**Course Scheduling Optimisation using Computation Intelligence and Optimisation (CIO)**

**Abstract**

This paper identifies and resolves the problem of course timetabling using computational intelligence to determine the best course timetable for MSc DSBA and MSc SE in 2025. The problem of scheduling each of the academic modules is challenging because several constraints such as time slots, faculty, and resource constraints. In order to overcome these problems, we applied the Genetic Algorithm (GA) to generate an effective timetable meeting specified constraints. The methodology includes five key steps: The steps included are initialization of the population, assessment of their fitness, use of selection method, crossover method, and mutation. At first, the population was assumed to be randomly created and the fitness of a solution was determined relative to constraints. Those mechanisms of selection made sure that better solutions were selected to be reproduced. The crossover and mutation operators were used to produce varied and better offspring solutions in order to navigate the solution space efficiently.To compare the results, the algorithm was run for 100 generations. As seen in the initialization phase, all the population scores were equal to about 6272 for all the variables. In the analysis, the GA reached peak values and became stabilized at score 6195 for all populations and iterations. To specify, the following outcomes: The results show that the proposed method can provide good feasible and optimal course schedule. The findings of this study suggest that Genetic Algorithms can be used to solve a range of optimization issues as applied to the problem of academic scheduling. In specific, future work could extend this work by developing hybrid algorithms as well as by incorporating dynamic constraints.

1. **Introduction**

Course scheduling is one of the most important factors in modern and constantly developing world of higher education necessary for successful functioning of academic programs. However, when planning the organization of course timetable, course scheduling that allows for the organization of effective learning environment regarding the requirements of student and faculty members on the one hand and the effective use of available resources on the other hand it may become a challenging undertaking. Due to a wide range of classes that have cropped up over the years, differences in students and faculty preference and availability, traditional scheduling methods do not fit all the constraints optimally. This paper explores the application of computational intelligence techniques, specifically Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA), to solve the challenging problem of course scheduling for two academic programs: As indicated by two of the degrees it offers, Master of Science in Data Science and Business Analytics (MSc DSBA) and Master of Science in Software Engineering (MSc SE). This work aims to develop an approach to schedule the courses to various time slots in a way to satisfy all the constraints such as the faculty’s availability, prerequisite courses, students’ preferences, etc., being met and at the same time achieve maximum efficiency.

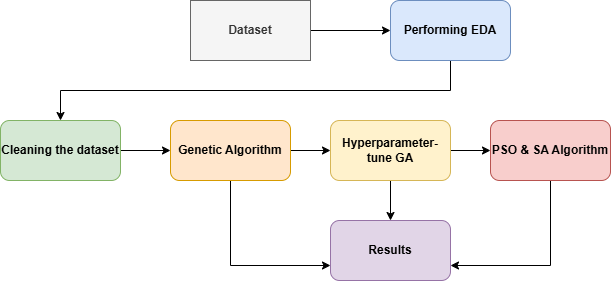
This scientific work is to provide a practical use to various educational institutions, which face this issue on a global scale, with the help of the advanced optimization techniques. In doing so, the proposed approach has the potential to present better scheduling practices regarding students and faculty that may in turn create positive carrier impact on learning at institutions.

### Literature Review

An exploration and discussion of current research in your chosen area/domain

### Methodology

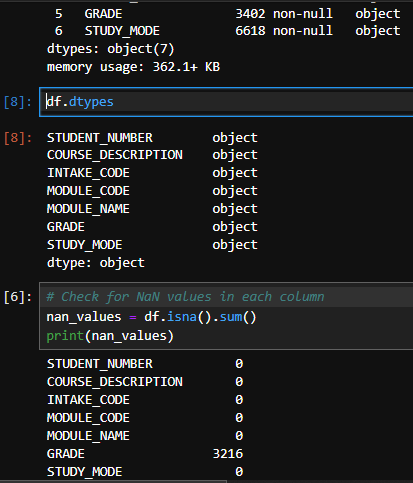
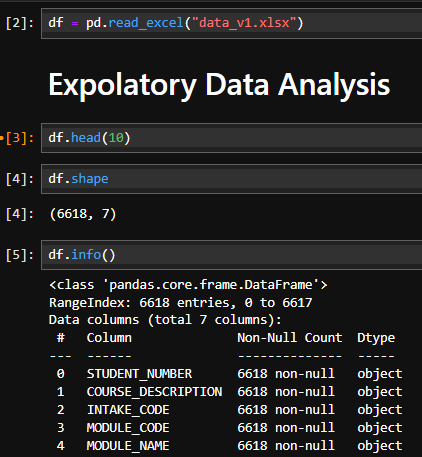
In this study, we first employed Expolatory Data Analysis(EDA) to better understand the proposed dataset, then we apply Data cleaning and three computational intelligence-based optimization algorithms to solve the course scheduling problem: Three optimal methods can be applied as follows: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA), as shown in figure 1. These algorithms have been chosen because they are good at solving general constrained nonlinear optimization problems with multiple objectives. As mentioned, all the algorithms to some extent operate trough the same principles but the difference is in their approach to the process of identifying the optimal solution to the course scheduling problem. Further in this article, you will find detailed information about each of these algorithms in terms of the required parameters and operators.



**Figure 1:** Mathedology

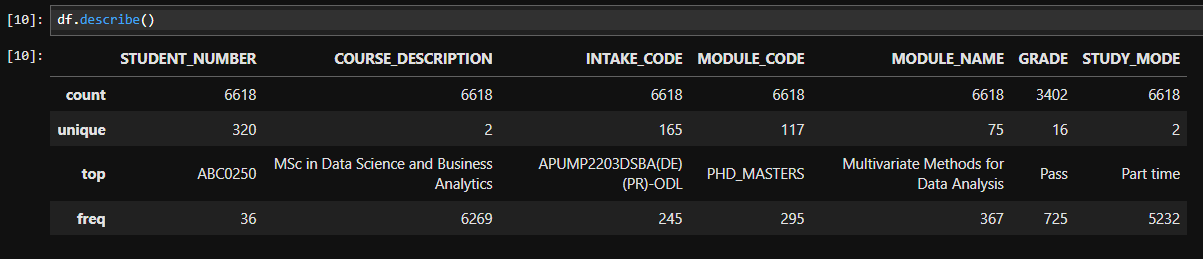
1. ***Expolatory Data Analaysis (EDA)***

Expolatory data analysis is an essentia part to conduct at the start of every artificial intelligence system. In short, it means closely looking at proposed dataset and describing it to discover its main traits, potential irregularities, and concealed patterns and connections.This insight into the proposed dataset is what will finally steer the observer through the next stages of machine learning pipeline, from data preparation to model creation and evaluation of outcomes[1]. When we first acquire a new dataset, we need to fully understand the data to manage it effectively in future machine learning tasks. Generally, we begin by describing the data in relation to the number of observations, different types of features, total missing rate, and rate of duplicate observations. With some pandas operations and an appropriate cheatsheet, we can easily display this information using a few short code snippets as shown in figure 2.



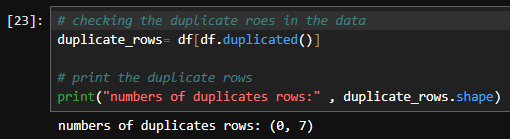
**Figure 2:** Expolatory Data Enalaysis

By implementing initiall steps of eda we come to know that our dataset have 6618 rows and 7 columns . All of our columns is objective in datatype and column “GRADE” column has 3216 null/missing values present in it. df.describe() pandas’ function is used to give more information abut the proposed dataset as shown in figure 3.



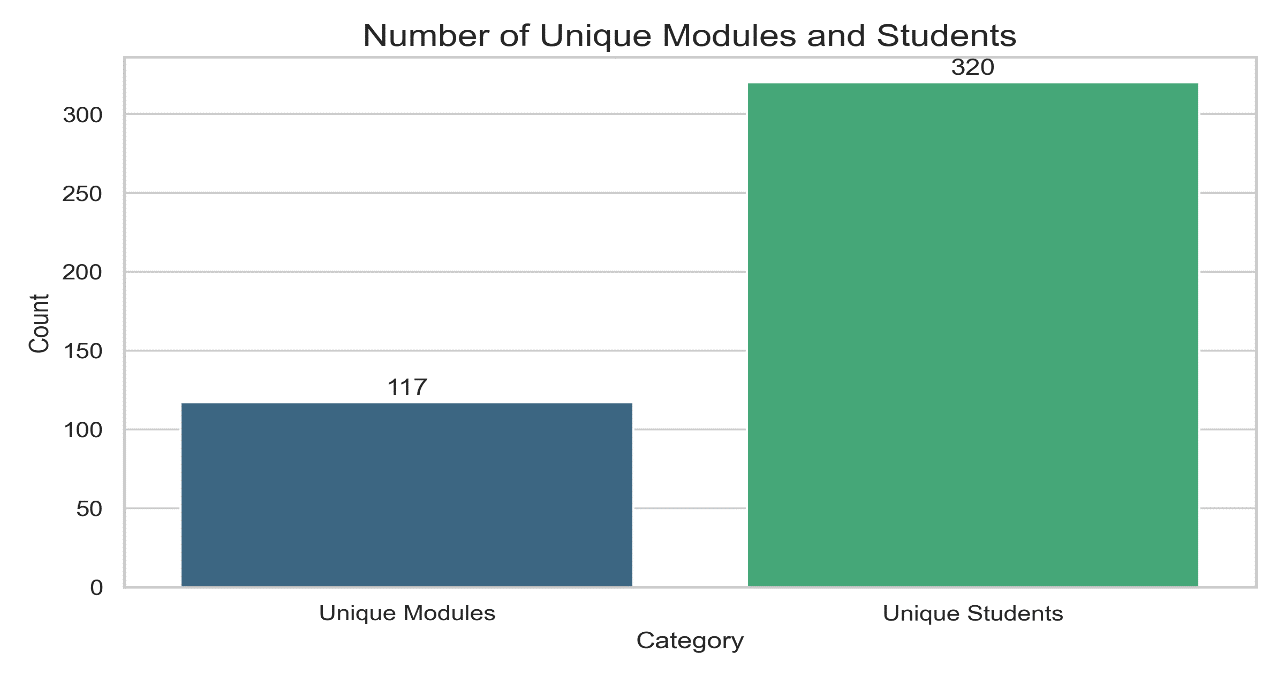
**Figure 3:** Dataset deep desription

By df.describe() function we get to know about the total counts, unique value, top value and the most frequent value present of every columns of the proposed dataset. After completeing this process the next step is to check duplicates values present in the dataset. By using ‘df.duplicated()’ function we check the duplicates values in propsoed dataset and come to know that dataset has no duplicate values present in the dataset, as shown in the figure 4.



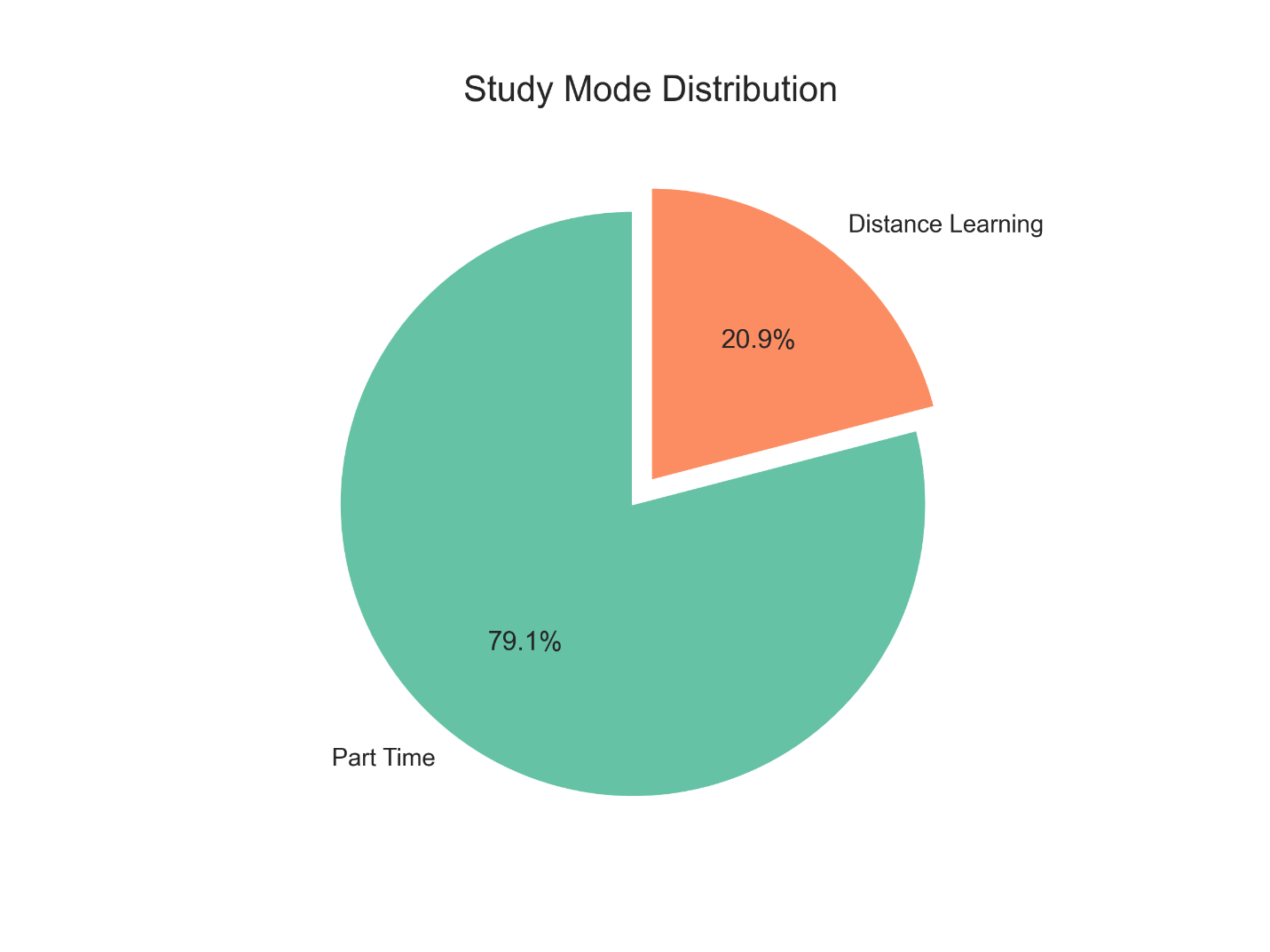
**Figure 4:** Duplicate values

After, checking the duplicates values in the dataset we then move to check unque modules and students present in the dataset.



**Figure 5:** Unique modules and student

As shown in figure 5, there are 117 unique modules and 320 unique students present in the proposed dataset. This helps us to undetsand the dataset more deeply. After checking the unique modules and students we jumped into the next step of expolatory data analysis by analyzing every columns of the proposed dataset.



**Figure 6:** Stduy Mode

As shown in the figure 6, we checked the column “Stduy\_Mode” distibution. This columns has two class “Part Time” and “Distance Learning”. 79.1% of the dataset has Part time class and rest 20.9% is Distance Learning. After checking the distibution of column “STUDY\_MODE”, we then check the disibuion of column “GRADE”.

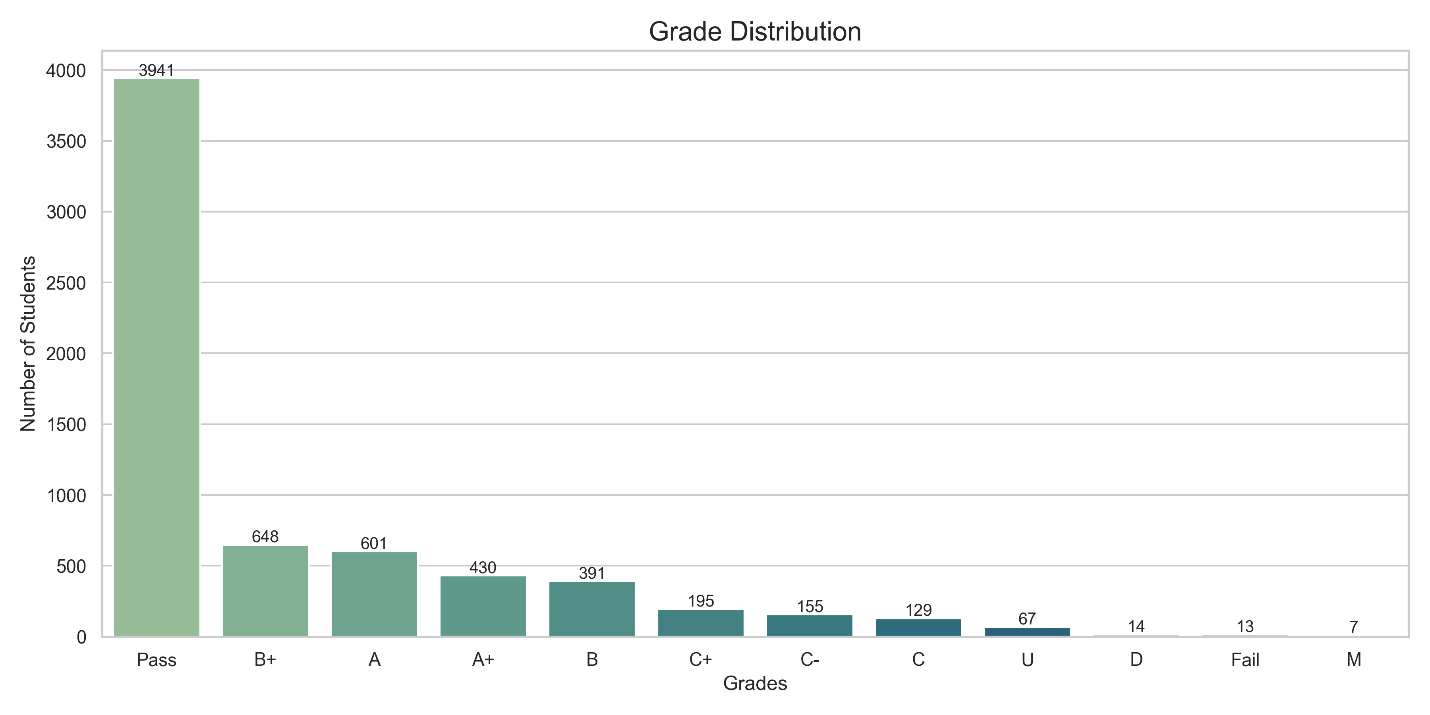


Figure 7: GRADE Distribution

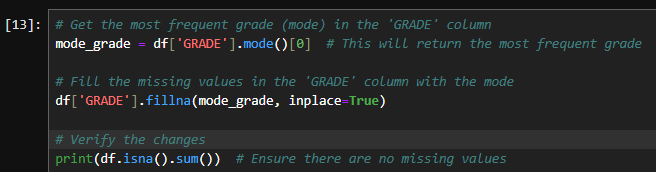
Figure 7 shows the distribution of column “Grade”. The class “pass” has the most counts 3941 and “M” has least counts which are 7.

1. ***Cleaning the Dataset***

After analyzing the dataset, we then jumped to next step to clean the dataset. Data cleaning, also called data cleansing or data scrubbing, is the process of finding and fixing or removing mistakes, inconsistencies, and inaccuracies in data sets. It is an important step in the machine learning workflow, since the quality and reliability of the data directly affect the precision and efficiency of the models created from it[2]. Data cleaning is crucial to make sure that the data is correct, uniform, and dependable, enabling machine learning models to provide precise and significant predictions. There are several reasons why data cleaning is extremely important in machine learning. First and foremost, data is frequently gathered from various sources, like databases, websites, and sensors. Each source can have its unique peculiarities and inconsistencies, resulting in differences in data formats, absent values, and repeated entries. Data cleaning works to resolve these problems, guaranteeing that the data is uniform and standardized overall[2].

Another important part of data cleaning is dealing with missing data. In real life, it is usual for data to have missing values, which can affect the accuracy and performance of machine learning models. Data cleaning methods are used to fill in missing values or remove cases with too much missing data, thus making sure that the dataset is whole and usable for analysis[2].

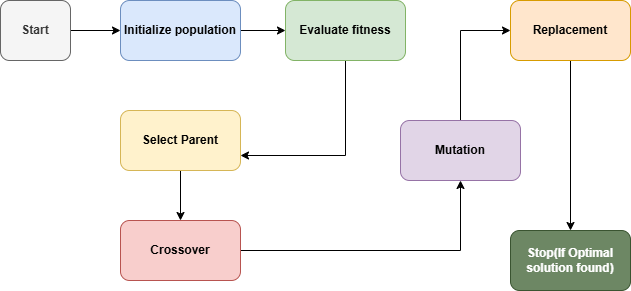
The proposed dataset is checked using function df.isna().sum() and it shows that the dataset’s column “GRADE ” has 3216 missing values present. These missing values are filled using mode as shown in figure 8.



**Figure 8:** Filling the missing values

1. ***Genetic Algorithm (GA)***

The next step in accomplishing the project after completing data cleaning phase is to construct the genetic algorithm to help minimize course scheduling conflicts. Figure 9 presents the propcess of building genetic algorithm.



**Figure 9:** Genetic Algorthm

Like other types of evolutionary algorithms, the Genetic Algorithm (GA) is an approach modeled after natural selection, from the natural world. It tends to over time enhance a set of prospective solutions in efforts to obtain the final solution. The main steps involved in GA are:

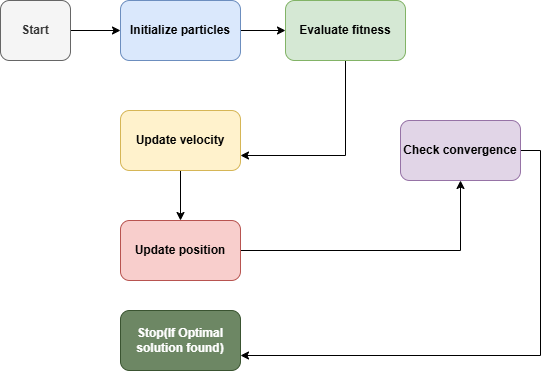
* **Initialization:** The first population of coming solutions (chromosomes) is created randomly.
* **Selection:** A fitness function gaols at evaluating the quality of a solution. The solutions are then selected based on the ability of the company to reproduce them efficiently.
* **Crossover:** What initially has been selected is mated –crossed over – so that two particular solutions appear as parents to give birth to two other solutions.
* **Mutation:** Variance is introduced into some of the offspring by random change so as to decrease the resemblance.
* **Replacement:** Selection replaces the old population with the new one and the process is continued for the fixed number of generations or until the optimum solution is achieved.

**GA Parameters:**

* **Population size:** The number of solutions available in each generation of solutions.
* **Crossover rate:** The opportunity to join together two kinds of solutions.
* **Mutation rate:** The possibility of just randomly making alteration to solutions.
* **Number of generations**: The number of cycles of passes by in order for it to complete its execution.

1. ***Particle Swarm Optimization (PSO)***

The next algoritm we build is Particle Swarm Optimization (PSO). Particle Swarm Optimization (PSO) is based on the simulation of bird flocking or fish schooling. In each solution, a “particle” is assumed which navigates through the search space. The movement of particles depends on its own IdP and the IdP of the swarm of particles in the space. The process of building Particle Swarm Optimization (PSO) is shown in figure 10.



**Figure 10:** PSO Algoithm

The key steps in PSO are:

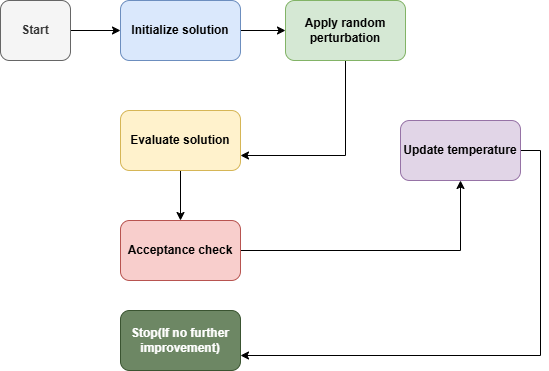
* **Initialization:** Particles are created with initial positions and velocities decided randomly from an initially given range.
* **Fitness evaluation:** The different particles fitness level is determine based on how suitable the scheduling constraints were.
* **Velocity update:** Based on the own last-known position and the last-known position of the whole Particle swarm, particles change their velocities.
* **Position update:** The particles then proceed in their new velocities.; The particles then proceed in their new velocities.
* **Convergence check:** And if the swarm converges (this means that the best solution has been identified) then the process stops.

**PSO Parameters**:

* **Population size:** The number of particles.
* **Inertia weight:** Regulates the ability of a particle’s prior velocity which affects its present velocity on the screen.
* **Personal best and global best:** The solutions which are better known for each particle of swarm and for the whole swarm.
* **Maximum velocity:** The highest velocity at which particles may travel.

1. ***Simulated Annealing (SA)***

The next algorithm which we used after the PSO model was Simulated Annealing (SA). Simulated annealing is a procedure that is inspired by the real process of annealing of metals for receiving low-energy status. I observed that it works by making random changes to the current solution and accepting those changes based on a probability that decreases with time. build the SA model process is illustrated in fig 11.



**Figure 11;** SA algorithm

The key steps in SA are:

* **Initialization:** When developing a dynamic system, one should start with an initial solution.
* **Random perturbation:** This is done with small random variation of the current solution.
* **Acceptance criterion:** Generally if the new solution is better it is accepted. If it is worse, nevertheless, it can be accepted with a certain probability.
* **Cooling schedule:** The temperature is slowly decreased and slowly enough to decrease the probability of accepting a worst solution in the subsequent iterations.
* **Stopping condition:** The algorithm ceases when the temperature has reduced low enough and so there can be no enhancement of the solution anymore.

**SA Parameters:**

* **Initial temperature:** Regulates the first probability of measuring worse solutions.
* **Cooling rate:** The rate in which temperature declines starting from normal temperature.
* **Maximum iterations:** It carries the number of iterations until which the algorithm should be run before it is stopped.

In our course scheduling problem, we applied all three algorithms to the same objective: reducing scheduling clashes, making effective course to time slot mapping and in making constraints such as faculty teaching availability, the prerequisite pattern and the student’s choices respectable. The results from all three algorithms have been compared to find out which of the three methods was the most effective in terms of computational time, and quality of the schedule that was generated.

**General Workflow**:

* **Input:** The course scheduling problem which involves different courses, the time slots that are available, the faculty availabilities, the preferred choices of students as well as the constraints.
* **Algorithm Execution:** The three algorithms of GA, PSO, and SA are initiated on the desired input data set separately.
* **Output:** Each algorithm also produces a schedule that is assessed using a predetermined fitness function.
* **Comparison:** To coordinate the activities of each shift, the schedule is compared to that of every other algorithm and decide which one gave the best results.

Table 1 shows, the summary of all algorithms used to solve the course scheduling problem.

**Table 1:** Summary of algorithms

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Parameters/Operators** | **Description** |
| Genetic Algorithm | Population size, crossover rate, mutation rate, number of generations | Evolves solutions through selection, crossover, and mutation. |
| Particle Swarm Optimization | Population size, inertia weight, personal best, global best, max velocity | Updates particle positions and velocities based on fitness. |
| Simulated Annealing | Initial temperature, cooling rate, maximum iterations | Makes random changes to solutions and accepts them based on a temperature-based probability. |

By applying these optimization algorithms, the course scheduling problem can be efficiently solved, ensuring an optimal allocation of courses while satisfying various constraints.

1. **Experiment/Implementation/Simulation**

The following section focuses on the design, implementation, and assessment of the developed computational intelligence approach used in the timetable allocation problem of MSc DSBA and SE programs. The task includes the coordination of the modules in an efficient way while following the certain rules which are part of the program , for example providing the RMCP and RMCE in the January, May, October’s semester. The models that have been incorporated are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA).

***A. Introduction***

Timetabling is perhaps one of the most important issues and challenges in delivering an academic course since it requires a number of constraints, these include; available modules, resource allocation and program guidelines. These complexities are further exacerbate in organizations with academic programs, particularly when resources and priorities of individual modules must be timely addressed. Many traditional approaches to solution development do not satisfy these requirements exhaustively, and, therefore, new more adaptive and intelligent methods are needed.

Essentially, this work uses Computational Intelligence (CI) methods to solve such problems. CI algorithms are widely applied to solving optimization problems owing to their flexibility, the capacity to work with non linear tasks, and, at the same certainty, their ability to find near optimal solutions. The experiment is centered on the ability of such algorithms to reduce the costs of scheduling, satisfy the constraints, as well as enhance the schedule for MSc DSBA and SE programmes.

The key objectives of this implementation include:

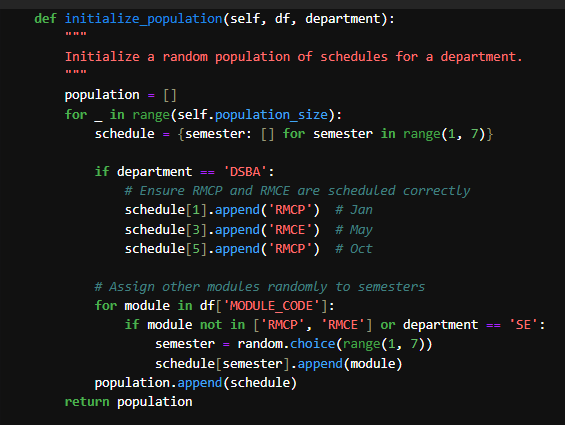
* Developing scheduling solutions to have RMCP delivered in the current semester and RMCE as the next semester offering.
* Decreasing the number of reactions by lowering the use of resources and penalties associated with scheduling.
* Comparing between GA, PSO and SA concerning their scalability as well as efficiency.

***B. Genetic Algorithm***

GA is a metaheuristic algorithm which is based on the process of one of the natural evolutions, natural selection. This involves the generation of a population of other potential solutions over several or other generations in a bid to arrive at the best or near best solution.

**Encoding Schedules**

Every candidate solution which is chromosome contains schedule for either of the two, DSBA or SE. The chromosome is implemented as a dictionary of semester numbers as keys and a list of offered module codes as values for each semester (1-6), as shown in figure 12.



**Figure 12:** Chromosomes initialization

**Fitness Function**

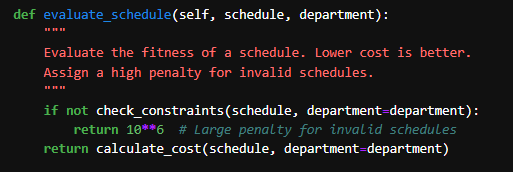
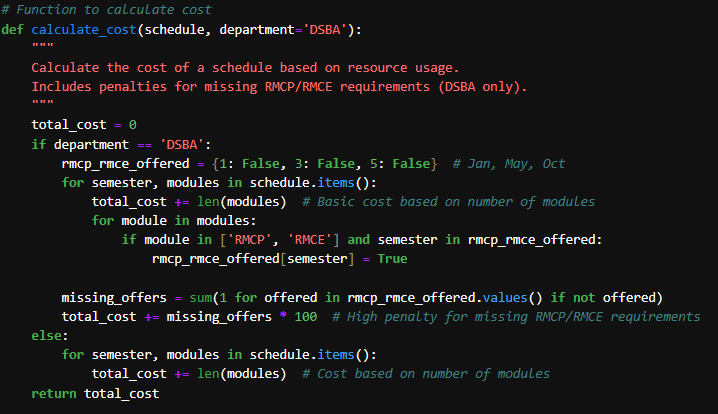
In the proposed Genetic Algorithm(GA), the ‘fitness function’ is the considered technique that serves to assess the population of schedules. In this code, it is implemented as the combination of the following two methods:

**1.evaluate\_schedule(schedule, department):**

This method calculates the fitness score of a particular schedule. It begins with checking whether the schedule is possible or not with given constraints through the check\_constraints() function. In case a schedule does not hold any constraints (is invalid), the fitness value returns to 10^6. If the schedule is valid, then it will put the cost of the schedule by the ‘calculate\_cost()’ function. Less has it, the better the fitness.

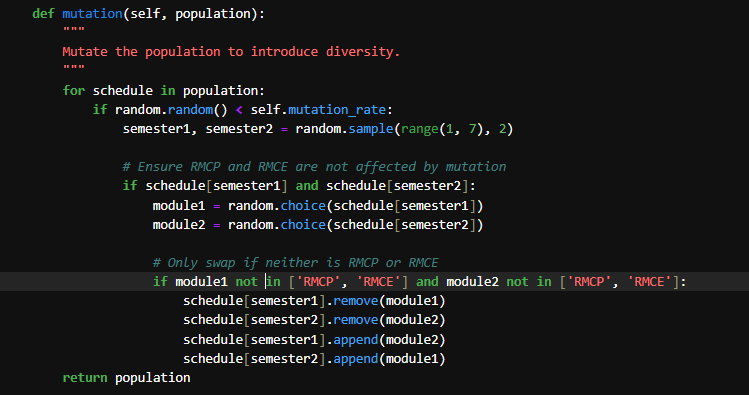
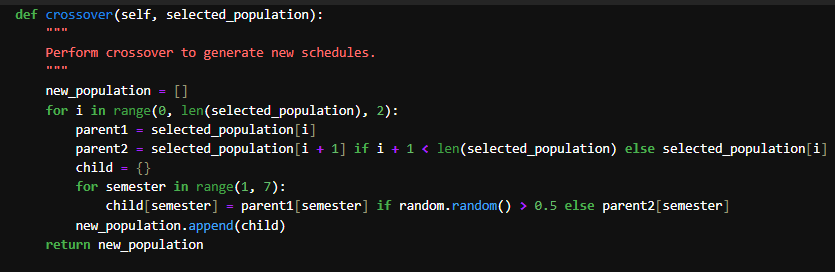
**2.calculate\_cost(schedule, department):**

This method calculates the cost of a schedule which is the fitness value for valid schedules in this work. It involves sanctions for failure to meet RMCP and RMCE (for DSBA department). It also sums up the cost according to the number of modules in a semester and the cost of the modules are added according to their credit ratings, as shown in figure 13.



**Figure 13:** Fitness function

**Genetic Operators**:  
The crossover and mutation processes ensure diversity in the population.These operators code snippets are shown in figure 14.



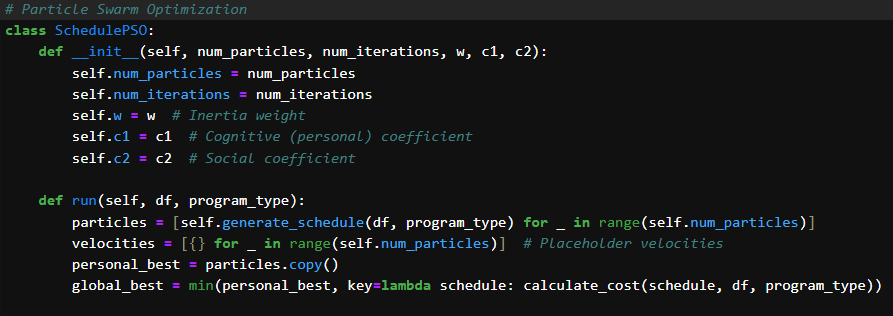
**Figure 14:** Genetic Operators

***C. Particle Swarm Optimization (PSO) Implementation***

After, Implementing Genetic Algorithm(GA), the next technique is called Particle Swarm Optimization (PSO), imitates swarm intelligence: particles move based on their best (local) solution and a best global solution. The PSO implementation uses particles (candidate schedules) and velocities to iteratively optimize the scheduling solution.

**Particle Initialization**

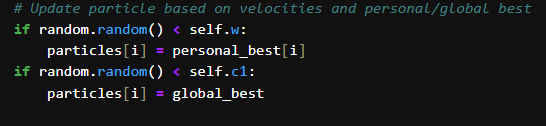
As in GA, this starts with a population of potential schedules. Other parameters that are defined for each particle include a schedule and velocity even though velocity here is not used. Particles are initialized with random schedules, similar to GA. A placeholder velocity is included for completeness, as shown in figure 15.



**Figure 15:** Particles Initialization

**Update Particles**

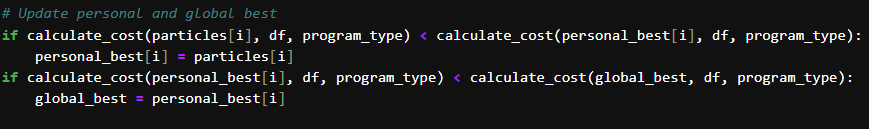
Particles move toward the global best and their personal best solutions, as shown in figure 16.



**Figure 16:** Update PSO Particles

**Update Bests**

Evaluate and update the personal best and global best solutions, as shown in figure 17.



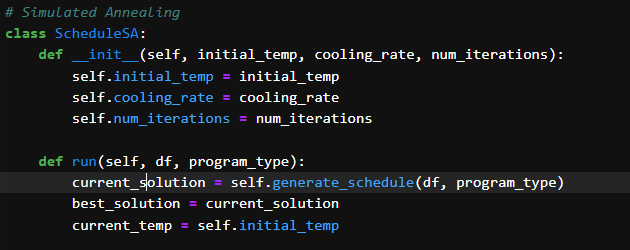
**Figure 17:** Update PSO Bests

***D. Simulated Annealing (SA)***

After implementing PSO technique the next step is to implemnt Simulated Anealing(SA). Simulated Anealing(SA) increases the feasible region by jumping from a current solution to another solution which is randomly generated from the current one. It sometimes takes worse solutions just to avoid getting trapped in a local optimum with the probability of acceptance reducing over time.

**Initialization**

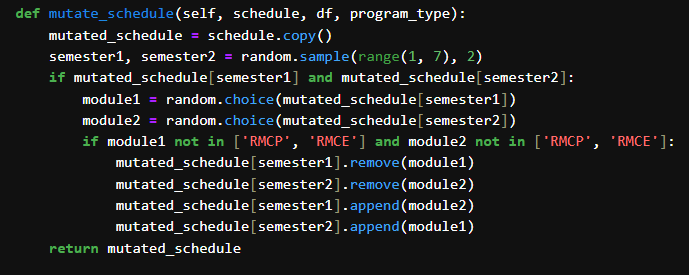
Simulated Anealing(SA) start with an initial solution and temperature as shown in figure 18.



**Figure 18:** SA Initialization

**Mutation**

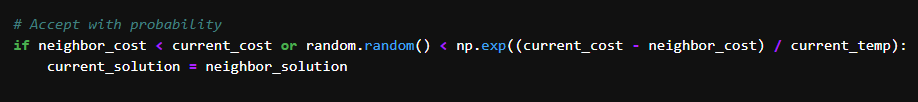
Mutate the current solution to generate a neighbor solution, as shown in figure 19.



**Figure 19:** SA Mutation

**Acceptance Probability**

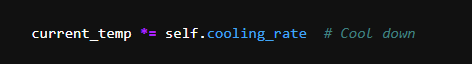
Accept worse solutions based on temperature as shown in figure 20.



**Figure 20:** Acceptence Probability

**Cooling**

Gradually lower the temperature as shown in figure 21.



**Figure 21:** SA Cooling

1. **Results, Analysis, and Discussion**

The optimization of course scheduling techniques such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Simulated Annealing (SA) for MSc DSBA and MSc SE programs helps to find the best solution to the problem with limited number of scheduling conflicts and also makes the best use of time slots and resources as far as possible. The outcomes have been evaluated according to the convergence behaviors, optimum solution or cost values obtained, and total efficiency over various trials.

***A. Genetic Algorithm***

**DSBA Results**

Table 2 provides the fitness value obtained as a result of the run of the implemented Genetic Algorithm for 100 generationsfor DSBA As it could be expected, the algorithm showed stable enhancement of fitness at the beginning of the iterations but after some generations its performance deteriorated.

**Table 2:** Genetic Algorithm Performance for DSBA Result

|  |  |  |  |
| --- | --- | --- | --- |
| **Generation** | **Best Fitness(DSBA)** | **Avg Fitness(DSBA)** | **Worst Fitness(DSBA)** |
| 1 | 6272 | 6272.00 | 6272 |
| 10 | 6168 | 6272.48 | 6386 |
| 20 | 6167 | 6232.12 | 6314 |
| 30 | 6190 | 6213.64 | 6262 |
| 50 | 6190 | 6194.60 | 6195 |
| 51 | 6195 | 6195.00 | 6195 |
| 70 | 6195 | 6195.00 | 6195 |
| 90 | 6195 | 6195.00 | 6195 |
| 100 | 6195 | 6195.00 | 6195 |

First time, Best, Average and Worst value of generation is uniform because in first generation, population contains similar or same solutions so algorithm performance is same. For more than 100 generations, the scores of Best, Average, and Worst decrease and after the 114th generation they converge to 6195, meaning that the result achieved is viable in the whole population and the result is also an optimal one for this problem.

**SE Results**

Table 3 shows a visible operation of the Genetic Algorithm for the schedule-establishment scheduling problem across a hundred generations involves an enhancement in terms of, and an approach towards the optimal solution according to the objective function. More specifically, the best fitness value becomes much higher initially, ranging from 349 in Gener1 to the best level of 309 through Gen16. After that, the algorithm becomes steady, with small increases and slight oscillations in mean fitness. From the Generation 92 onwards all the best, average and worst fitness values tend to bepopulating the value 326 which shows that the algorithm has got saturated and the best possible solution is being derived. Steady convergence is shown here to be due to the ability of the GA to optimize the schedule, while at the same time retaining high levels of diversification in the early iterations and high levels of exploitation in the later iterations.

**Table 3:** Genetic Algorithm Performance for SE Result

|  |  |  |  |
| --- | --- | --- | --- |
| **Generation** | **Best Fitness** | **Average Fitness** | **Worst Fitness** |
| 1 | 349 | 349.00 | 349 |
| 10 | 317 | 362.84 | 387 |
| 20 | 309 | 329.20 | 371 |
| 30 | 309 | 326.60 | 370 |
| 50 | 326 | 335.84 | 350 |
| 70 | 326 | 330.80 | 350 |
| 90 | 326 | 327.92 | 350 |
| 100 | 326 | 326.00 | 326 |

***B. Genetic Algorithm with PSO and SA***

**Genetic Algorithm (GA)**

Table 4 below gives the fitness values of the configuration through the genes configured by the Genetic Algorithm for 50 generation. The observed behaviour of the algorithm of the aim was where it improved the fitness throughout the initial generations but failed to do so as the generations increased.

**Table 4:** GA Performance

|  |  |
| --- | --- |
| **Generation** | **Best Fitness** |
| 1 | 6621 |
| 10 | 6398 |
| 20 | 6436 |
| 30 | 6538 |
| 40 | 6538 |
| 50 | 6538 |

**Particle Swarm Optimization (PSO)**

The performance of PSO is given in Table 5 below. The algorithm did not see any enhancements in the best cost value over 50 iterations, and the cost value was constant at 6621.

Table 5: PSO Performance

|  |  |
| --- | --- |
| **Iteration** | **Best Cost** |
| 1 | 6621 |
| 10 | 6621 |
| 20 | 6621 |
| 30 | 6621 |
| 50 | 6621 |

**Simulated Annealing (SA)**

Table 6 describes the results for the Simulated Annealing method. As is the case with PSO, the algorithm also did not arrive at improved cost values and was stagnated at 6621 for 100 iterations.

Table 6: SA Performnace

|  |  |
| --- | --- |
| **Iteration** | **Best Cost** |
| 1 | 6621 |
| 25 | 6621 |
| 50 | 6621 |
| 75 | 6621 |
| 100 | 6621 |

***C. Analysis and Comparison of Optimization Methods***

**Table 4:** Comparative Analysis of Optimization Methods

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Best Fitness/Cost** | **Convergence** | **Performance Summary** |
| Genetic Algorithm (DSBA) | 6195 | Moderate | Effective in reducing fitness over multiple generations but stagnated near the optimal solution. |
| Genetic Algorithm (SE) | 326 | Moderate | Demonstrated steady improvement and eventual convergence to a stable solution. |
| Particle Swarm Optimization | 6621 | None | Failed to optimize beyond the initial value, indicating poor adaptation to the problem constraints. |
| Simulated Annealing | 6621 | None | No improvement observed over iterations, suggesting ineffective exploration. |

Table 7 shows that the GA for DSBA achieved a minimum fitness of **6195** over multiple generations. The algorithm showed consistent improvement early on but stagnated, potentially converging to a local minimum. Despite the stagnation, its performance was effective compared to other methods. Similarly, the GA for SE reduced the fitness value from an initial high to a final 326, stabilizing over the last few generations. This demonstrates GA's capability for steady optimization and convergence, reflecting its reliability for scheduling tasks. PSO's inability to optimize beyond the initial fitness value of **6621** highlights poor adaptability to the problem's constraints. This might result from inadequate hyperparameter tuning or the swarm's lack of diversity. SA's failure to improve upon the initial solution also points to insufficient exploration, likely caused by an ineffective cooling schedule or a high dependency on the initial configuration.

In summary, Genetic Algorithm(GA) outperformed PSO and SA for both DSBA and SE problems, showcasing its robustness in optimizing complex scheduling tasks. While GA demonstrated moderate convergence and effectiveness, PSO and SA struggled, underscoring the importance of hyperparameter tuning and problem-specific customization in optimization techniques.

***D. Discussion***

These results reveal that all the three computational intelligence techniques have distinct performance differencing; between MSc DSBA and MSc SE course timetable. The GA was found to be the most efficient, having an early generation non-overlap for both problems when the best fitness values are considered. However, using GA, fitness value was obtained at a value of 6195 for DSBA and 326 for SE. These results portray it as a flexible and resilient scheduling model suited for multiple scheduling complications. However, in both cases, the fact the performance of GA stops increasing at some point shows that GA might need further fine tuning or the use of a different kind of metaheuristic to overcome local optima.

On the other hand, none of PSO and SA had any improvement, having a fitness value of 6621 at zero iterations. This lack of improvement can be attributed to the fact that the parameter settings have not been optimized, or that the search capabilities are limited in most cases when used on extremely constrained problems such as course scheduling. It may be seen from these outcomes that PSO and SA may have to undergo considerable adaptation, including better ways of initializing particles or information, or changing into adaptive methods, to deliver satisfactorily in such circumstances.

1. **Conclusion and Future research**

The GA that was proposed was able to outperform the other proposed algorithms when optimizing the course schedules of the MSc DSBA and the MSc SE, with fitness values of 6195 and 326 respectively. However, as marked by the stagnation after 30 generations, there remains much scope for optimisation of this convergence. On the other hand, PSO and SA do not enhance cost values, which indicates a low suitability under most conditions or require extensive modifications of algorithms or problems.

**Future research**

* Since GA outperformed other methods for both DSBA and SE, it could be enhanced with additional heuristics, such as elitism or adaptive mutation rates, to achieve better convergence. Moreover, incorporating problem-specific knowledge, such as soft constraints prioritization, could further optimize results.
* Introduce adaptive penalty mechanisms to handle hard and soft constraints dynamically during optimization. This can help guide the algorithms to explore feasible solutions more effectively while balancing trade-offs between conflicting objectives in course scheduling.
* Test the optimization methods on larger, real-world datasets with more courses, instructors, and constraints to evaluate their scalability and generalizability. Developing a framework that integrates real-time feedback from stakeholders (e.g., students and faculty) can ensure practical applicability and improve satisfaction with the schedules.

1. **References**

[1] Santos, M. (2023, May 30). A data scientist's essential guide to exploratory data analysis. Towards Data Science. Retrieved December 8, 2024, from <https://towardsdatascience.com/a-data-scientists-essential-guide-to-exploratory-data-analysis-25637eee0cf6>

[2]Collins, N. (2023, November 17). What is data cleaning in machine learning? Robots.net. Retrieved December 8, 2024, from <https://robots.net/fintech/what-is-data-cleaning-in-machine-learning/>