



CT099-3-M-CIO

COMPUTATIONAL INTELLIGENCE OPTIMIZATION

Assignment 2 – Project Proposal

Title:

Utilizing Genetic Algorithms for Optimal Academic Scheduling: An Application to Multiple Degree Programs

Student Name	KOH JIA YI
TP Number	TP072780
Intake Code	APDMF2211AI
Programme	MSc in Artificial Intelligence
Module Lecturer	Assoc. Prof. Dr. Imran Medi

Table of Contents

Abstract.....	3
1 Introduction.....	4
2 Related Works	5
3 Methodology	7
3.1 The data.....	7
3.2 Encoding Process	11
3.3 Selection Method	12
3.4 Crossover Method.....	14
3.5 Mutation Method	16
3.6 Fitness Function	17
3.7 Experiment Setup.....	20
4 Results and Discussion	22
4.1 Baseline setup	22
4.2 Random Search	24
4.3 Selection Method and Crossover Method.....	25
4.4 Final Configuration.....	26
4.5 Comparison of Baseline Algorithm and Final Algorithm	28
5 Limitation and Future Works	30
6 Conclusion	31
7 References.....	32

Abstract

This study explores the application of Genetic Algorithms (GA) to the complex problem of academic scheduling for multiple degree programs. Traditional methods often struggle to balance hard constraints (e.g., module prerequisites) and soft preferences (e.g., spreading challenging modules), while also minimizing module delivery costs. We implemented a GA, optimized its parameters via random search, and assessed the impact of selection and crossover operators on performance. The final GA demonstrated significant improvements over the baseline, effectively managing the intricate requirements of the scheduling task. Our findings underscore the potential of GAs for such real-world optimization problems, laying a foundation for further research in this area.

1 Introduction

The increasing complexity and diversification of educational programs in universities necessitates innovative and efficient methods for course scheduling. Traditionally, this process involves considerable time and resources, often resulting in suboptimal solutions. Effective scheduling requires accommodating the needs and constraints of multiple stakeholders, including students, faculty, and administrators. Consequently, an optimized timetable should allow students to complete their program within the stipulated timeframe, minimize operational costs, and optimize resource allocation.

In particular, the challenge intensifies for part-time modular programs that have multiple intakes per year, with students at different stages in their programs. Not only does the scheduling need to ensure that students complete their course within the specified 2.5 years, but it also needs to consider operational costs, as running each module incurs substantial expenditure for the university.

Addressing this complexity calls for sophisticated solutions that can handle the multi-objective, constraint-based nature of course scheduling. The advent of computational intelligence (CI) presents promising prospects for tackling this task. Among the CI methods, Genetic Algorithm (GA) stands out as a versatile, powerful tool for optimization problems. With its inherent ability to search for optimal solutions in large, complex spaces, GA is aptly suited for course scheduling optimization.

This project aims to design and implement a Genetic Algorithm-based solution to optimize course scheduling at APU. The proposed GA-based solution considers critical constraints like limiting the number of modules per intake, ensuring modules do not run consecutively, scheduling specific modules only at stipulated times of the year, and minimizing the number of students without a module to take in a given intake. The findings from this investigation are expected to offer valuable insights and practical tools for efficient course scheduling, ultimately contributing to improved educational delivery and resource management in the university context.

2 Related Works

The research conducted by (Rezaeipanah, A., Matoori, S.S, & Ahmadi, G, 2021) presents a novel approach, IPGALS, to the University Course Timetabling Problem (UCTP). The authors modify the Genetic Algorithm (GA), fusing it with a parallel structure and Local Search (LS) to better tackle the problem. The IPGALS employs the 'island model' of the Parallel Genetic Algorithm, which contributes to the diversity of solutions. It also proposes a unique chromosome representation model for each event, ensuring that the hard constraints of the timetable are always met. The authors integrate a variety of crossover and mutation operators into the IPGALS to increase its optimization potential. The elitism operator is incorporated to preserve high-quality solutions for subsequent generations. Local Search is used to further refine solutions by introducing unused genes into the solution. Shared memory stores the best chromosomes across all parallel GAs to improve future solutions. Despite the promising approach, IPGALS has limitations with large instances of the problem. The authors suggest future work to refine genetic operators, explore new neighbourhood techniques, and further analyse individual components to enhance the performance of IPGALS. They argue that a proper combination and arrangement of advanced genetic and heuristic operators could greatly improve the GA's performance for UCTP.

Furthermore, the research by (Jian & Ning-Ning, 2013) explored the application of the Genetic Algorithm (GA) in timetable creation. This process involves satisfying 'hard constraints' (fixed, necessary conditions) and 'soft constraints' (flexible, preferential conditions). The genetic algorithm is used to create an encoded timetable problem, which is optimized through operations such as selection, crossover, and mutation. A fitness function is also employed to measure the quality of the timetable solution. Despite the GA's effectiveness in solving multi-objective optimization problems like course timetabling, further research and development are needed due to various uncertainties and control parameters. In cases where resources are lacking, or constraint conditions are too severe, manual intervention might be required. Future work could focus on improving the scheduling algorithms' versatility, studying the impact of parameter variation, and exploring related problem fields.

In the study by (Sapru, Reddy, & B.Sivaselvan, 2010) presents a Genetic Algorithm (GA) with a novel Guided Mutation operator for timetable scheduling. Simulations show that this algorithm can reliably converge to feasible solutions quicker than traditional GA. The use of Guided Mutation introduces greater genetic variation, enabling the exploration of new points in the solution space. The findings suggest that this approach could be effectively applied to

similar scheduling problems. Future work will focus on further performance studies of this algorithm.

Moreover, the paper by (Alnowaini & Aljomai, 2021) proposes a genetic algorithm-based solution to university scheduling problems. It deals with various constraints, including scheduling conflicts and room capacity. The system employs a dynamic chromosome model in its algorithm, with an encoding process involving selection, crossover, and mutation of chromosomes. The system uses a fitness function to measure the appropriateness of timetabling. The paper also evaluates the benefits and limitations of the proposed system, showing a 93% optimality rate in initial tests. The new system aims to reduce scheduling time and cost, increase flexibility, but it has limitations like the inability to accommodate all constraints.

The discourse presented by (Akkan & Gulcu, 2018) advances a novel approach to the curriculum-based university timetabling problem (UCTP) by incorporating robustness in solution generation. This work focuses on providing adaptive timetables that can handle disruptions, specifically time change requests. Utilizing a hybrid multi-objective genetic algorithm (MOGA), supplemented with hill-climbing and simulated annealing, the problem is modeled as a bi-criteria optimization problem. The algorithm's performance, tested on ITC-2007 instances, reveals high robustness and high-quality solutions, offering diverse alternatives for decision-makers.

In conclusion, the literature presents a range of studies focusing on the application and enhancement of Genetic Algorithms to address the university course timetabling problem. The reviewed works share common features, most notably the utilization of GAs and the addition of advanced techniques to improve the algorithm's performance and solutions' quality. It's evident that dealing with constraints in timetable scheduling is a central concern across all studies, but the approach to addressing these constraints differs, showcasing the diversity in methods.

3 Methodology

The methodology employed in this project involves a systematic approach for creating an optimized course schedule using a Genetic Algorithm (GA). This approach focuses on three specific courses: MSc in Artificial Intelligence (AI), MSc in Data Science and Business Analytics (DSBA), and MSc in Software Engineering (SE).

3.1 The data

This project operates on three courses: MSc in Artificial Intelligence (AI), MSc in Data Science and Business Analytics (DSBA), and MSc in Software Engineering (SE). Each course consists of various modules, some of which are core modules, while others are electives or specialisation modules.

The modules that are included in each course are detailed in the diagram below:

Module	Module Code	Module Name	
PIP	CT088-0-M	Programming In Python	Pre-requisite
IRP	CT119-0-M	Introduction to R Programming	
FAI	CT118-0-M	Fundamentals of Artificial Intelligence	
AI	CT098-3-M	Artificial Intelligence	Core
IPCV	CT103-3-M	Image Processing and Computer Vision	
FL	CT102-3-M	Fuzzy Logic	
AML	CT046-3-M	Applied Machine Learning	
CIO	CT099-3-M	Computational Intelligence Optimization	
NLP	CT052-3-M	Natural Language Processing	
RMCE*	CT095-6-M	Research Methodology in Computing and Engineering	
AR	CT097-3-M	Applied Robotics	Electives (choose 3)
PR	CT014-3-M	Pattern Recognition	
ESKE	CT101-3-M	Expert Systems and Knowledge Engineering	
BIS	CT048-3-M	Business Intelligence Systems	
MMDA	AQ049-3-M	Multivariate Methods for Data Analysis	
DL	CT100-3-M	Deep Learning	
Proj	CT096-12-M	Project	Project

Figure 1. Module list for MSc in Artificial Intelligence.

Module	Module Code		
BDAT	CT047-3-M	Big Data Analytics & Technologies	Core
DM	CT051-3-M	Data Management	
BIS	CT048-3-M	Business Intelligence Systems	
AML	CT046-3-M	Applied Machine Learning	
RMCP**	CT087-3-M	Research Methodology for Capstone Project	
MMDA	AQ049-3-M	Multivariate Methods for Data Analysis	
DAP*	CT050-3-M	Data Analytical Programming	
ABAV	CT045-3-M	Advanced Business Analytics and Visualization	
CPI***	CT085-5-M	Capstone Project 1	
CP2***	CT086-7-M	Capstone Project 2	
» BUSINESS INTELLIGENCE PATHWAY			
BSSMMA	BM031-3-M	Behavioural Science, Social Media and Marketing Analytics	Specialisation
TSF	AQ050-3-M	Time Series Analysis and Forecasting	
MDA	AQ051-3-M	Multilevel Data Analysis	Electives (Choose 1)
SEM	BM026-3-M	Strategies in Emerging Markets	
ORO	AQ052-3-M	Operational Research and Optimization	
» DATA ENGINEERING PATHWAY			
CIS	CT105-3-M	Cloud Infrastructure and Services	Specialisation
DL	CT100-3-M	Deep Learning	
NLP	CT052-3-M	Natural Language Processing	Electives (Choose 1)
BIA	CT106-3-M	Building IoT Applications	
DPM	CT107-3-M	Data Protection and Management	

Figure 2. Module list for MSc in Data Science and Business Analytics, which includes 2 pathways with their own specialize pathway and electives.

Module	Module Code	Module Name	
SDD	CT089-0-M	Software Design and Development	Pre-requisite
SDM	CT091-0-M	System Development Methods	
BESP	BM070-0-M	Business Environment and Strategic Planning	
SEP	CT090-0-M	Software Engineering Principles	
MSDP	CT067-3-M	Managing Software Development Projects	Core
RELM	CT074-3-M	Reliability Management	
OOSSE	CT071-3-M	Object Oriented Software Systems Engineering	
SESE	CT078-3-M	Software Engineering Support Environments	
SQE	CT079-3-M	Software Quality Engineering	
SECT	CT077-3-M	Security Technologies	
RMCE*	CT095-6-M	Research Methodology in Computing and Engineering	
IA	CT069-3-M	Internet Applications	Electives (choose 3)
NDP	CT070-3-M	Network Design & Performance	
BDAT	CT047-3-M	Big Data Analytics and Technologies	
DM	CT051-3-M	Data Management	
NLP	CT052-3-M	Natural Language Processing	
Proj	CT096-12-M	Project	

Figure 3. Module list for MSc in Software Engineering.

The primary objective of this study is to generate an optimized course schedule, represented as a chromosome in our Genetic Algorithm, such that students have modules to undertake during every intake period. To accomplish this, actual student data is utilized, which provides a realistic basis for our scheduling approach.

A vital step in the methodology involves the extraction of modules already completed by the students from the existing course data. This ensures that the algorithm does not schedule modules that a student has already completed, thereby facilitating their academic progression.

The student's name and the modules they've taken are extracted and saved into a dictionary, with the student's name acting as the key. This process is visually represented in the following diagram.

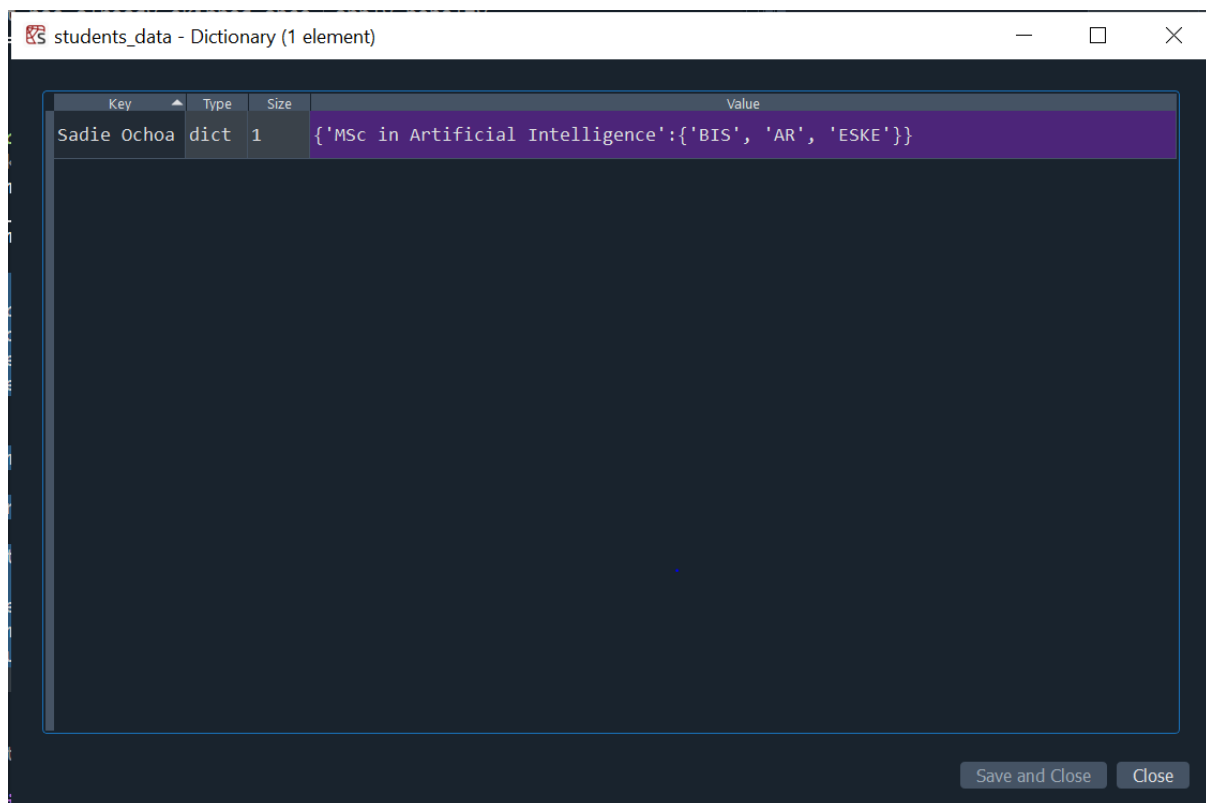
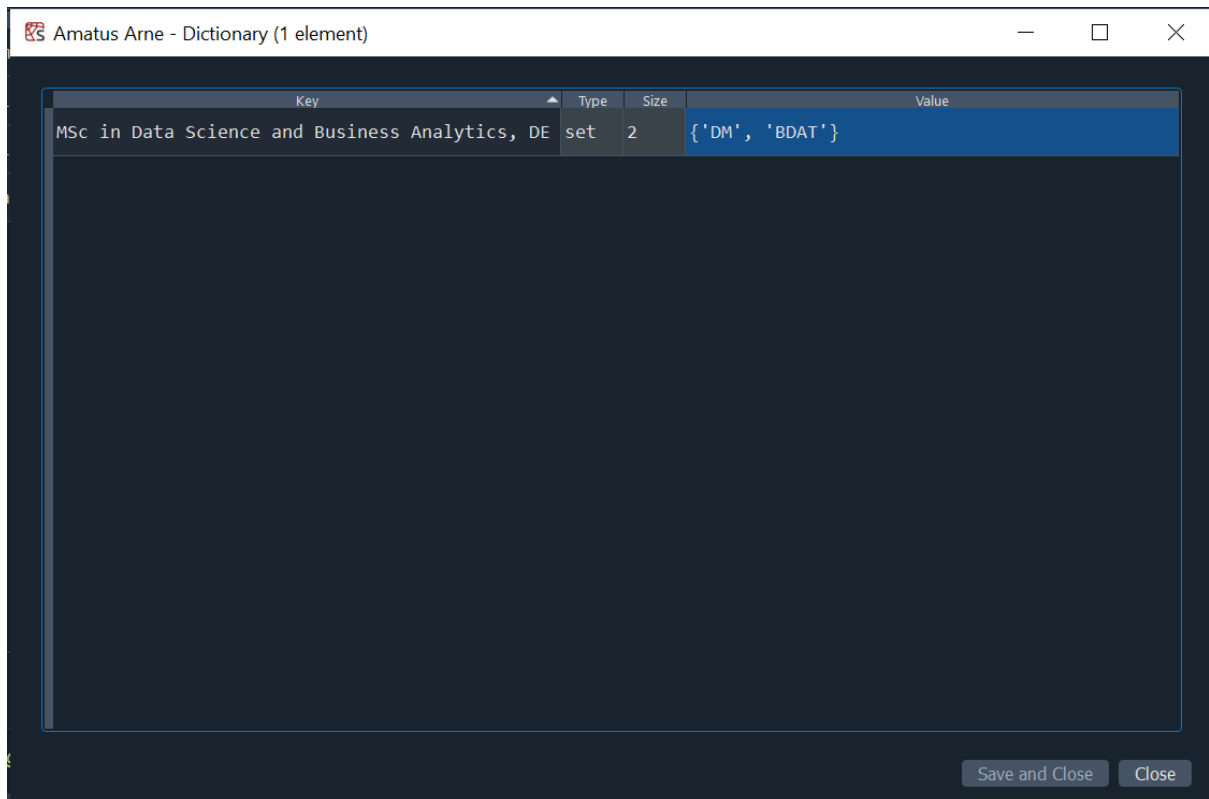


Figure 4. Students taken modules stored in a dictionary with student's name acting as the key.

Additionally, as students of Data Science have specific academic pathways, it is necessary to extract the pathway data as well. This data extraction and its implications for the scheduling process are also illustrated in the subsequent diagram.



Key	Type	Size	Value
MSc in Data Science and Business Analytics, DE	set	2	{'DM', 'BDAT'}

Figure 5. Data science students are required to also store the pathways.

The implementation of this strategy allows our GA to generate a more accurate and efficient schedule, thereby contributing to a more effective educational experience for each student.

3.2 Encoding Process

Chromosome Structure:

The configuration of the chromosome structure can be demonstrated with the accompanying diagram:

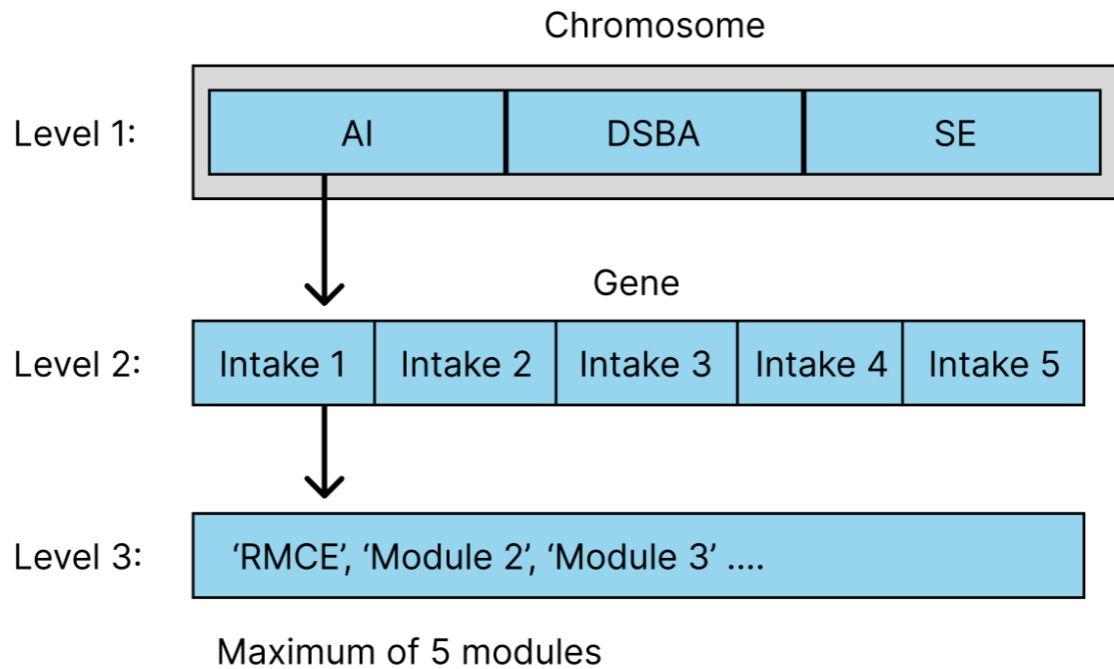


Figure 6. This figure visualizes the three-tier structure of the chromosome, starting with a nested list comprising three sub-lists. Each of these sub-lists symbolizes a specific course. Further within each course-based sub-list, five more sub-lists are housed, each corresponding to one of five distinct intakes. In the deepest layer of this structure, each intake-based sub-list contains a random assortment of modules, with the maximum count capped at five.

The structure of the chromosome, as visualized in Figure 4, follows a three-tier architecture. The initial layer of the chromosome is a nested list comprised of three distinct sub-lists, each representing a unique course: AI, DS, and SE.

Proceeding to the second layer of the structure, the contents of each course-based sub-list are further divided into five additional sub-lists. These represent the five different intakes (intake 1 – 5).

The deepest layer reveals the contents of each intake-based sub-list. Here, a variable number of modules (up to a maximum of five) are present. The modules are randomly chosen from the relevant module lists. To ensure diversity and flexibility in the schedule, the selection process is designed to prevent module repetitions within each intake. Furthermore, if a module is

chosen for a particular intake, it is excluded from the selection for the subsequent intake. However, repetitions of modules across non-consecutive intakes are permissible, thereby providing a level of flexibility to the scheduling process. Furthermore, as a pre-determined factor, the 'RMCE' module is incorporated into the first, third, and fifth intakes.

3.3 Selection Method

In the realm of Genetic Algorithms (GAs), the selection method is a process that chooses individuals, or "chromosomes," from the current population to create a pool of parents for the next generation. Selection methods aim to favour individuals that have higher fitness scores, based on the principle of survival of the fittest. However, they usually also maintain a chance for less fit individuals to become parents, to maintain diversity in the population and avoid premature convergence.

There are several types of selection methods used in GAs:

1. **Tournament Selection:** This method involves running "tournaments" among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover. The tournament size can be varied to control the selection pressure. With larger tournament sizes, the selection process becomes more deterministic, and the best individuals are more often selected (Sheng Fang, 2010). Conversely, with smaller sizes, selection is more random and gives lesser fit individuals a better chance.
2. **Roulette Wheel Selection:** Also known as fitness proportionate selection, this method involves giving each individual a slice of a circular "roulette wheel," where the size of the slice is proportional to the individual's fitness. The wheel is spun, and the individual whose slice the spinner lands on is selected. This method gives better individuals a higher chance of being selected, but still allows less fit individuals to be chosen.
3. **Rank Selection:** This method first ranks the population according to fitness, then every chromosome receives selection probability according to its rank. Individuals are then selected based on these probabilities. This method can be useful when the fitness values vary greatly, as it can help to prevent premature convergence by focusing more on the rank order of fitness rather than the absolute fitness values.
4. **Elitism:** While not a standalone selection method, elitism is a technique that involves directly passing the best individual(s) in the current generation to the next generation.

This ensures that the best solutions found so far are not lost due to the stochastic processes of selection, crossover, and mutation.

In the context of this project, the selection method plays a crucial role in finding the most optimal scheduling of courses. The project utilizes a Tournament Selection method, where "tournaments" are held among a subset of individuals randomly chosen from the population. The goal of these tournaments is to select the best candidates (in this case, the best course schedules) to be passed on to the next generation.

Each tournament is conducted among three individuals, as indicated by $\text{tournamentsize}=3$. Within these mini-tournaments, the course schedule with the highest fitness score - meaning the one that best meets the specified constraints and objectives - is declared the winner. This "winning" schedule is then selected as a parent for generating the next generation.

The selection process is repeated multiple times until a sufficient number of parents have been chosen for the next generation. This ensures that the most successful course schedules (those that best optimize the shared modules, cost, and student requirements) are more likely to pass their characteristics to the next generation of schedules, improving the population over time.

It's worth noting that Tournament Selection also offers an opportunity for less optimal schedules to be selected sometimes. This maintains diversity within the population, preventing premature convergence on sub-optimal solutions and helping the genetic algorithm explore a wider range of potential schedules.

By utilizing this Tournament Selection process, the project effectively guides the evolution of course schedules towards more optimal arrangements, improving the quality of course management and potentially leading to better student satisfaction and resource utilization.

3.4 Crossover Method

In the realm of genetic algorithms, crossover operations are pivotal as they simulate the biological process of recombination, enabling the exchange of genetic material between two parent chromosomes to generate offspring. The one-point crossover is among the simplest and most commonly used crossover techniques.

Consider two parent chromosomes:

Parent 1: [1, 1, 0, 1, 0, 1, 0, 1]

Parent 2: [0, 0, 1, 0, 1, 0, 1, 0]

The one-point crossover technique begins with the selection of a random crossover point in the parent chromosomes. The same crossover point is used in both parents.

Assume that the third position is the chosen crossover point:

Parent 1: [1, 1, | 0, 1, 0, 1, 0, 1]

Parent 2: [0, 0, | 1, 0, 1, 0, 1, 0]

The "|" symbol represents the crossover point. After this point, the segments of the two parent chromosomes are swapped, generating two offspring:

Offspring 1: [1, 1, | 1, 0, 1, 0, 1, 0]

Offspring 2: [0, 0, | 0, 1, 0, 1, 0, 1]

The crossover operation thus gives birth to new chromosomes, or solutions, that incorporate elements from both parents.

In the context of this problem, each chromosome is a three-dimensional list where each sublist represents a specific course, e.g., AI, DSBA, SE. The five sublists within each course represent different intakes. Furthermore, each of these sublists contains the modules for that specific intake.

For example, consider two chromosomes (schedules). Two points are randomly chosen within the parents' chromosomes, and the subsequence between these points are exchanged. In this example, the points are chosen before the second and fourth elements. The operation unfolds as follows:

Parent 1: [['AR','ESKE', 'PR', 'BIS'], | ['FL', 'IPCV', 'DL'], ['RMCE', 'CIO', 'ESKE', 'AI'], ['NLP', 'DL'], | ['RMCE', 'FL']], [['RMCE', 'BSSMMA', 'NLP'], ['DL', 'TSF', 'CIS'], ['RMCE', 'BSSMMA'], ['DAP', 'CIS', 'DL', 'BIS'], ['RMCE', 'TSF', 'MDA', 'DM', 'BSSMMA']], [['RMCE', 'NLP'], ['SESE', 'NDP', 'RELM', 'BDAT', 'SQE'], ['RMCE', 'IA'], ['NLP', 'OOSSE', 'SECT'], ['RMCE', 'DM']]]

Parent 2: [['DL', 'FL', 'AR', 'ESKE'], | ['BIS', 'AI', 'CIO', 'RMCE'], ['PR', 'NLP', 'DL'], | ['RMCE', 'FL'], ['AR', 'ESKE', 'PR', 'BIS']], [['RMCE', 'TSF', 'MDA', 'DM', 'BSSMMA'], ['DL', 'BIS', 'CIS', 'DAP'], ['RMCE', 'BSSMMA', 'NLP'], ['DL', 'TSF', 'CIS'], ['DAP', 'CIS', 'DL', 'BIS']], [['SESE', 'NDP', 'RELM', 'BDAT', 'SQE'], ['RMCE', 'NLP'], ['IA', 'NLP', 'OOSSE', 'SECT'], ['RMCE', 'DM'], ['RMCE', 'IA']]]

Following the execution of the two-point crossover operation, the resulting offspring are (swapped elements are highlighted in yellow):

Offspring 1: [['AR','ESKE', 'PR', 'BIS'], ['BIS', 'AI', 'CIO', 'RMCE'], ['PR', 'NLP', 'DL'], ['NLP', 'DL'], ['RMCE', 'FL']], [['RMCE', 'BSSMMA', 'NLP'], ['DL', 'BIS', 'CIS', 'DAP'], ['RMCE', 'BSSMMA', 'NLP'], ['DL', 'TSF', 'CIS'], ['RMCE', 'TSF', 'MDA', 'DM', 'BSSMMA']], [['RMCE', 'NLP'], ['RMCE', 'NLP'], ['IA', 'NLP', 'OOSSE', 'SECT'], ['NLP', 'OOSSE', 'SECT'], ['RMCE', 'DM']]]

Offspring 2: [['DL', 'FL', 'AR', 'ESKE'], ['FL', 'IPCV', 'DL'], ['RMCE', 'CIO', 'ESKE', 'AI'], ['RMCE', 'FL'], ['AR', 'ESKE', 'PR', 'BIS']], [['RMCE', 'TSF', 'MDA', 'DM', 'BSSMMA'], ['DL', 'TSF', 'CIS'], ['RMCE', 'BSSMMA'], ['DAP', 'CIS', 'DL', 'BIS'], ['DAP', 'CIS', 'DL', 'BIS']], [['SESE', 'NDP', 'RELM', 'BDAT', 'SQE'], ['SESE', 'NDP', 'RELM', 'BDAT', 'SQE'], ['RMCE', 'IA'], ['RMCE', 'DM'], ['RMCE', 'IA']]]

In these offspring, it can be observed that the segments from Parent 1 and Parent 2 that lie between the crossover points have been swapped, producing new sequences in the offspring. As before, each offspring is a new chromosome that represents a schedule which has inherited

elements from both parents. The precise location of the crossover point (or points in multi-point crossover) can significantly influence the performance of the genetic algorithm and is typically chosen randomly in each generation.

In the scope of this project, an array of crossover techniques will undergo rigorous examination. The findings drawn from these tests will be comprehensively elaborated in the forthcoming results section.

3.5 Mutation Method

The function starts by randomly selecting one intake from a course. The selection of an empty intake results in no mutation, therefore, the original intake is returned.

If the intake is not empty, the function continues by choosing a random index within this intake, representing a specific module in the intake. For instance, within the second intake ['FL', 'IPCV', 'DL'] of the first course of the given chromosome, it might randomly select the index 1 corresponding to the module 'IPCV'.

Subsequently, a list of possible replacement modules is created, containing all modules that are not currently in the selected intake.

However, if no replacement modules are available, the function returns the original intake without any mutations. If the module to be replaced is not "RMCE", a module from the list of possible modules is randomly selected as the replacement. If 'IPCV' was the selected module and it is not "RMCE", it might be replaced by 'AR' for example, if 'AR' exists in the list of possible modules.

Finally, the function returns the intake after mutation. In this case, the second intake of the first course would change from ['FL', 'IPCV', 'DL'] to ['FL', 'AR', 'DL'].

This mutation operation provides an additional level of diversity to the population of chromosomes, allowing the genetic algorithm to explore different potential solutions in the search space.

3.6 Fitness Function

In the scheduling problem at hand, the objective is to construct an optimal schedule that balances cost, student progression, and course requirements. The measure of optimality in this context is given by a fitness function. The larger the fitness function, the better the schedule. The fitness function uses both penalties and rewards to encourage or discourage certain scheduling configurations.

The hard constraints of the scheduling problem are:

1. `Penalty_students_no_module`: This penalty is levied when a student doesn't have any modules to take in a given intake. The goal is to ensure that all students have courses to take in every semester.
2. `Penalty_consecutive_intakes`: In the process of crossover and mutation, there is a possibility that the same module is scheduled in two consecutive intakes. This situation is undesirable and hence, a penalty is applied.
3. `Penalty_module_cost_penalty`: Each module incurs a cost of 5000 to teach. Therefore, the more unique modules offered, the higher the cost. This penalty is used to balance the number of modules offered and the total teaching cost.

The fitness function also encourages certain configurations with rewards:

1. `Shared_module_reward`: There are certain modules that are shared across courses. Scheduling these shared modules in the same intake across all courses can lead to significant cost savings. Hence, the algorithm is rewarded for such arrangements.
2. `First_semester_reward`: There is a preference for scheduling modules that don't require much prior knowledge in the first semester. The algorithm is rewarded for scheduling these modules in the first semester, as it aids in the smooth progression of students.

There are also soft constraints that guide the scheduling but aren't strictly enforced. They merely influence the reward/penalty structure of the fitness function and hence, the overall scheduling.

The final fitness value is computed using a weighted sum of the rewards and penalties. Each reward or penalty is multiplied by a weight that reflects its relative importance. The formula is as follows:

$$\text{Fitness_value} = (\text{shared_modules_reward} * w1) + (\text{first_semester_reward} * w2) - (\text{penalty_students_no_module} * w3) - (\text{penalty_consecutive_intakes} * w4) - (\text{module_cost_penalty} * w5)$$

In this formula, w1 through w5 are weights that reflect the relative importance of each factor. For instance, with the weights w1=5, w2=1, w3=10, w4=20, w5=5, the fitness value computation will be as follows:

$$\text{Fitness_value} = (\text{shared_modules_reward} * 5) + \text{first_semester_reward} - (\text{penalty_students_no_module} * 10) - (\text{penalty_consecutive_intakes} * 20) - (\text{module_cost_penalty} * 5)$$

Once again, the aim is to find a schedule (chromosome) that maximizes the fitness function. This schedule would represent an optimal solution to the course scheduling problem.

The fitness function works by assigning penalties and rewards to different aspects of the schedule. It assesses the schedule's fitness based on several constraints, which are mentioned previously.

Firstly, the function defines the modules and electives specific to each course and the modules shared by multiple courses. It then calculates a penalty based on the total number of unique modules in each intake across all courses in the chromosome. The more unique modules, the higher the cost, as each unique module has a cost of \$5000 to teach.

The function then goes through each student's data and checks the modules they have already taken. Based on this, the function identifies the modules that the student needs to take in the current intake. There is a penalty imposed for students who does not have a module to take, driving the algorithm towards optimization.

In addition, if the student has already taken the 'RMCE' module and doesn't have any new modules to take in the current intake, they are allowed to skip the intake without incurring a penalty. Should the same scenario recur in subsequent intakes, a penalty is implemented to discourage such absences.

There are also specific conditions for students in the 'MSc in Data Science and Business Analytics' course. If all the modules they are to take in an intake are from a different pathway

(i.e., not in their chosen pathway), a penalty is applied. If the student has already taken the maximum number of electives, and the intake only offers elective subjects, a discouragement penalty is enforced.

Another penalty is introduced for instances of the same module being scheduled in consecutive intakes, as it may lead to resource inefficiency. In contrast, the function also awards a reward for scheduling modules that are shared among pairs of courses within the same intake. This strategy promotes efficient use of teaching resources as a shared module can concurrently educate students from multiple courses.

Additionally, there is a reward system favoring the scheduling of modules that don't require much prior knowledge in the first semester. This encourages a smoother academic progression for students.

The fitness function operates within a framework of hard and soft constraints that guide the scheduling. While the hard constraints are strictly enforced, the soft constraints influence the reward/penalty structure of the fitness function without binding enforcement.

Upon the calculation of these rewards and penalties, the function returns the derived values. The overarching aim is to identify a schedule or chromosome that maximizes the rewards and minimizes the penalties, thereby achieving the highest possible fitness value. Such a schedule embodies the optimal solution to the course scheduling problem.

3.7 Experiment Setup

This section delves into the specific configurations employed in each experimental setup. Given the stable nature of our experiment design, with the random seed already set, the results will be consistent across trials. Hence, it is sufficient to conduct a single run for each experiment.

Experiment 1 (Baseline Experiment)

Parameters	Setup (value)
Population Size (Number of Chromosomes)	50
Number of Generations (iterations), N	100
Crossover Probability	0.9
Mutation Probability	0.5
Selection Method	Tournament
Crossover Method	Two Point
Mutation Method	Custom (Changes a module in a random intake to another module that is not already in the same intake).

Table 1. Configuration of the baseline experiment.

The outcomes derived from this experimental phase will establish a foundational baseline for subsequent trials.

Experiment 2 (Random Search)

The objective of this experiment is to identify an optimal parameter configuration through the use of a random search method. The parameters subject to evaluation are detailed in the table that follows:

Parameters	Setup (value)
Population Size (Number of Chromosomes)	50-250
Number of Generations (iterations), N	200-1000
Crossover Probability	0.5-1.0
Mutation Probability	0.5-1.0
Selection Method	Tournament
Crossover Method	Two Point

Mutation Method	Custom (Changes a module in a random intake to another module that is not already in the same intake).
-----------------	--

Table 2. Configuration of the second experiment, which includes the ranges of values to be searched for each parameter.

The optimal parameter value found in this experiment will be used in the third experiment.

Experiment 3 (Searching for best selection and crossover method)

Utilizing the findings from the second experiment, this experiment aims to find the best selection and crossover method. This is done by conducting a grid search on different pairs of selection and crossover method. There a 2-selection method and 2 crossover method to be tested, resulting in 4 trails to be tested, which can be represented in the table below:

Selection Method	Crossover Method
Tournament	One Point
Tournament	Two Point
Roulette	One Point
Roulette	Two Point

Table 3. Pairs of selection method and crossover method to be tested.

4 Results and Discussion

This section provides a comprehensive analysis of the experimental outcomes described in the prior section. These include evaluations of the baseline setup, determining optimal parameters through random search, and identifying the most effective selection and crossover methods.

4.1 Baseline setup

The results of the baseline setup are detailed as below:

	Values
Best Fitness Score	706
Average Fitness Score	-12672.6

Table 4. Results of the baseline experiments.

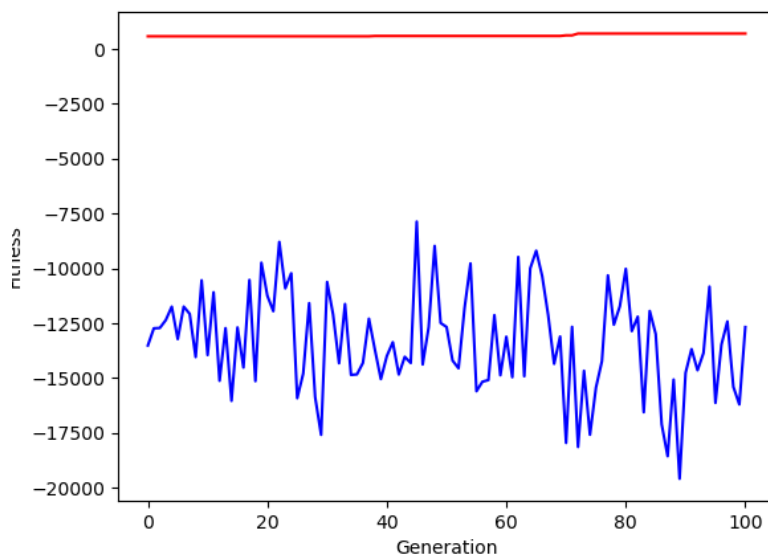


Figure 7. The blue line indicates the average fitness value of each generation whereas the red line indicates the max fitness value of each generation.

```

Shared modules reward: 2
First semester reward: 996
Penalty for students not taking any module: 6
Penalty for the same module in consecutive intakes: 0
Module cost penalty: 48
Total cost: 240000
AI
Intake 1: ['RMCE', 'NLP', 'BIS', 'CIO', 'AML']
Intake 2: ['IPCV', 'AR']
Intake 3: ['RMCE', 'DL', 'FL']
Intake 4: ['NLP', 'AI', 'CIO', 'PR']
Intake 5: ['RMCE', 'IPCV', 'AR', 'ESKE']
DSBA
Intake 1: ['RMCE', 'TSF', 'MMDA', 'SEM', 'DM']
Intake 2: ['BIA', 'DPM']
Intake 3: ['RMCE', 'ABAV', 'DM', 'AML']
Intake 4: ['ORO', 'DAP']
Intake 5: ['RMCE', 'BIS', 'ABAV']
SE
Intake 1: ['RMCE', 'RELM', 'OOSSE']
Intake 2: ['SQE', 'IA', 'NDP', 'SECT', 'SESE']
Intake 3: ['RMCE', 'DM', 'RELM', 'OOSSE', 'MSDP']
Intake 4: ['SECT', 'IA', 'NDP', 'NLP', 'SQE']
Intake 5: ['RMCE', 'DM', 'SESE', 'OOSSE']

```

Figure 8. Output of the baseline algorithm, which includes the number of violations of each constraints and the generated schedule.

The results obtained from the baseline setup indicate moderate success, despite some violations of the hard constraints. Examining the performance graph, it's clear that the maximum fitness (represented by the red line) remains nearly constant across all generations. This suggests that the best solution in the population doesn't see much improvement over time. Conversely, the average fitness (depicted by the blue line) exhibits high fluctuation, implying a significant diversity in fitness values within the population over generations. These observations hint at the possibility of a local optimum solution dominating the population early, preventing the exploration of potentially better solutions.

4.2 Random Search

10 iterations of random search are carried out and the results are detailed below:

Iteration s	Population Size	Number of Generations	Crossover Probability	Mutation Probability	Best Fitness Value
1	200	400	0.6	0.8	955
2	150	1000	0.9	1.0	834
3	200	200	1.0	1.0	789
4	150	600	0.5	0.9	903
5	100	400	0.9	0.5	960
6	50	1000	1.0	0.7	950
7	200	400	0.8	0.6	990
8	200	800	0.8	0.6	791
9	250	1000	1.0	1.0	896
10	100	1000	0.5	0.8	965

Table 5. Results of 10 iterations of random search on multiple parameters.

The random search approach yields insightful results. Among the various iterations, iteration 7 presents the highest fitness value of 990. This superior performance may stem from the balanced combination of parameters: a population size of 200, 400 generations, crossover probability of 0.8, and mutation probability of 0.6.

Interestingly, an increase in population size or number of generations does not always correlate with improved results. For example, iterations with larger populations and higher generations such as 2 and 9, do not necessarily outperform others. This observation underscores the phenomenon of diminishing returns, suggesting that beyond a certain point, the benefits of increased computational resources taper off.

The table below indicates the optimal setup from iteration 7:

	Values
Population size	200
Number of Generations	400
Crossover Probability	0.8
Mutation Probability	0.6

Table 6. Optimal configuration of parameters from the second experiment.

This particular configuration allows a balance between exploration and exploitation, enabling the genetic algorithm to effectively traverse the search space and converge towards a highly fit solution.

4.3 Selection Method and Crossover Method

Selection Method	Crossover Method	Best Fitness Value
Tournament	One Point	978
Tournament	Two Point	990
Roulette	One Point	652
Roulette	Two Point	670

Table 7. Results of the grid search on different selection and crossover method.

The results obtained from the exploration of different combinations of selection and crossover methods provide valuable insights into the optimization of the problem-solving process.

With regards to the tournament selection method, both one-point and two-point crossover techniques resulted in relatively high fitness values, 978 and 990 respectively. The tournament selection method is known for its robustness and ability to preserve diversity in the population, which can lead to improved exploration of the search space. The higher fitness value with two-point crossover suggests that this method may allow for better exploration of the solution space and potentially a more optimal combination of genetic material.

On the other hand, the roulette selection method, combined with either one-point or two-point crossover, produced noticeably lower fitness values, 652 and 670 respectively. The roulette selection method's probabilistic nature, which tends to favor individuals with higher fitness, could potentially lead to premature convergence and reduced population diversity.

Consequently, the search process might get stuck in local optima, resulting in lower overall fitness values.

In conclusion, based on these results, the combination of tournament selection and two-point crossover appears to be the most promising configuration for this problem. It seems to strike an effective balance between preserving diversity in the population and enabling advantageous genetic combinations, leading to higher fitness values. However, it's important to note that the efficiency of these methods may be problem-dependent, and further experimentation might be required for different problem instances.

4.4 Final Configuration

Following the comprehensive set of experiments, the most effective configuration is determined and shown in the table below:

Parameters	Setup (value)
Population Size (Number of Chromosomes)	200
Number of Generations (iterations), N	400
Crossover Probability	0.8
Mutation Probability	0.6
Selection Method	Tournament
Crossover Method	Two Point
Mutation Method	Custom (Changes a module in a random intake to another module that is not already in the same intake).

Table 8. Final Configuration of the best performing genetic algorithm.

```
Shared modules reward: 11
First semester reward: 1105
Penalty for students not taking any module: 0
Penalty for the same module in consecutive intakes: 0
Module cost penalty: 34
Total cost: 170000

AI
Intake 1: ['RMCE', 'CIO', 'IPCV']
Intake 2: ['AML', 'DL', 'BIS', 'FL', 'AR']
Intake 3: ['RMCE', 'AI', 'NLP', 'IPCV']
Intake 4: ['AML', 'DL', 'PR', 'CIO']
Intake 5: ['RMCE', 'FL', 'BIS']

DSBA
Intake 1: ['RMCE', 'DM']
Intake 2: ['DL', 'ORO', 'BDAT', 'BIS', 'AML']
Intake 3: ['RMCE', 'DM']
Intake 4: ['SEM', 'AML', 'DL']
Intake 5: ['RMCE', 'BIS', 'MDA', 'DPM', 'DM']

SE
Intake 1: ['RMCE', 'SESE', 'RELM', 'SQE', 'DM']
Intake 2: ['BDAT', 'SECT']
Intake 3: ['RMCE', 'DM', 'NLP', 'SQE']
Intake 4: ['RELM', 'SECT']
Intake 5: ['RMCE', 'DM']
```

Figure 9. Output of the final algorithm, which includes the number of violations of each constraints and the generated schedule.

4.5 Comparison of Baseline Algorithm and Final Algorithm

Algorithm	Baseline	Final
Fitness	706	990
Shared Module Reward	2	11
First Semester Reward	996	1105
Penalty for students not taking any module	6	0
Penalty for consecutive modules	0	0
Module Cost Penalty	48	34
Final Cost	240000	170000

Table 9. Comparison table of the baseline algorithm and the final algorithm in terms of fitness and various constraints.

In light of the comparative analysis presented above, several key observations can be made. The algorithm's performance witnessed a significant improvement from the baseline configuration to the final configuration. These changes are a testament to the effectiveness of the parameter tuning and optimization methods employed.

The most notable improvement can be seen in the fitness score, which grew from an initial 706 to 990 in the final configuration. This drastic increase underscores the efficacy of the final configuration, indicating a better compliance with both the hard and soft constraints in the class scheduling problem. Consequently, this ensures a more feasible and well-rounded schedule that adequately meets the demands and requirements of the educational institution and the students alike.

The shared module reward also witnessed a substantial increase, from a reward of 2 in the baseline to 11 in the final configuration. This improvement indicates a better organization of shared modules across different courses, enhancing the efficiency and interdisciplinarity of the class schedule. As a result, the cost can be significantly lowered.

Further improvements are evident in the penalties accrued in the final configuration as opposed to the baseline. In particular, the penalty for students not taking any module fell to 0 from the baseline's 6, reflecting an optimal distribution of modules that ensures every student is enrolled in at least one module. Similarly, the algorithm successfully eliminated the penalty for

scheduling the same module in consecutive intakes, demonstrating its adeptness at scheduling modules in a way that prevents such overlaps.

Moreover, the module cost penalty was slightly reduced, reflecting a better utilization of resources and more economical allocation of classes. The total cost saw a drastic decrease, from 240,000 in the baseline configuration to 170,000 in the final, indicating a more cost-efficient schedule that simultaneously optimizes learning outcomes and resource usage.

In summary, the final configuration of the algorithm exhibits an exceptional improvement over the baseline across all measured metrics. It not only increases the fitness of the solutions but also ensures better student engagement, improved academic collaboration, and more economical resource allocation. Therefore, the experiment successfully demonstrates the application of genetic algorithms for effective and efficient academic scheduling.

5 Limitation and Future Works

In examining the conducted study, it is important to recognize its limitations. The current implementation is based on a simplistic representation of the academic scheduling problem, and while it proves effective for the given setup, it may not be entirely applicable to more complex situations. Real-world class scheduling often involves an intricate interplay of various constraints and requirements, including individual preferences of teachers and students, availability of classrooms, and even scheduling of non-academic activities. The current model does not account for these factors, thereby limiting its applicability in practical scenarios.

Additionally, the project focused solely on the best fitness value as a performance comparison metric, neglecting others like average and minimum fitness values across the population. This could potentially mask the full spectrum of the algorithm's performance and the overall fitness landscape.

The selection of genetic operators and parameters was somewhat arbitrary, subject to a degree of trial and error. While the final configuration yielded promising results, it is possible that further optimizations could lead to even better performance. Moreover, the genetic algorithm approach, although powerful, can potentially fall into the trap of premature convergence or become stuck in local optima, meaning that there could be other, more optimal solutions that the algorithm failed to explore.

As for future work, several enhancements can be envisioned. Firstly, the model could be refined to account for more realistic constraints, thereby improving its applicability and practical relevance. This might involve incorporating faculty preferences, classroom availabilities, non-academic activities, and perhaps even real-time adjustments in response to unforeseen events.

Secondly, a more comprehensive evaluation could be implemented that includes average and minimum fitness values alongside the best, to provide a more holistic understanding of the algorithm's performance.

Lastly, more systematic and rigorous methods could be employed to optimize the genetic operators and parameters. For instance, techniques like grid search, simulated annealing, or particle swarm optimization could be explored to fine-tune these elements more effectively.

On the other hand, alternative or complementary algorithmic approaches could be investigated. While the genetic algorithm has proven its effectiveness, hybrid models that combine the strengths of different algorithms could potentially yield superior results. For instance, a hybrid model incorporating genetic algorithms and machine learning methods might allow for better generalization capabilities and predictive performance.

In conclusion, while the conducted study has achieved promising results, it also opens up several avenues for further investigation and improvement. By addressing the mentioned limitations and exploring the suggested future works, it is hoped that this line of research can lead to even more effective and practical solutions for academic scheduling.

6 Conclusion

The study presented has offered valuable insights into the application of genetic algorithms to solve a complex, real-world problem of academic scheduling. Through a series of methodical experiments and careful fine-tuning of parameters, the study successfully demonstrated the potential of genetic algorithms to generate high-quality solutions to this problem.

The initial baseline setup of the algorithm, while not perfect, served as a steppingstone for further exploration. Subsequent iterations of the algorithm, guided by a methodical parameter search, demonstrated a significant improvement in the solution quality. Furthermore, the experiment comparing different selection and crossover methods underscored the importance of these operators in the effectiveness of a genetic algorithm.

The final solution achieved a considerable improvement in the objective function, balancing between the hard and soft constraints while minimizing cost. However, this does not conclude the exploration of the problem. The study also identified several limitations and future research directions, such as the incorporation of more realistic constraints, optimization of the algorithm's parameters, and investigation into alternative or hybrid algorithmic approaches.

In essence, this research has provided a robust foundation for future explorations in this area. By building upon the findings and directions outlined in this study, future work can aim to deliver even more sophisticated and effective solutions for academic scheduling, with wider applicability and practical significance.

7 References

- Akkan, C., & Gulcu, A. (2018). A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. *Computers & Operations Research, Volume 90*, 22-32. doi:<https://doi.org/10.1016/j.cor.2017.09.007>.
- Alnowaini, G., & Aljomai, A. A. (2021). Time table scheduling using Genetic Algorithms employing guided mutation. *2021 International Conference of Technology, Science and Administration (ICTSA)*, 1-6.
- Jian, N., & Ning-Ning, Y. (2013). Genetic Algorithm and Its Application in Scheduling System. *TELKOMNIKA Indonesian Journal of Electrical Engineering 11(4)*, 1934-1939.
- Rezaeipanah, A., Matoori, S.S, & Ahmadi, G. (2021). A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. *Appl Intell 51*, 467–492. doi:<https://doi.org/10.1007/s10489-020-01833-x>
- Sapru, V., Reddy, K., & B.Sivaselvan. (2010). Time table scheduling using Genetic Algorithms employing guided mutation. *2010 IEEE International Conference on Computational Intelligence and Computing Research*, 1-4. doi:10.1109/ICCIC.2010.5705788.
- Sheng Fang, Y. (2010). A Review of Tournament Selection in Genetic Programming. *Advances in Computation and Intelligence - 5th International Symposium, ISICA 2010, Wuhan, China, October 22-24, 2010.*, 181-192.