Application of Genetic Algorithms: a Case Study on Course Scheduling

Fu Jiang[1], Jianping Shuai[2], Yi Nong[3] and Yuan Gao[4,*]

[1]GuangXi Vocational College of Safety Engineering, Nanning, 530200, China
[2]Guilin University of Electronic Technology, Nanning, 530200, China
[3]Guangxi Medical University, Nanning, 530200, China
[4]Guangxi Vocational and Technical University of Agriculture, University East Road, No.176, 530200, Nanning, China

*m13677715851@163.com

Abstract. Addressing the complexity of university course scheduling problems, this study develops an intelligent scheduling model based on Genetic Algorithm (GA). By analyzing multi-constraint characteristics in course arrangement, we propose a multi-dimensional evaluation strategy integrating period preference, weekly combination optimization, feasibility degree, and satisfaction optimization criteria. Within the GA framework, a chromosome encoding scheme tailored for scheduling requirements is designed, accompanied by an optimization mechanism incorporating roulette wheel selection, multi-point crossover, and dynamic mutation operators. Systematic parameter configuration is implemented for population initialization and iteration strategies. Experimental results demonstrate that the proposed scheduling system effectively balances teaching resource constraints and optimization objectives, achieving global optimization in course arrangement. The operational outcomes validate the feasibility and effectiveness of GA in resolving multi-constrained scheduling problems, providing technical reference for the informatization of academic management in higher education institutions. This research offers a practical solution framework for curriculum scheduling optimization while expanding the application scope of evolutionary computation in educational administration systems.

Keywords: Genetic Algorithms; A Case Study on Course Scheduling; Application

1 Introduction

The Genetic Algorithm (GA) [1] is a global stochastic optimization search algorithm designed by drawing inspiration from the genetic evolution laws in nature [2]. During the search process, the genetic algorithm can automatically acquire and accumulate knowledge about the search space. Under the action of selection, crossover, and mutation operators, individuals evolve towards a direction of better fitness, with the aim of finding the optimal solution to a problem.Course scheduling is a core aspect of teaching

H. Kannan et al. (eds.), *Proceedings of the 5th International Conference on Internet, Education and Information Technology (IEIT 2025)*, Atlantis Highlights in Social Sciences, Education and Humanities 40,
https://doi.org/10.2991/978-94-6463-803-5_47

management in higher education institutions and is highly complex. It requires consideration of time conflicts between courses, as well as the allocation of teachers and classrooms. Moreover, it is essential to ensure teaching quality by maximizing the optimality of class periods and week combinations. As the number of courses increases, the potential number of scheduling combinations grows exponentially, leading to an extremely large search space for the problem.

Reviewing the relevant researches, scholars have attempted to apply various methods to solve the course scheduling problem. Traditional rule-based approaches, easy as they are to implement, often fail to find the global optimal solution. Heuristic algorithms like simulated annealing and tabu search enhance solution quality to some extent, but their performance is constrained by algorithm parameter settings and initial solution selection. Recently, some studies have explored using intelligent optimization algorithms such as particle swarm optimization and ant colony optimization to address this problem. These algorithms demonstrate certain advantages in search efficiency and solution quality.

Existing research faces numerous challenges. On the one hand, accurately quantifying diverse constraints and optimization objectives in course scheduling, such as session optimality, weekly combination optimality, feasibility, and satisfaction optimality, remains an urgent issue to be resolved. On the other hand, swiftly identifying high-quality solutions within the vast search space is also a current research difficulty. To address these problems, this paper takes the common course scheduling problem in higher education institutions as an example. It innovatively combines strategies for session optimality, weekly combination optimality, feasibility, and satisfaction optimality, and employs genetic algorithms to design a course scheduling algorithm for higher education. Through simulating natural selection and genetic mechanisms, genetic algorithms can automatically acquire and accumulate knowledge about the search space during the search process, thereby promising to locate more superior solutions. This study aims to assist in the teaching management of higher education institutions, enhance scheduling efficiency, and improve the quality of generated timetables.

2 Concepts Related to Genetic Algorithms

2.1 Algorithm Overview

Genetic algorithms were first proposed by John Holland in the 1970s [3]. These algorithms simulate the mechanisms of natural selection and genetics, using computer simulations to transform the process of problem-solving into processes similar to those of chromosomal gene crossover and mutation in biological evolution [4]. Genetic algorithms search for optimal solutions by simulating natural evolution through basic operations such as selection, crossover, and mutation. They have been widely applied in various fields, including the positioning and routing problems of autonomous vehicles [5], combinatorial optimization, machine learning, signal processing, adaptive control, and artificial life.

2.2 Algorithmic Procedure

The application of genetic algorithms to solve problems is an iterative process that continuously searches the solution space [6]. It begins with a population composed of a certain number of individuals, where each individual represents a solution to the problem. The fitness value of each individual in the population is then calculated, and it is determined whether the termination condition of the algorithm is met. If the condition is satisfied, the best individual, which is the optimal solution, is outputted, and the genetic algorithm search is terminated. If the termination condition is not met, individuals are selected, crossed, and mutated according to certain strategies, resulting in a new generation of offspring. The fitness values of each individual in the offspring population are then recalculated, and this process is repeated iteratively. The population evolves over generations to produce increasingly better approximate solutions to the problem until the termination condition of the algorithm is met. The specific general steps of a genetic algorithm are as follows:

Step 1: Encode chromosome genes to create individuals;

Step 2: Initialize the population;

Step 3: Calculate the fitness value of each individual in the population;

Step 4: Determine whether the termination condition of the algorithm is met. If the termination condition is met, proceed to Step 9;

Step 5: Select individuals that will enter the next generation according to a certain selection rule;

Step 6: Select individuals from the population for crossover operations according to a certain selection rule;

Step 7: Select individuals from the population for mutation operations according to a certain selection rule;

Step 8: Return to Step 3;

Step 9: Output a satisfactory solution or an optimal solution to the problem.

The evolutionary flowchart of the genetic algorithm is shown in Figure 1.

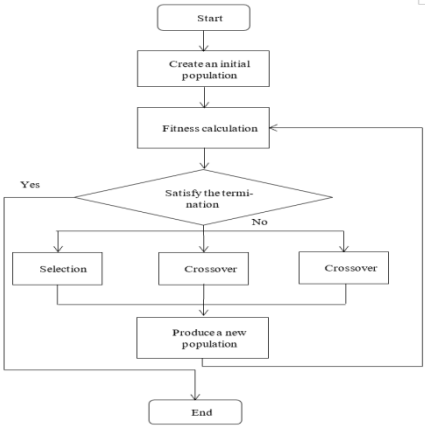

Fig. 1. The evolutionary flowchart of the genetic algorithm.

2.3 Algorithmic Characteristics

Unlike traditional optimization algorithms, genetic algorithms have the following characteristics:

Parallelism. Compared to traditional search algorithms that start from a single point, genetic algorithms exhibit parallel search through a set of points [7]. When the required solution precision is not high, genetic algorithms can solve problems more quickly. The parallelism of genetic algorithms is primarily reflected in their implicit parallelism. Since genetic algorithms search using a population-based approach, they can simultaneously explore multiple regions of the solution space.

Problem independence. Genetic algorithms do not require knowledge of the problem itself. They establish a connection with specific problems through a fitness function, without the need for additional auxiliary knowledge. The fitness function serves as the sole basis for judgment. The search process involves applying genetic operations to the encoded individuals in the population, rather than directly acting on the specific variables of the optimization problem. When dealing with similar problems, simply replacing the fitness function is sufficient. Moreover, genetic algorithms have a global optimization capability, resulting in higher search efficiency. They begin the search in the solution space with a set of possible solutions, covering a wide range, which leads to more efficient exploration.

3 Design of a Course Scheduling Algorithm based on Genetic Algorithms

The objective of this paper is to implement a course scheduling system that combines algorithmic scheduling with manual adjustments. After the algorithm is executed, the scheduling system generates a course schedule based on the data produced by the algorithm. Generally, the automatically generated schedule by the algorithm may have some shortcomings. In such cases, manual adjustments can be made to the generated schedule. Subsequently, the selected schedule is used by the system to create timetables for each teacher and class [8].

Utilizing genetic algorithms for course scheduling can enhance the efficiency and quality of the timetable, addressing the issues of errors that often arise in manual scheduling. This approach also frees educational administrators from the burdensome task of course scheduling. By using a scheduling system to generate timetables, the management of scheduling information resources can be informatized, and the content of the timetable can be conveniently and quickly visualized. The main tasks involve managing data related to scheduling resources, performing scheduling calculations, and processing the results of the scheduling. The primary workflow is as follows:

(1)Information maintenance involves the statistical entry and management of data required by academic administrators for scheduling courses. The primary data involved includes classroom data, teacher data, and data related to classes and students that need to be scheduled. The main operations include adding, modifying, and deleting data.

(2)Academic administrators develop relevant teaching plans and tasks based on the requirements and regulations of the school for the current semester.

(3)The scheduling and adjustment of courses are completed to generate a course scheduling plan.

(4)Based on the generated course scheduling plan, the scheduling system produces corresponding class schedules, classroom schedules, and teacher schedules, among other information.

First, consider the classroom schedule as a two-dimensional table, and then fill in the courses according to certain rules. This can be regarded as a combination C_m^n, where m represents the total number of periods for all laboratory courses, which is determined to be m=25, based on the actual situation of classroom management at the author's workplace. The number of combinations for scheduling laboratory courses for some values of n is shown in Table 1.

Table 1. Combinations of Classroom Scheduling for *m*=25.

n	1	2	3	4
C_m^n	25	300	2300	12560

3.1 Course Code

To design and implement a course scheduling management system using genetic algorithms, it is essential to select an appropriate encoding scheme. For course timetables, employing a Boolean matrix for encoding is a suitable choice. In the Boolean matrix, the horizontal axis represents periods, and the vertical axis represents days of the week. The element aij in the matrix indicates the i th period on day j, where aijcan take values of 0 or 1. A value of 0 signifies that no course is scheduled, while a value of 1 indicates that a course is scheduled

3.2 Determine the Fitness Function

In the process of solving using genetic algorithms, the quality of chromosomes determines whether they are eliminated. The quality of chromosomes is judged through a fitness function. In the context of course scheduling, determining the fitness function requires not only considering the basic and optimization constraints of the courses being scheduled, but also taking into account the occupation of resources such as teachers. Consequently, the fitness function of the scheduling algorithm should possess adaptability. The following discussion elaborates on the parameters involved in determining the fitness function.

Class period suitability weight. Class period suitability reflects the varying time - based requirements for classes across different periods. Given that teaching effectiveness is better when teachers conduct classes during periods with high suitability, class period suitability should be assigned a large weight in the fitness function. The specific weight value can be determined based on the results of a survey on teachers' preferences for class periods. As mentioned earlier, there are a total of m (m=25) periods in a week.

Since teachers haveted to obtain the Class Period Suitability Table, as s varying requirements for the timing of each period, this results in the suitability of each period. For teachers, periods with higher suitability tend to yield better teaching outcomes. A sample survey of teachers was conduchown in Table 2. The data in Table 2 demonstrate the higher the value of the suitability, the greater the quality.

Table 2. Class Period Suitability.

Period	Monday	Tuesday	Wednesday	Thursday	Friday
1st & 2nd	0.90	0.97	0.98	0.93	0.91
3rd & 4th	0.89	0.89	0.91	0.84	0.82
5th & 6th	0.76	0.81	0.75	0.77	0.70
7th & 8th	0.59	0.68	0.65	0.62	0.61
9th & 10th	0.33	0.34	0.39	0.32	0.20

Weight of the Optimality of Weekly Schedule Combinations. The optimality of weekly schedule combinations ensures that courses are evenly distributed throughout the week, thereby enhancing the teaching effectiveness. This parameter is also crucial in the fitness function. However, its weight should be slightly lower than that of the optimality of class periods. This is because the optimization of weekly schedule combinations represents a further adjustment based on the optimality of class periods. To ensure the effectiveness of teaching, the scheduling of classes should be as evenly distributed throughout the week as possible. Therefore, the class scheduling strategy must take into full consideration the quality of the combination of different days within a week.

Table 3 presents the weekly schedule combination quality for a course that is arranged to have two class periods within a week in a selection of classrooms. If a course is scheduled for two periods on Monday at the same time, then the priority value is 0.04. The priority values for other times are shown in Table 2-3.

Table 3. Frequency Combination Goodness Table.

Day/Combination	Priority Value	Day Combination	Priority
Mon&One	0.04	Tue&Five	0.76
Mon&Two	0.09	Wed&Three	0.04
Mon&Three	1.00	Wed&Four	0.12
Mon&Four	0.59	Wed&Five	1.00
Mon&Five	0.40	Thur&Four	0.05
Tue&Two	0.05	Thur&Five	0.11
Tue&Three	0.10	Fri&Five	1.00
Tue&Four	1.00		

Feasibility. Feasibility is a key indicator for measuring whether a course scheduling plan is viable. It reflects the conflict situation in the allocation of teacher and classroom resources. In the fitness function, feasibility should account for a certain weight to ensure that the course scheduling plans generated by the algorithm are of practical significance. However, since feasibility is a fundamental requirement for a course scheduling plan, its weight should not be set too high, so as to avoid overshadowing the importance of other optimization objectives. The measurement of class scheduling feasibility refers to whether there are conflicts in the allocation of teachers and classrooms. A "0" is used to indicate conflicts in the scheduling allocation of teachers and classrooms, while a "1" indicates no conflicts.

Degree of Satisfaction. The "degree of satisfaction" refers to the extent to which special requirements for the scheduling of some teachers and courses can be met. Degree of Satisfaction of Optimization refers to the extent to which the special requirements for the course scheduling of some teachers and courses can be met. The weight of this parameter in the fitness function should be determined according to the actual needs. If the school has specific optimization requirements for course scheduling, such as reducing the number of consecutive classes or optimizing the scheduling time for specific teachers, the weight of the degree of satisfaction of optimization should be increased accordingly.The value of the "degree of satisfaction" ranges from 0 to 1, and if all requirements can be met, the value is 1.

In summary, when selecting the weights, methods such as expert scoring, Analytic Hierarchy Process (AHP), or entropy weight method can be adopted for quantification. Additionally, in order to further enhance the accuracy and robustness of the fitness function, it is advisable to consider introducing weight optimization methods. For instance, genetic algorithms can be utilized to search for and optimize the weights, aiming to find the optimal combination of weights. Although this approach increases the complexity of the algorithm, it is expected to improve the overall quality of the course scheduling scheme. Therefore, based on the analysis of the four parameters of class period suitability, weekly combination suitability, class period suitability, and combination suitability, the fitness function f for the classroom scheduling algorithm is given as:

$$f = \underline{(\lambda_1 \times p_1 + \lambda_2 \times p_2)} p_3 \times p_4 \tag{1}$$

where p_1 is the class period suitability, p2 is the weekly combination suitability, λ_1 and λ_2 represent the weights of class period suitability and combination suitability, respectively, $\lambda_1+\lambda_2$, p_3 is the feasibility, and p_4 is the degree of satisfaction.

3.3 Genetic Operations

Selection operation. This paper adopts the "roulette selection" method, and the specific implementation steps are as follows:

Step 1: Sum up the fitness function values of all courses to obtain *fTotal*;

Step 2: Generate a random number r within the range of 0 to *fTotal*;

Step 3: *choice*=0, *cTotal*=0, *i*=0, where *choice* represents the course selected by roulette selection, *cTotal* represents the accumulated fitness function value, *i* represents the fitness function value, *f*(*i*) represents the fitness function value of the *i* th course calculated according to formula (2-1), and *i*=0 is to start the accumulation process from the first course;

Step 4: *cTotal=cTotal+f(i)*;

Step 5: If *cTotal>r*, then let *choice=i*, end the selection process, otherwise *i=i*+1, and continue to execute Step 4.

Crossover operation. The crossover operation combines the excellent genes from the original varieties to produce a new generation. By continuously mating, a better species can be obtained. For the course encoding represented by a Boolean matrix, it is necessary to keep the number of "1" s constant during the crossover process, which reflects the weekly class periods of a course. The crossover operation uses two courses as parent courses (courses obtained after the selection operation) to exchange part of the codes in the two courses to produce new courses. For the course encoding represented by a Boolean matrix, the crossover will be achieved by exchanging the nth "1" bit of the parent courses, where n is randomly generated. Assuming n=2, when two parent courses perform crossover, it is to recombine from the second "1" bit of course 1 with the last "1" bit of course 2 to generate new courses.

Mutation operation. The mutation process involves the offspring produced after mating, and mutations are carried out according to the preset mutation probability. For binary bit strings, the mutation method is to change the 0s in the string to 1s and the 1s to 0s. The mutation process can perform mutation operations on a single bit or the entire string. For the "mutation operation" of the course scheduling algorithm, it mainly involves changing the binary encoding of the corresponding course, turning the corresponding 0s to 1s and 1s to 0s.

Termination condition of genetic algorithm. The termination condition of the genetic algorithm for course scheduling adopted in this paper is to determine whether the fitness target has been reached or whether the algorithm has iterated a certain number of times.

3.4 Course Scheduling Algorithm

Algorithm 3.1 describes the course scheduling algorithm designed using the genetic algorithm (the algorithm is briefly referred to as JSJLCPK).

Algorithm 3.1 JSJLCPK

Input: *Courses List*

Output: Recommended course set

Begin

1: *generation*←0; // Initialize the generation number to zero

2: *P*(*t*)←initialize(*generation*) ; // Encode the current generation population, including course information, classrooms, and time, etc.

3: $f(P(t)) \leftarrow \{f(), f() \ ,\ldots, f()\}$; // Calculate the fitness (expected value) of course i

4: evaluate (*P*(*t*)); // Measure the performance of each class in the original timetable

5: *R*0←rand (1); *R*1←rand(1); *R*2←rand(1); *t*←0.8; *p*←0.02;

// Randomly generate R0 for the selection function, R1 for the crossover rate, and R2 for the mutation rate, assign initial values to the crossover rate parameter t, and mutation probability p

6: fConflict($P(t)$); // Detect and correct conflicts in courses and classrooms

7: while (generation) do

8: begin

9: generation←generation+1; // Increment the generation number by one

10: select($f(P(t))$); // Select individuals with strong adaptability from the current generation population (current optimal courses)

11: begin

12: $P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$; (2)//Use the fitness f(x) to calculate the probability of a certain encoded course being inherited to the next population during the update iteration process

13: $q_i = \sum_{i=1}^{N} \mathrm{P}(x_\mathrm{j})$ (3) // After each course has been selected, crossed, and mutated, the probability of excellent courses is continuously updated and accumulated

14: if ($R0<q_i$) then $\mathrm{P}(_i)$; // If the randomly generated value is lower than this probability, then select this course

15: end

16: if ($R1<t$) then

17: $PC(t)$←crossover($P(t)$); // Randomly select two timetables and pair them according to the principle to generate a new timetable

18: if ($R2<$p) then

19: PM(t)←mutate($PC(t)$); // Randomly mutate the selected timetable according to the principle

20: fConflict($P(t)$); // Detect and correct conflicts in courses and classrooms

21: evaluate ($PM(t)$); // Measure the performance of each class in the (mutated) new timetable

22: P(t+1)←elitist ($PM(t) \cup P(t)$); // Retain the optimal courses of the previous generation and the new generation

23: f(P(t))V{f($^{x_1(t)}$), f() ,…, f()}; // Calculate the fitness (expected value) of the new timetable

24: t←t+1; // Update the generation number

25: if (*generation*>=1000 or $f(P(t))$ meets the threshold) then break;

26: end while

27: *timetables*←Allocate ($P(t)$); // Allocate classrooms and time according to the chromosome (course) and output:

28: return *timetables*←transform($P(t)$) ; // The course encoded into the course

29: End.

3.5 Algorithm Analysis

Let the number of class periods be n, the total number of periods in the timetable be m, and k represent the maximum number of iterations of the genetic algorithm:

Statements (1) and (2) take O (1) time, statement (3) takes O(n) time, statement (4) takes O(m) time, statement (5) takes O (1) time, and statement (6) takes O(m) time;

Statements (7) to (26) are a loop that can execute up to k rounds, and the complexity of each round of the loop body is analyzed below:

Statements (8), (9), (11), (14), (15), (16), (17), (18), (19), (22), (23), (24), (25), and (26) take O(1) time, statements (10), (12), (13), and (23) take O(n) time, and statements (20) and (21) take O(m) time. The rest of the statements in the loop body take O (1) time, so the total time consumed by the loop body is O(k(n+m)),

Statements (27) and (28) take O(m) time;

Therefore, the total time required is: O(k(n+m)).

4 Conclusion

This paper uses the genetic algorithm to solve the class scheduling problem, combining the strategies of class period suitability, weekly combination suitability, feasibility, and degree of satisfaction. Based on the framework and steps of the genetic algorithm, the encoding, initialization of the population, and design of genetic operators are completed according to the specific characteristics of the class scheduling problem. Then, a detailed design of the class scheduling system is carried out, proving that the class scheduling system can effectively complete class scheduling and use.

Specifically, during the search process, the algorithm automatically acquires and accumulates knowledge about the search space, and continuously evolves towards a higher-quality fitness level through operations such as selection, crossover, and mutation. In the design of the fitness function, this paper comprehensively considers four parameters: the optimality of class periods, the optimality of weekly schedule combinations, feasibility, and optimization satisfaction degree, and determines the weight of each parameter according to the actual needs. The experimental results show that the course scheduling schemes generated by the algorithm perform excellently in terms of the optimality of class periods, the optimality of weekly schedule combinations, and feasibility.

Compared with other course scheduling methods, the genetic algorithm-based course scheduling method proposed in this paper has the following advantages: Firstly, it can automatically search for the global optimal solution, avoiding the defect of traditional methods that are prone to falling into local optimality. Secondly, it can adapt to various constraints and optimization objectives, improving the flexibility and practicality of the course scheduling scheme. Thirdly, it has a relatively fast search speed and convergence performance, and can generate high-quality course scheduling schemes within a relatively short period.

However, there are still some deficiencies in this study. For example, in the selection of weights in the fitness function, although methods such as expert scoring are used for quantification, there is still a certain degree of subjectivity. Future research can consider introducing more objective and scientific methods for determining weights, such as data-driven weight optimization methods, to further improve the accuracy and robustness of the algorithm. In addition, it is also possible to explore the combination of the

genetic algorithm with other intelligent optimization algorithms to form a more efficient and stable course scheduling algorithm.

Acknowledgement

This study is supported by the Guangxi science and technology planning project in 2023 (AD23026247).

References

1. Bagley, J. D. The behavior of adaptive systems which employ genetic and correlation algorithms: technical report (1967).
2. Gao, Y. Design and implementation of a comprehensive management system for computer public laboratories. Guangxi University (2021).
3. Huang M., Wang L.B., Xiao M.H., Fu Y., Zuo Z.K. An efficient genetic algorithm based on adaptive boundary constraints. Journal of Peking University (Natural Science Edition), 64: 665-672 (2024).
4. Jiao, W. G., Ding F.G., Shi J.H. A clustering routing algorithm based on improved genetic algorithm. Journal of Beijing University of Posts and Telecommunications, 46: 83-88 (2023).
5. John H. H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, Control and Artificial Intelligence. MIT Press, Cambridge (1992).
6. Li, E., & Zhang, Z.X. Improved genetic algorithm for searching optimal vehicle routing under dynamic orders. Computer Engineering and Applications, 60: 353-364 (2024).
7. Liu, Q.X, Wang, T. T., Wang, Y.N. Solving vehicle routing problem with non-dominated sorting particle swarm genetic algorithm. Journal of Jilin University (Engineering Edition), 1-10 (2024).
8. Ma, B., Hu, D. and Wu, X. The Location Routing Problem of the Car-Sharing System with Autonomous Electric Vehicles. KSCE Journal of Civil Engineering, 25: 3107–3120 (2021).