond. *International Journal of Trend in Scientific Research and Development*, **5**, 874–878. 20

RIVEST, R.L., SHAMIR, A. & TAUMAN, Y. (2001). How to leak a secret. In C. Boyd, ed., *ASIACRYPT*, vol. 2248 of *Lecture Notes in Computer Science*, 552–565, Springer. 37

SAWHNEY, N. (2015). Voatz - a secure mobile voting platform. https://www.voatz.com/, [Online; accessed March 08, 2023]. 21, 23, 24, 25

SHINANO, Y., HIGAKI, M. & HIRABAYASHI, R. (1995). A generalized utility for parallel branch and bound algorithms. In *Proceedings.Seventh IEEE Symposium on Parallel and Distributed Processing*, 392–401. x, 7, 8

SINGH, K., HEULOT, N. & HAMIDA, E.B. (2018). Towards anonymous, unlinkable, and confidential transactions in blockchain. In *iThings/GreenCom/CPSCom/SmartData*, 1642–1649, IEEE. 20

SKELLA, J. & NAAMANI, N. (2017). Horizon state - the future of digital voting. https://horizonstate.com/, [Online; accessed March 08, 2023]. 21, 23, 25

SONTAKKE, M.A.P.J.P.K.R. (2017). Design and analysis of chain block system for evaluation of brake load. *International Journal of Trend in Scientific Research and Development*, **1**, 1237–1242. 22

SULONG, W.N., SERMSOOK, K., SOOKNIT, O. & WORAPUN, W. (2023). Usage status, usage requirements, and satisfaction in using massive open online course (mooc) for general education courses at rajamangala university of technology srivijaya. *Journal of Education and Learning*, **12**, 117. 1

TALBI, E. (2009). *Metaheuristics: From Design to Implementation*. Wiley. x, 10, 12, 13

TANANA, D. (2019). Avalanche blockchain protocol for distributed computing security. In *BlackSeaCom*, 1–3, IEEE. 22

THYER, M., KUCZERA, G. & BATES, B.C. (1999). Probabilistic optimization for conceptual rainfall-runoff models: A comparison of the shuffled complex evolution and simulated annealing algorithms. *Water Resources Research*, **35**, 767–773. x, 13, 14

VOGET, S. & KOLONKO, M. (1998). Multidimensional optimization with a fuzzy genetic algorithm. *Journal of Heuristics*. 6

WANG, H., MA, S., DAI, H.N., IMRAN, M. & WANG, T. (2020). Blockchain-based data privacy management with nudge theory in open banking. *Future Gener. Comput. Syst.*, **110**, 812–823. 20

WANG, W., HOANG, D.T., HU, P., XIONG, Z., NIYATO, D., WANG, P., WEN, Y. & KIM, D.I. (2019). A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, **7**, 22328–22370. 20

WRIGHT, A.H. (1991). Genetic algorithms for real parameter optimization. In G.J. RAWLINS, ed., *Foundations of Genetic Algorithms*, vol. 1, 205–218, Elsevier. 16

XU, J. & ZHANG, J. (2014). Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis. In *Proceedings of the 33rd Chinese Control Conference*, 8633–8638. 14

ZITNIK, M. (2013). Zero-knowledge proofs. *ACM Crossroads*, **20**, 65–67. 37