algorithms

MDPI

Article

Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming

Xu Han and Dian Wang *

School of Technology, Beijing Forestry University, Beijing 100083, China; xuhan99@bjfu.edu.cn
* Correspondence: wangdian@bjfu.edu.cn

Abstract: The university course scheduling problem (UCSP) is a challenging combinatorial optimization problem that requires optimization of the quality of the schedule and resource utilization while meeting multiple constraints involving courses, teachers, students, and classrooms. Although various algorithms have been applied to solve the UCSP, most of the existing methods are limited to scheduling independent courses, neglecting the impact of joint courses on the overall scheduling results. To address this limitation, this paper proposed an innovative mixed-integer linear programming model capable of handling the complex constraints of both joint and independent courses simultaneously. To improve the computational efficiency and solution quality, a hybrid method combining a genetic algorithm and dynamic programming, named POGA-DP, was designed. Compared to the traditional algorithms, POGA-DP introduced exchange operations based on a judgment mechanism and mutation operations with a forced repair mechanism to effectively avoid local optima. Additionally, by incorporating a greedy algorithm for classroom allocation, the utilization of classroom resources was further enhanced. To verify the performance of the new method, this study not only tested it on real UCSP instances at Beijing Forestry University but also conducted comparative experiments with several classic algorithms, including a traditional GA, Ant Colony Optimization (ACO), the Producer–Scrounger Method (PSM), and particle swarm optimization (PSO). The results showed that POGA-DP improved the scheduling quality by 46.99% compared to that of the traditional GA and reduced classroom usage by up to 29.27%. Furthermore, POGA-DP increased the classroom utilization by 0.989% compared to that with the traditional GA and demonstrated an outstanding performance in solving joint course scheduling problems. This study also analyzed the stability of the scheduling results, revealing that POGA-DP maintained a high level of consistency in scheduling across adjacent weeks, proving its feasibility and stability in practical applications. In conclusion, POGA-DP outperformed the existing algorithms in the UCSP, making it particularly suitable for efficient scheduling under complex constraints.

Keywords: genetic algorithms; university course scheduling problem; dynamic programming; judgment mechanism; forced mutation operation

Academic Editor: Francisco Saldanha da Gama

Received: 13 February 2025
Revised: 3 March 2025
Accepted: 5 March 2025
Published: 10 March 2025

Citation: Han, X.; Wang, D. Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming. *Algorithms* **2025**, *18*, 158. https://doi.org/10.3390/a18030158

1. Introduction

The university course scheduling problem (UCSP) is a complex optimization issue in the field of education, aiming to reasonably allocate course times and teaching resources to students and faculty [1]. The UCSP is classified as a typical NP-complete problem [2], meaning that finding the optimal solution requires significant computational resources. The constraint set of the UCSP is usually divided into hard constraints and soft constraints. Hard constraints are conditions that must be strictly followed during the scheduling

process, and violating these constraints renders the schedule infeasible. These include the school's curriculum planning policies, the equipment and facilities of the available classrooms, and the availability of teachers and students. Soft constraints, on the other hand, are conditions that are preferable to satisfy but can be violated at the cost of reducing the quality of the schedule. These include teacher preferences for timetable arrangements, student expectations for offered courses, and the unique management requirements of each institution [3]. Therefore, creating a satisfactory schedule is not a simple task, as it requires balancing multiple constraints.

In recent years, many scholars have used heuristic algorithms to solve scheduling problems [4,5], including the UCSP. Some commonly used heuristic methods include particle swarm optimization (PSO) [6–8], Simulated Annealing (SA) [9–11], and Tabu Search (TS) [12–15]. Among these, Hossain et al. [16] designed PSOSS and introduced a forced exchange operation, which significantly optimized PSO. Shiau [17] incorporates a local search mechanism into PSO, which enhances the solution refinement, but the method heavily depends on the quality of the initial population and lacks robustness in highly constrained scheduling environments. Ayob and Jaradat [18] propose a hybrid ant colony system (ACS-SA) that combines Simulated Annealing and particle swarm optimization. While this hybrid approach enhances the exploration and exploitation, SA is highly sensitive to parameters such as the initial temperature and cooling rate. Improper parameter settings may result in slow convergence or premature convergence to a local optimum, reducing the scheduling efficiency. Hiryanto [19] applies DCM and Vertex Graph Coloring (VGC) to significantly improve the computational efficiency. However, this method primarily focuses on satisfying hard constraints and has a limited ability to handle soft constraints, making it less suitable for scheduling scenarios that require greater flexibility. Genetic algorithms also demonstrate significant effectiveness in solving scheduling problems. Wen et al. [20] propose a QIGA based on Q-learning, and Squires et al. [21] develop LSWT-GA. Xiang, Hu, Yu, and Wang [22] propose a heuristic algorithm for the UCSP that combines a GA, a GCP, and enhanced Tabu Search.

Some researchers have used exact algorithms by formulating the UCSP as a linear programming problem and solving it using integer linear programming (ILP). Domenech and Lusa [23] developed a MILP model that incorporated teachers' teaching loads and preferences for different categories of teachers. Esmaeilbeigi et al. [24] propose an enhanced branch-and-check algorithm based on Benders decomposition. However, decomposition-based methods often require well-structured problem formulations, and their efficiency deteriorates in real-world scheduling scenarios with complex and dynamic constraints. Constraint-based approaches have also been explored. Some researchers have employed the epsilon-constraint method to address distributed scheduling issues [25,26], while others have integrated mixed-integer programming (MIP) with Constraint Programming (CP) in hybrid models [27–30].

Although these methods have made some progress in addressing the UCSP, limitations still remain. Most of the existing algorithms focus on theoretical optimal solutions or simplified experimental environments, neglecting the complexity and resource constraints of real-world scenarios. In particular, many algorithms only deal with independent course scheduling, overlooking the significance of joint course scheduling in practical applications [31]. Joint course scheduling involves multiple classes sharing certain courses, requiring the schedule to simultaneously satisfy the timetables of several groups, classroom capacities, and teacher availability. This makes joint course scheduling more challenging than independent course scheduling.

The current research gap lies in the inability of many of the existing algorithms to effectively address the complexity of joint course scheduling. While joint scheduling

helps improve the utilization of classroom resources, it also significantly increases the difficulty of scheduling, and the performance of the existing algorithms in this area has been unsatisfactory. Therefore, developing an algorithm that can efficiently handle both independent and joint course scheduling is of great importance.

To address this challenge, this paper proposes a progressively optimized heuristic algorithm that combines genetic algorithm (GA) and dynamic programming (DP) techniques, referred to as POGA-DP. This algorithm is not only capable of handling complex constraints within the UCSP but is also particularly suited to joint course scheduling problems. Compared to previous studies, our contributions are as follows:

(1) A progressively optimized hybrid genetic algorithm tool (POGA-DP) was developed and tested on 520 course scheduling tasks at Beijing Forestry University to improve the resource utilization and meet the practical scheduling needs. The innovation of this algorithm lies in its combination of the advantages of the GA and DP, enhancing the computational efficiency while avoiding local optima.
(2) We not only verified the effectiveness and stability of POGA-DP in complex instances but also conducted a comprehensive performance comparison with a GA, the Producer-Scavenger Method (PSM) [32], and the Particle Ant Colony Algorithm (ACO). The results demonstrated the significant advantages of POGA-DP in terms of the resource utilization and scheduling quality.
(3) This paper designed multiple fitness functions to optimize different objectives (such as th eresource utilization and the number of classrooms used), further enhancing the practicality and stability of the algorithm.
(4) To ensure the continuity of the scheduling scheme, we also analyzed the similarity of the course arrangements between adjacent weeks, verifying the stability and applicability of POGA-DP in long-term scheduling.

In terms of the other sections of this paper, Section 2 details the soft constraints, hard constraints, and specific aspects of the UCSP studied in this paper using a mixed-integer linear programming model. Section 3 provides a detailed description of the POGA-DP algorithm. Section 4 presents the results of the numerical experiments. Section 5 summarizes the contents of this paper.

2. Model Building

In this section, we provide a detailed introduction to the UCSP and develop a mixed-integer linear programming model for the problem. Section 2.1 offers a thorough summary of the soft and hard constraints faced by the UCSP. Section 2.2 describes the representation methods for the decision variables and sets, while Section 2.3 elaborates on the relevant constraint conditions.

2.1. Description of the Problem

The university course scheduling problem (UCSP) is a complex combinatorial optimization problem, especially in the case of combined courses. During the course scheduling process in higher education institutions, various factors such as the teaching plans, courses, instructors, administrative classes, classrooms, teaching classes, and time slots are involved. These factors are interconnected and constrain each other. The teaching plan [22] is primarily formulated by the school's academic dean based on the actual conditions of the school and college, including components such as the courses, instructors, administrative classes, and teaching classes. Specifically, a course for a certain administrative class taught by a particular instructor in a specific teaching class is referred to as a teaching event. The UCSP can be represented as assigning appropriate class times and classrooms to n teaching events. Since our scheduling scope was a semester, we decomposed the semester into a

set K of weekdays within a week, a number W of weeks, and a set P of time slots in a day. For demonstration purposes, we divided a day into 5 time slots and assumed that classes were held on 5 days a week, resulting in 25 time slots available for scheduling [33].

The solution to this problem is subject to both hard and soft constraints. To enhance the practicality of the scheduling, the following restrictions are considered:

Hard constraints:

(1) In the same time slot, a teacher can teach only one course;
(2) In the same time slot, a classroom can accommodate only one course;
(3) In the same time slot, a class can be scheduled for only one course;
(4) The scheduling process must meet the time requirements for each course;
(5) The classroom assigned to a course must be able to accommodate all students, especially in the case of combined classes;
(6) When a course is a combined class, multiple administrative classes need to attend the same course at the same time and in the same classroom;
(7) Special courses require assignment to specific classrooms, such as laboratories for experiments, music rooms, and chemistry labs;
(8) When the time slot of a teaching event is fixed, the time slot for that teaching class cannot be changed;
(9) The course hours need to be completed within the specified weeks.

Soft constraints:

(1) Distribute the same course evenly throughout the week;
(2) Distribute the courses for each administrative class evenly throughout the week to avoid scheduling too many courses on the same day, which could lead to an excessive burden on students;
(3) Avoid having too many teaching hours for instructors within a single day to prevent excessive concentration;
(4) Strive to accommodate the preferred teaching times of the instructors;
(5) Aim to optimize the utilization of classrooms and equipment to avoid resource wastage.

2.2. Decision Variables and Sets

Table 1 summarizes the meanings of the relevant mathematical symbols.

2.3. The Fitness Function

Due to resource limitations and diverse needs, various fitness functions with different focuses have appeared in the previous literature. Xiang, Hu, Yu, and Wang [22] designed a fitness function that measured the impact of a fast neighborhood evaluation strategy, while Thepphakorn and Pongcharoen [34] calculated the total operational cost of university course scheduling. Additionally, some related studies have focused on the dispersion of courses within the timetable [35,36]. To verify the advantages of POGA-DP over other algorithms in terms of its practicality and resource utilization, this paper referenced the fitness function from Hossain et al. [16] and designed a teacher preference value that aligned with Chinese teaching practices. The teaching preference value for each teacher in a specific time slot was represented by an integer, as shown in Table 2. Higher values indicated the stronger preference of a teacher to teach during a given time slot, while negative values represented the negative preference of the teacher. The overall preference value for the teachers was calculated using the following formula:

$$F_1 = \sum_{t \in T} (Q_t - V_t) \tag{1}$$

$$V_t = \omega_1 \sum \theta_\tau 2^{\tau-1} + \omega_2 \sum T_{max} \quad (2)$$

In Equation (2), V_t represents the penalty function for measuring teacher dissatisfaction. To accommodate the teaching preferences of different countries, we introduce the weight parameters ω_1 and ω_2, which correspond to cases where teachers prefer not to have consecutive classes and prefer to have consecutive classes, respectively. The term $\sum \theta_\tau 2^{\tau-1}$ reflects that as the number of consecutive teaching sessions increases, teacher dissatisfaction grows exponentially, while T_{max} represents the maximum acceptable gap between teaching sessions in a day. This design allows the scheduling system to be flexibly adjusted to meet the preferences of teachers from different regions.

Table 1. Variable descriptions.

Sets:	
C	Set of courses;
T	Set of teachers;
R	Set of classrooms;
A	Set of course types;
D	Set of days in a week, with a weekly schedule consisting of 5 days;
W	Set of weeks;
E	Set of administrative classes;
P	Set of teaching classes;
M	Set of teaching events;
K	Set of time slots in a week, with 5 days in a week and 5 time slots in a day;
B_c	Weekly contact hours of course c, $c \in C$;
C_t	Set of courses that the teacher t needs to teach; $t \in T$.
C_r	Set of courses scheduled in classroom r, $r \in R$;
C_a	Set of courses of type a, $a \in A$;
C_e	Set of courses for administrative class e, $e \in E$;
C_m	Set of courses specified by teaching event m, $m \in M$;
R_c	Set of classrooms of course c, $c \in C$;
R_a	Set of classrooms of type a, $a \in A$;
W_m	Set of weeks specified by teaching event m, $m \in M$;
E_p	Set of administrative classes belonging to teaching class p, $p \in P$;
M_1	Set of teaching events with fixed time slots;
K_m	Set of time slots specified by teaching event m, $m \in M$.
Parameters:	
N_r	The maximum capacity of classroom r, $r \in R$;
S_p	The number of administrative classes belonging to teaching class p, $p \in P$;
$\delta_{r,w,k}$	The number of students in classroom r during time slot k in week w, $r \in R$, $w \in W$, $k \in K$;
$Y_{e,d}$	The number of class hours scheduled for administrative class e on day d, $e \in E$, $d \in D$;
$Z_{t,d}$	The number of teaching hours scheduled for teacher t on day d, $t \in T$, $d \in D$;
Decision Variables:	
$x_{w,k,c}$	1 if course c is scheduled in week w during time slot k; 0 otherwise.
$L_{w,k,r,e}$	1 if administrative class e uses classroom r during time slot k in week w; 0 otherwise.
$\theta_{r,k}$	1 if a course is scheduled in time slot k of classroom r during the set of weeks w; 0 otherwise.
$\sigma_{w,d,c}$	1 if course c is scheduled on day d of week w; 0 otherwise.
a_c	The type of course c, $c \in C$.
a_r	The type of classroom r, $r \in R$.
Q_t	The quality of the teaching schedule for teacher t, $t \in T$.
τ	The number of consecutive teaching sessions for teacher t, $t \in T$.
θ_τ	1 if $\tau > 1$; 0 otherwise.

Table 2. Overall preference value of the teachers.

Timeslot	Mon	Tue	Wed	Thu	Fri
1	1	3	2	−2	6
2	2	4	3	−2	6
3	4	6	−2	4	5
4	3	3	−2	4	−2
5	0	0	−2	3	−2

2.4. Hard Constraints

1. Constraint Equation (3) ensures that a teacher can only teach one course at a time.

$$\sum_{c \in C_t} x_{w,k,c} \leq 1, \forall w \in W, k \in K, t \in T \tag{3}$$

2. Constraint Equation (4) ensures that a classroom can only accommodate one course at a time.

$$\sum_{c \in C_r} x_{w,k,c} \leq 1, \forall w \in W, k \in K, r \in R \tag{4}$$

3. Constraint Equation (5) ensures that a class can only be scheduled for one course at a time.

$$\sum_{r \in R} L_{w,k,,r,e} \leq 1, \forall w \in W, k \in K, e \in E \tag{5}$$

4. Constraint Equation (6) ensures that the required number of teaching hours for each course is met.

$$\sum_{w \in W} \sum_{k \in K} x_{w,k,c} = b_c, \forall c \in C, b_c \in B_c \tag{6}$$

5. Constraint Equation (7) ensures that the number of students in a class does not exceed the capacity of the classroom.

$$\delta_{r,w,k} \leq N_r, \forall r \in R, w \in W, k \in K \tag{7}$$

6. Constraint Equation (8) ensures that the relevant administrative classes for combined courses are scheduled at the same time and in the same location.

$$\sum_{e \in E_p} L_{w,k,r,e} = S_p, \forall w \in W, k \in K, r \in R \tag{8}$$

7. Constraint Equation (9) ensures that special courses are scheduled in specialized classrooms.

$$a_c = a_r, \forall c \in C, r \in R_r \tag{9}$$

8. Constraint Equation (10) ensures that if the time slot for a teaching event is fixed, the time slot for the teaching class remains unchanged.

$$x_{w,k,c} = 1, \forall m \in M_1, w \in W_m, k \in k_m, c \in c_m \tag{10}$$

9. Constraint Equation (11) ensures that the required number of class hours for each course is met.

$$2 \sum_{w \in W} \sum_{k \in K} x_{w,k,c} = b_c, \forall c \in C, b_c \in B_c \tag{11}$$

2.5. Soft Constraints

1. Constraint Equation (12) specifies that a course should be distributed as evenly as possible throughout the week.

$$\min_{c \in C}\left(\sum_{w \in W}\sum_{k \in K} x_{w,k,c} - \sum_{w \in W}\sum_{d \in D} \sigma_{w,d,c}\right) \tag{12}$$

2. Constraints Equations (13) and (14) specify that the courses for each administrative class should be distributed as evenly as possible.

$$\min_{e \in E} \frac{1}{4} \sum_{d \in D} (Y_{e,d} - \bar{Y}_e)^2 \tag{13}$$

$$\bar{Y}_e = \frac{1}{5} \sum_{d \in D} Y_{e,d}, \forall e \in E \tag{14}$$

3. Constraints Equations (15) and (16) specify that a teacher's teaching times should be distributed as evenly as possible.

$$\min_{t \in T} \frac{1}{4} \sum_{d \in D} (Z_{t,d} - \bar{Z}_t)^2 \tag{15}$$

$$\bar{Z}_t = \frac{1}{5} \sum_{d \in D} Z_{t,d}, \forall t \in T \tag{16}$$

4. Constraint Equation (17) specifies that the teaching time preferences of teachers should be met as closely as possible.

$$\min\left(\sum_{t \in T} (Q_t - V_t)\right) \tag{17}$$

5. Constraint Equation (18) specifies that the utilization of classrooms and equipment should be maximized as much as possible.

$$\max\left(\frac{1}{\sum_{r \in R}\sum_{k \in K} \theta_{r,k}} \sum_{r \in R}\sum_{w \in W}\sum_{k \in K} \frac{\theta_{r,k}\delta_{r,w,k}}{20N_r}\right) \tag{18}$$

Hard constraints refer to conditions that must be strictly met; any violation of these constraints would render the scheduling plan invalid. Soft constraints refer to conditions that are desirable to meet as closely as possible; although violations of these constraints do not invalidate the scheduling plan, they do reduce the quality of the plan. Our primary goal is to adhere to all hard constraints while minimizing violations of the soft constraints.

However, due to the complexity of combined and independent scheduling, the need to process large-scale data, and the constraints of costs and resources, most popular algorithms have faced difficulties in practical applications. For example, combined scheduling requires the coordination of multiple class and teacher schedules, which increases the complexity of the scheduling process. Additionally, processing a large volume of scheduling data requires efficient algorithms and optimization strategies, which imposes high demands on the existing methods. Traditional heuristic and greedy algorithms have often struggled to handle such complex problems and have been prone to local optima. Furthermore, schools typically face challenges of limited resources, such as insufficient classroom availability and teacher time conflicts, which further increase the difficulty of achieving an optimized scheduling solution. Therefore, the next section describes a more effective approach.

3. Optimizing the UCSP Using POGA-DP

This section proposed a two-phase metaheuristic algorithm that combines genetic algorithms and dynamic programming to effectively address the complex scheduling needs for university courses.

3.1. The Solution Method

The general process of the hybrid genetic–dynamic programming algorithm proposed in this paper can be divided into the following two phases.

In the first phase, a genetic algorithm was initially used to schedule the class times for combined scheduling tasks. After obtaining a solution with a sufficient fit, the genetic algorithm was then applied to scheduling the time slots for independent scheduling tasks. Subsequently, a combination of greedy algorithms and dynamic programming was employed to allocate classroom locations for the courses, ensuring the optimal distribution of classroom resources.

3.2. Time Slot Scheduling Based on Genetic Algorithms

In the relevant literature, genetic algorithms (GAs) and their variants have been shown to be highly effective in solving scheduling problems [37–39]. These algorithms, by simulating the processes of natural selection and genetic mutation, were able to find near-optimal solutions within complex search spaces. Therefore, this paper employs a genetic algorithm to progressively optimize the scheduling of the course time slots, aiming to effectively address the complex university scheduling problem, which includes both combined and independent scheduling tasks.

3.2.1. Chromosome Encoding

To clearly illustrate the complex relationships among different factors in the course scheduling problem, the encoding method for chromosomes is shown in Figure 1. The figure shows a summary view of the solution for administrative classes, where $S_1, S_2, \ldots, S_n$ represent the solutions for the 1st, 2nd, …, nth administrative class. S_i displays the detailed course allocation for each of the 25 time slots for an administrative class, with each time slot indicating one of the five teaching periods on the five working days of the week. (e_1, k_1) represents the course schedule for administrative class e_1 during time slot k_1, (e_1, k_2) represents the course schedule for administrative class e_1 during time slot k_2, and so on. Each time slot includes information about the corresponding administrative class, assigned teaching class, assigned course, assigned teacher, and the specific time slot.

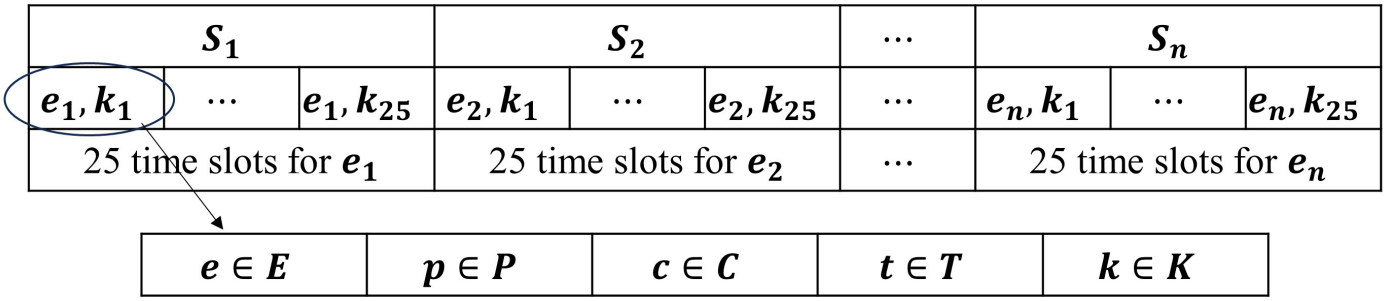

Figure 1. Descriptive chromosome representation.

3.2.2. The Swap Operation with the Judgment Mechanism

The crossover operation [40] involves exchanging segments of genetic information between two individuals during the genetic process, thereby generating new individuals. In this way, new advantageous genes may be produced, significantly enhancing the search

capability of the algorithm. Consider a UCSP consisting of two classes, e_1 and e_2, with multiple courses scheduled across 25 time slots. Figure 2 shows two different solutions, A and B, for the UCSP. Compared to solution A, in solution B, the time slots for courses c_4 and c_5 have been swapped.

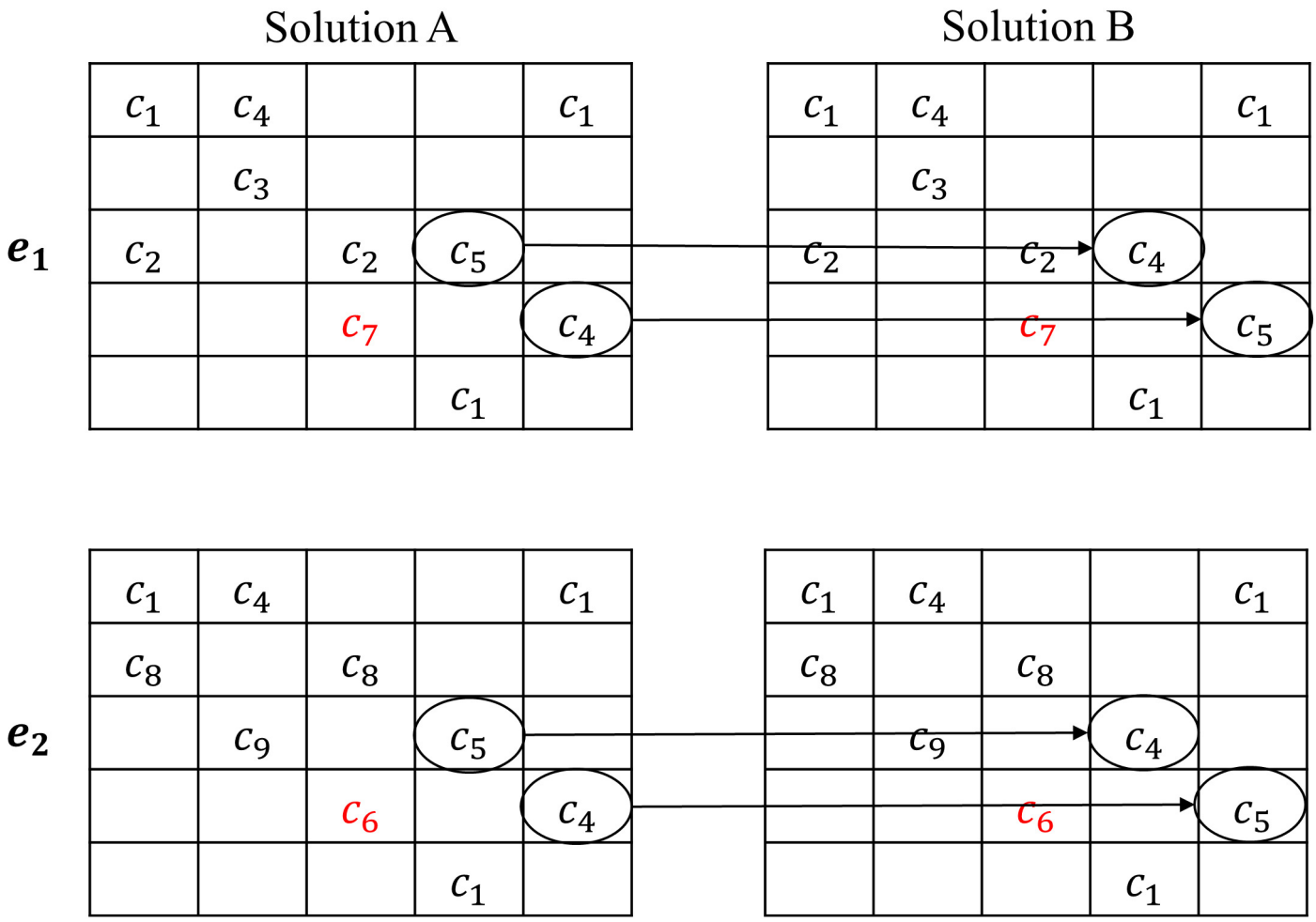

Figure 2. Course swap diagram for administrative classes.

It is important to note that when swapping courses for e_1, courses c_5 and c_7 cannot have their time slots exchanged. This is because c_5 is a joint class for both e_1 and e_2, and if c_5 is moved to the c_7 time slot, this would result in e_2 needing to schedule both c_5 and c_6 in the same time slot, which clearly violates the hard constraints. Specifically, if a course needs to be moved to a new time slot, all related entities (such as classes or teachers) must be available at that time. Therefore, the swap operation involves a validation mechanism to ensure that invalid solutions are not generated during the swap process. This mechanism works by checking whether the related entities of a randomly selected course are free in the new time slot.

3.2.3. The Forced Mutation Operation with the Repair Mechanism

We perform mutation operations based on a specific probability threshold. However, since most of the constraints in the UCSP are interrelated, many mutation operations are infeasible due to constraint violations. Therefore, forced mutation operations involve a repair mechanism to ensure that invalid solutions are not generated in the process. The repair mechanism works by randomly relocating conflicting courses to non-conflicting positions. As shown in Figure 3, consider a UCSP involving two classes, e_1 and e_2, with multiple

course schedules within specified time slots. If a mutation changes the time slot of course c_2 for e_1 from slot 4 to slot 3, this will result in teacher t_2 having to teach both c_2 and c_4 in the same time slot, which clearly violates the constraints. The repair mechanism involves randomly relocating the conflicting course c_2 to other time slots until the conflict is resolved.

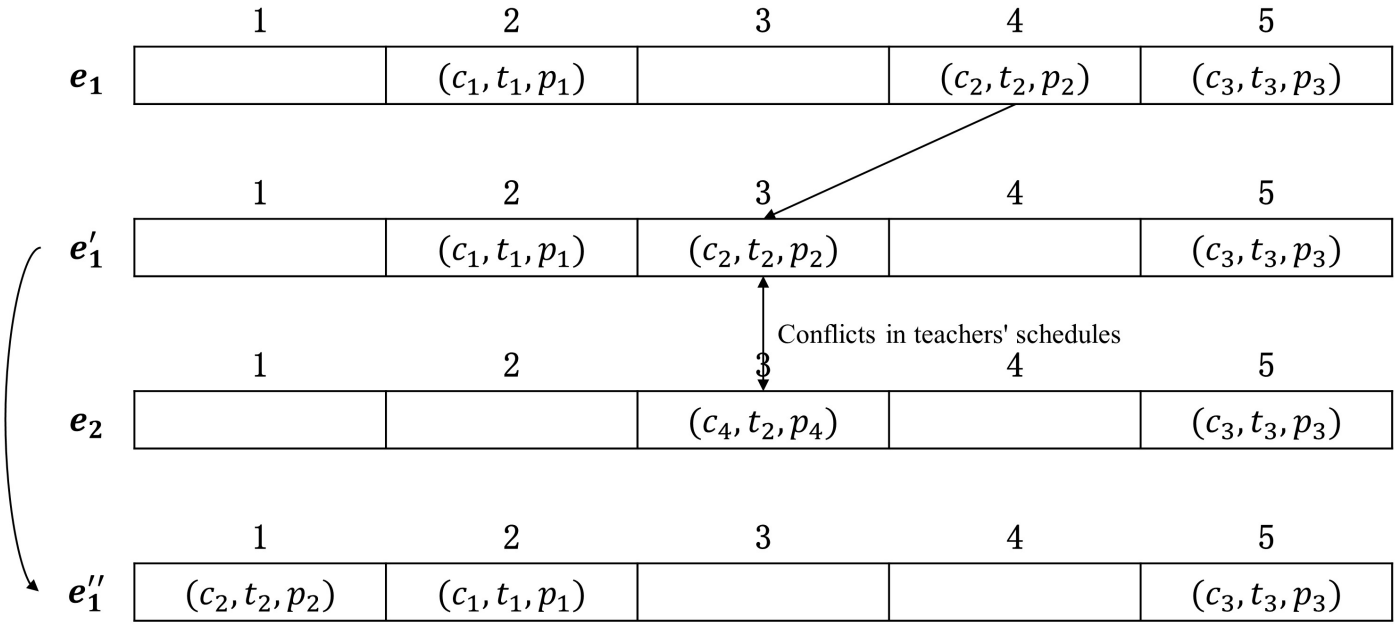

Figure 3. Diagram of the repair mechanism.

3.3. Classroom Scheduling Based on Dynamic Programming

In solving the classroom allocation problem, we employed an improved dynamic programming approach to minimize the seat resource wastage. We defined $dp[i]$ as the minimum seat wastage for scheduling the first i teaching classes in the set of classrooms. Here, N_j represents the maximum capacity of classroom j, P_j denotes the number of students in teaching class i, W_i denotes the set of weeks for teaching class i, and $\nu_{j,w}$ indicates the usage of classroom j in week w. The initial conditions are as follows:

$$dp[0] = 1 \tag{19}$$

$$dp[i] = \infty, 0 < i \leq \ m \tag{20}$$

$$\nu_{j,w} = 1, 1 < j \leq j \leq n, 1 \leq w \leq \ 20 \tag{21}$$

$$F(x, y) = \begin{cases} x - y, x \geq \ y \\ \infty, x < y \end{cases} \tag{22}$$

For $dp[1]$, this can be updated using the following formula:

$$\begin{cases} dp[1] = \min\limits_{1 \leq j \leq n} (dp[1], dp[0] + \sum\limits_{w \in W_i} F(N_j, P_1 * \nu_{j,w})) \\ \nu_{j,w} = 0, \forall w \in W_i \end{cases} \tag{23}$$

where $\nu_{j,w} = 0$ indicates that once classroom j has been used, it will not be available in week w, aiming to avoid the reuse of an already occupied classroom. Therefore, the iterative equation is given by

$$\begin{cases} dp[j] = \min\limits_{1 \leq j \leq n} (dp[i], dp[i-1] + \sum\limits_{w \in W_i} F(N_j, P_i * \nu_{j,w})) \\ \nu_{j,w} = 0, \forall w \in W_i \end{cases} \tag{24}$$

It is worth mentioning that iterating from the smallest to the largest class sizes will lead to a higher average utilization and flexibility in the scheduling results. Consider a UCSP consisting of four classes with the seating demands $P = [30, 50, 20, 20]$, weekly demands $W = [(1,1),(2,2),(2,3),(1,1)]$, classroom capacities $N = [40, 30, 60]$, and weekly availability $\nu = [(1,1,0),(1,1,1),(1,1,1)]$. As shown in Figure 4, although the results for $dp[4]$ are the same, a different ordering will produce two distinct results. The ordered scheduling improves the average utilization of the classrooms and accommodates changes in future class sizes better.

	$dp[1]$	$dp[2]$	$dp[3]$	$dp[4]$
Sorted	10	30	40	50
Unsorted	0	10	30	50

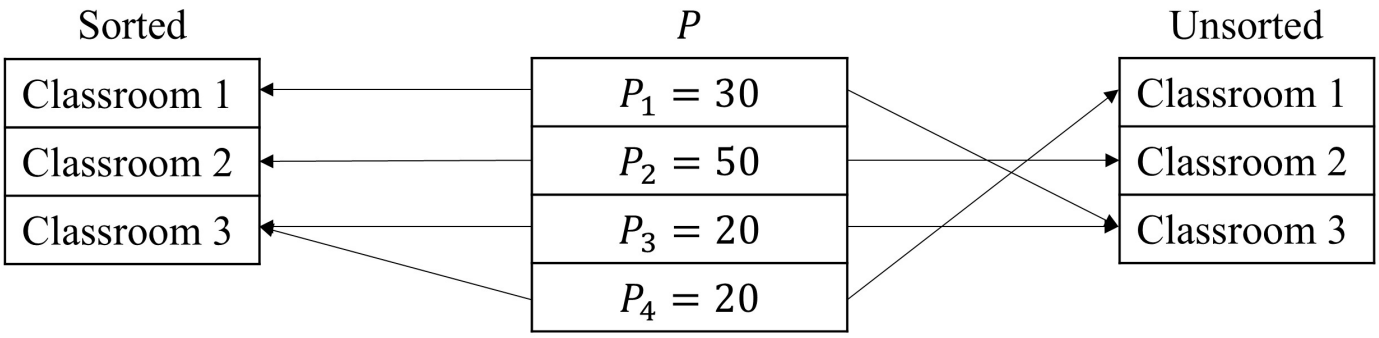

Figure 4. Classroom scheduling implemented using dynamic programming.

4. Numerical Experiments

This section explores the effectiveness and performance of the POGA-DP method in UCSP-BJFU instances. We compare the proposed approach with various traditional algorithms, including a genetic algorithm (GA), the Producer–Scrounger Method (PSM), Ant Colony Optimization (ACO), and particle swarm optimization (PSO). Additionally, we analyze and validate the impact of the scheduling similarity between adjacent weeks on the final scheduling results.

GAs have been successfully applied in various fields [41]. To facilitate their implementation in the UCSP, this study employs a single-point crossover operation with a crossover probability of 0.3. The mutation operation is performed by randomly selecting a teacher and modifying their course time slot with a mutation probability of 0.01.

The PSM is a population-based optimization algorithm proposed by Akhand [32]. It enhances the problem-solving capability by simulating different roles and behavioral patterns within a group. In each iteration, the producers attempt to explore better solutions, the scroungers move toward the producers, and dispersed members perform random movements to increase the solution diversity and prevent the algorithm from becoming trapped in local optima. To optimize the university course scheduling problem, this study employs a swapping selection probability of 0.3 in the PSM, enhancing the algorithm's search capability and solution efficiency.

PSO is a swarm-intelligence-based optimization method [42,43] that enhances the problem-solving capabilities by simulating the collaborative search behavior of particle populations. In each iteration, a particle's velocity and position are updated based on its personal best position and the global best position. To optimize the university course

scheduling problem, this study adopts an inertia weight of 0.8 to balance the global and local search capabilities. Additionally, both the individual learning factor (c1) and the social learning factor (c2) are set to 1.5, ensuring that the particles can explore new solution spaces while efficiently converging to high-quality solutions.

ACO is a bio-inspired optimization method that simulates the pheromone deposition and sensing mechanisms of ants during foraging to identify the optimal paths. In this study, the pheromone evaporation factor is set to 0.5 to prevent premature convergence to local optima while enhancing the global search capability. Additionally, a global pheromone update strategy is adopted, where only the best solution found in each iteration reinforces the pheromone trail, thereby improving the solution quality. To optimize the university course scheduling problem, the pheromone importance parameter α is set to 1, and the heuristic information importance parameter β is set to 2. This configuration ensures a balance between exploring new solutions and leveraging the historical best solutions, thereby enhancing the stability of the scheduling results and the overall optimization performance.

4.1. Experimental Examples and Settings

Based on the varying proportions of joint and independent scheduling, we generated a total of fifteen instances. Instances 1 to 5 are small-scale examples containing only independent scheduling, instances 6 to 10 are small-scale examples with only joint scheduling, and instances 11 to 15 are large-scale examples with a high proportion of joint scheduling, closely matching real-world demands. Table 3 presents the specific composition of these instances.

Table 3. Instance composition.

Instance	Combined Courses	Independent Courses	Teachers	Total Courses
Instance 1	0	6	5	6
Instance 2	0	16	8	16
Instance 3	0	26	13	26
Instance 4	0	33	19	33
Instance 5	0	57	38	57
Instance 6	18	0	7	18
Instance 7	42	0	16	42
Instance 8	177	0	55	177
Instance 9	302	0	92	302
Instance 10	463	0	148	463
Instance 11	5	6	8	11
Instance 12	10	6	10	16
Instance 13	42	16	24	58
Instance 14	177	57	92	234
Instance 15	463	57	176	520

The parameter settings of heuristic algorithms have a significant impact on their performance. The main parameters involved in this paper's algorithm include the maximum number of iterations T_{max}, the mutation probability P_m, and the crossover probability P_c. Based on preliminary experiments, we set T_{max}, P_m, and P_c to 1000, 0.01, and 0.8, respectively, to ensure that each experiment can be completed within 60 min. Since the experimental dataset comes from Chinese universities, we specifically consider the preference of Chinese university teachers to avoid consecutive classes. Therefore, in the penalty function V_t , we set the weight parameter ω_1 to 1 to reflect the penalty for consecutive teaching and ω_2 to 0 to disregard the penalty for non-consecutive teaching.

4.2. Experimental Results

To verify the reproducibility and stability of the experimental results, we performed 10 iterations for each instance. Table 4 displays the average fitness of different algorithms, as well as a comparison of the best fitness achieved by different methods under varying population sizes.

Table 4. Comparison of fitness values.

Instance	Average Fitness					Best Fitness				
	GA	PSO	PSM	POGA-DP	ACO	GA	PSO	PSM	POGA-DP	ACO
Instance 1	41.0	33.7	38.5	41.0	39.8	41	35	41	41	40
Instance 2	73.4	60.5	76.4	106.7	75.2	82	66	79	108	78
Instance 3	131.1	109.0	130.2	206.8	128.5	144	115	138	212	136
Instance 4	139.8	114.7	140.7	245.2	138.1	153	126	146	263	145
Instance 5	233.6	178.9	242.8	449.6	238.4	262	194	254	459	250
Instance 6	56.8	55.3	57.2	72.0	56.5	57	59	61	73	60
Instance 7	100.0	97.2	105.9	143.3	103.8	102	98	108	146	105
Instance 8	338.3	278.4	331.6	544.5	330.2	360	297	348	551	342
Instance 9	602.5	506.6	572.1	924.0	580.3	631	524	599	932	590
Instance 10	929.6	732.5	897.0	1437.2	910.7	977	722	922	1448	915
Instance 11	40.0	34.4	49.8	77.0	46.2	40	36	52	77	48
Instance 12	63.3	58.9	64.6	100.9	62.5	66	61	67	101	65
Instance 13	175.9	133.6	175.3	244.0	172.8	184	140	177	248	174
Instance 14	551.3	452.8	561.4	905.1	550.6	583	472	578	912	570
Instance 15	1133	941.3	1077.1	1719.8	1075.4	1183	970	1104	1739	1098

From Table 4, it can be seen that in simpler instances (such as Instance 1), the performance differences among the algorithms are relatively small. However, as the complexity of the instances increases, particularly when the number of joint courses and the total number of courses significantly grow, POGA-DP demonstrates a remarkable advantage in both its average fitness and best fitness compared to these values with the other algorithms. For example, in Instance 15, the average fitness of the GA, PSO, PSM, ACO, and POGA-DP is 1133, 941.3, 1077.1, 1075.4, and 1719.8, respectively, clearly indicating that POGA-DP is far ahead of the other algorithms. This indicates that while POGA-DP performs comparably to the other algorithms when handling low-complexity problems, it exhibits exceptional optimization capabilities as the complexity of the problem increases, especially when there are many joint courses.

In this study, a new classroom utilization metric was proposed, aimed at more effectively allocating classroom resources and improving the quality of the scheduling results. The classroom utilization rate of the scheduling results is calculated using the following formula:

$$Occupancy = \frac{1}{\sum_{r\in R}\sum_{k\in K}\theta_{r,k}} \sum_{r\in R}\sum_{w\in W}\sum_{k\in K}\frac{\theta_{r,k}\delta_{r,w,k}}{20N_r} \tag{25}$$

In practical applications, due to factors such as the length of the semester, holidays, and activities, the scheduling of teaching events often does not span the entire semester. After assigning a suitable classroom r to teaching event m, if this teaching event occurs only from week 1 to week 10 and no other teaching events are scheduled for r in subsequent weeks in this period, the space resources of the classroom will be significantly wasted. By considering dimensions such as weeks and time slots in the Occupancy metric, this approach reflects the actual usage of the classroom better. Table 5 presents a comparison of the GA and POGA-DP in solving the UCSP-BJFU with different instances, detailing the

number of classrooms required, the overall classroom utilization rate, and the standard deviation in the utilization rates.

Table 5. Classroom utilization in the scheduling scheme.

Instance	Occupancy		Occupancy Std. Dev.		Classroom Usage	
	GA	POGA-DP	GA	POGA-DP	GA	POGA-DP
Instance 1	66.000%	66.000%	0.000%	0.000%	3	3
Instance 2	49.611%	48.542%	0.371%	0.329%	6	5
Instance 3	37.422%	38.374%	0.642%	0.770%	12	10
Instance 4	34.790%	36.586%	0.628%	1.604%	13	13
Instance 5	27.461%	35.268%	1.184%	0.886%	25	17
Instance 6	45.720%	47.382%	0.255%	0.270%	9	6
Instance 7	51.909%	51.871%	0.176%	0.276%	14	11
Instance 8	45.739%	40.733%	0.590%	0.629%	34	23
Instance 9	43.357%	45.067%	0.870%	0.608%	58	37
Instance 10	46.111%	42.073%	1.290%	0.952%	72	49
Instance 11	51.435%	55.250%	0.157%	0.000%	6	6
Instance 12	50.357%	51.972%	0.247%	0.095%	8	8
Instance 13	45.708%	50.612%	0.406%	0.368%	20	15
Instance 14	40.199%	39.042%	1.045%	0.641%	48	38
Instance 15	42.111%	43.995%	1.525%	0.885%	82	58

In the comparison of the classroom utilization rates, the GA and POGA-DP exhibit overall similar occupancy rates. However, POGA-DP achieves a higher occupancy in certain instances (e.g., Instances 5, 8, 10, and 13). For example, in Instance 5, the occupancy rate of the GA is 27.461%, whereas POGA-DP increases it to 35.268%, indicating that POGA-DP may prioritize improving the classroom utilization during the optimization process. Nevertheless, in some cases (e.g., Instances 8 and 10), POGA-DP instead reduces the occupancy rate. For instance, in Instance 8, the GA achieves an occupancy rate of 45.739%, while POGA-DP only reaches 40.733%, suggesting that POGA-DP's optimization strategy does not solely aim to maximize the utilization but also takes balanced resource allocation into account.

In terms of standard deviation, the GA exhibits greater fluctuations in certain instances, such as Instance 10 (GA: 1.290%,; POGA-DP: 0.952%) and Instance 15 (GA: 1.525%; POGA-DP: 0.885%). This indicates that the GA has larger variations in the utilization rates in these cases, whereas POGA-DP demonstrates relatively greater stability. However, in some instances (e.g., Instance 4), POGA-DP shows a higher fluctuation (1.604%) compared to that with the GA (0.628%), suggesting that POGA-DP may lead to more dynamic resource adjustments in specific cases, potentially affecting the overall scheduling stability. Despite the slightly higher variability in isolated cases, POGA-DP maintains a generally superior level of stability and significantly enhances the resource utilization efficiency across multiple instances.

In terms of the number of classrooms required, the GA method generally demands more classrooms. For example, in Instance 9, the GA requires 58 classrooms, whereas POGA-DP only needs 37, indicating that POGA-DP can reduce the classroom demand while maintaining the same or even higher occupancy rates, thereby improving the resource utilization efficiency. This trend is evident in multiple instances (e.g., Instances 5, 8, 10, 14, and 15). Notably, in Instance 15, the GA requires 82 classrooms, whereas POGA-DP only requires 58, achieving nearly a 30% reduction in the resource usage. However, in certain cases, the number of classrooms required by the GA and POGA-DP remains the same

(e.g., Instances 1, 11, and 12), suggesting that both methods exhibit a similar optimization performance under specific conditions.

Overall, POGA-DP demonstrates a superior performance in handling complex scheduling problems, effectively increasing utilization while reducing the number of classrooms used, especially in multiple complex instances. Although the GA can achieve a higher classroom utilization in certain specific scenarios, it typically requires more classrooms, particularly in complex problems, where this gap is particularly pronounced. For instance, in Instance 15, which represents the most complex scheduling problem, POGA-DP's classroom utilization is slightly higher than that of the GA, while the number of classrooms used is significantly lower, further proving POGA-DP's significant advantages in resource optimization.

4.3. The Impact of the Course Arrangement Similarity Between Adjacent Weeks

The similarity of course arrangements between adjacent weeks typically refers to whether the same course is scheduled in the same time slot and classroom across two consecutive weeks. This similarity helps students and teachers establish fixed daily routines and reduces confusion being caused by frequent changes. Table 6 shows the scheduling results without considering the course arrangement similarity, which exhibit significant instability and unpredictability, making it difficult to meet practical needs. In contrast, the scheduling results displayed in Table 7 show a high level of consistency in the course arrangements between adjacent weeks, significantly enhancing the stability and predictability of the scheduling plan.

Table 6. Results without considering course arrangement similarity.

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	0	8	0	0	0	0	8	0	0	0	4	0	5	0	0	0	0	0	5	0	5	7	0	7
2	0	5	8	4	2	7	0	0	0	0	5	0	0	0	0	0	0	0	0	2	0	8	5	7	0
3	0	0	5	0	8	4	2	0	9	7	3	0	2	0	9	0	0	0	0	3	0	5	8	5	7
4	5	3	8	4	9	0	2	7	3	0	2	0	9	0	0	0	0	0	0	5	8	5	7	0	0
5	5	3	8	0	9	0	2	3	7	0	2	0	0	0	0	0	0	0	0	8	5	7	5	9	0
6	3	5	0	0	0	0	2	0	7	8	2	0	0	3	0	0	0	8	0	5	0	0	0	5	7
7	0	0	0	5	9	0	2	7	3	8	2	6	9	0	0	0	3	0	0	5	8	5	7	0	0
8	0	5	3	8	0	0	0	2	7	3	0	2	6	0	0	0	0	0	0	0	5	8	5	7	0
9	3	5	0	8	9	0	2	7	3	0	2	6	9	0	0	0	0	5	0	0	0	0	7	8	5
10	0	0	5	3	8	0	0	0	2	7	3	0	2	6	0	0	0	0	0	0	0	5	8	5	7
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
13	5	0	8	0	9	0	0	0	0	0	0	6	9	0	0	0	0	1	1	5	8	5	0	0	0
14	0	2	0	5	0	8	0	0	6	0	2	0	0	0	0	0	0	1	1	0	5	0	5	0	8
15	5	0	0	0	9	0	2	0	0	0	2	6	9	0	0	0	0	1	1	0	0	0	5	0	5
16	0	0	0	0	0	0	0	5	0	0	0	6	0	0	5	0	0	0	0	5	0	0	0	0	0
17	5	0	0	0	0	0	8	0	0	0	0	5	0	0	0	0	0	0	0	0	8	5	0	0	0
18	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7. Results considering course arrangement similarity.

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	5	0	8	4	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	5	8	5	7	0	0
2	5	0	8	4	0	0	2	7	0	0	2	0	0	0	0	0	0	0	0	5	8	5	7	0	0
3	5	3	8	4	9	0	2	7	3	0	2	0	9	0	0	0	0	0	0	5	8	5	7	0	0
4	5	3	8	4	9	0	2	7	3	0	2	0	9	0	0	0	0	0	0	5	8	5	7	0	0
5	5	3	8	0	9	0	2	7	3	0	2	0	9	0	0	0	0	0	0	5	8	5	7	0	0
6	5	3	8	0	0	0	2	7	3	0	2	0	0	0	0	0	0	0	0	5	8	5	7	0	0
7	5	3	8	0	9	0	2	7	3	0	2	6	9	0	0	0	0	0	0	5	8	5	7	0	0
8	5	3	8	0	0	0	2	7	3	0	2	6	0	0	0	0	0	0	0	5	8	5	7	0	0
9	5	3	8	0	9	0	2	7	3	0	2	6	9	0	0	0	0	0	0	5	8	5	7	0	0
10	5	3	8	0	0	0	2	7	3	0	2	6	0	0	0	0	0	0	0	5	8	5	7	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
13	5	0	8	0	9	0	0	0	0	0	0	6	9	0	0	0	0	1	1	5	8	5	0	0	0
14	5	0	8	0	0	0	2	0	0	0	2	6	0	0	0	0	0	1	1	5	8	5	0	0	0
15	5	0	0	0	9	0	2	0	0	0	2	6	9	0	0	0	0	1	1	5	0	5	0	0	0
16	5	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	5	0	5	0	0	0
17	5	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	8	5	0	0	0
18	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5. Discussion

We increased the number of iterations from 20 to 1000 and measured the fitness values of the different algorithms for Instance 15, with the results shown in Figure 5. The experimental results indicate that in the early stages of iteration, POGA-DP performed similarly to the GA, slightly outperforming PSO and ACO. However, after 40 iterations, the fitness value of POGA-DP rapidly increased to 961, nearly reaching the level of the GA, the PSM, and ACO after 500 iterations and significantly surpassing PSO's 733. This demonstrates that as the number of iterations increases, POGA-DP can continuously optimize the solution quality, gradually finding better solutions and showcasing its strong optimization capability in handling complex course scheduling problems.

In previous studies, GAs and PSO have typically shown various limitations when addressing large-scale complex optimization problems. For instance, GAs often suffer from a slow convergence speed, while PSO tends to become trapped in local optima. In this experiment, the GA and PSO maintained these characteristics: the GA exhibited a slow improvement in the fitness values with fewer iterations, and PSO stagnated after reaching a certain number of iterations. The PSM's performance also aligned with the literature, as it quickly converged in terms of its fitness with fewer iterations but showed a diminished improvement afterward. POGA-DP's outstanding performance in this experiment demonstrated its advantage in improving the fitness values, particularly after 100 iterations, where its growth rate significantly surpassed that of the other algorithms. This improvement is closely related to the algorithm's design, which integrates dynamic programming with a genetic algorithm.

One limitation of this study lies in the lack of diversity in the experimental datasets. While the experiments demonstrated the superiority of POGA-DP under the given conditions, all tests were conducted on the same type of dataset, which may not have fully reflected the algorithm's generality across different problem scales, constraint conditions, or optimization objectives. Additionally, although POGA-DP performed well, the flexibility of the comparative algorithms in parameter tuning may not have been fully explored. For instance, the performance of the GA and PSO is highly influenced by the parameter selection and adjustment.

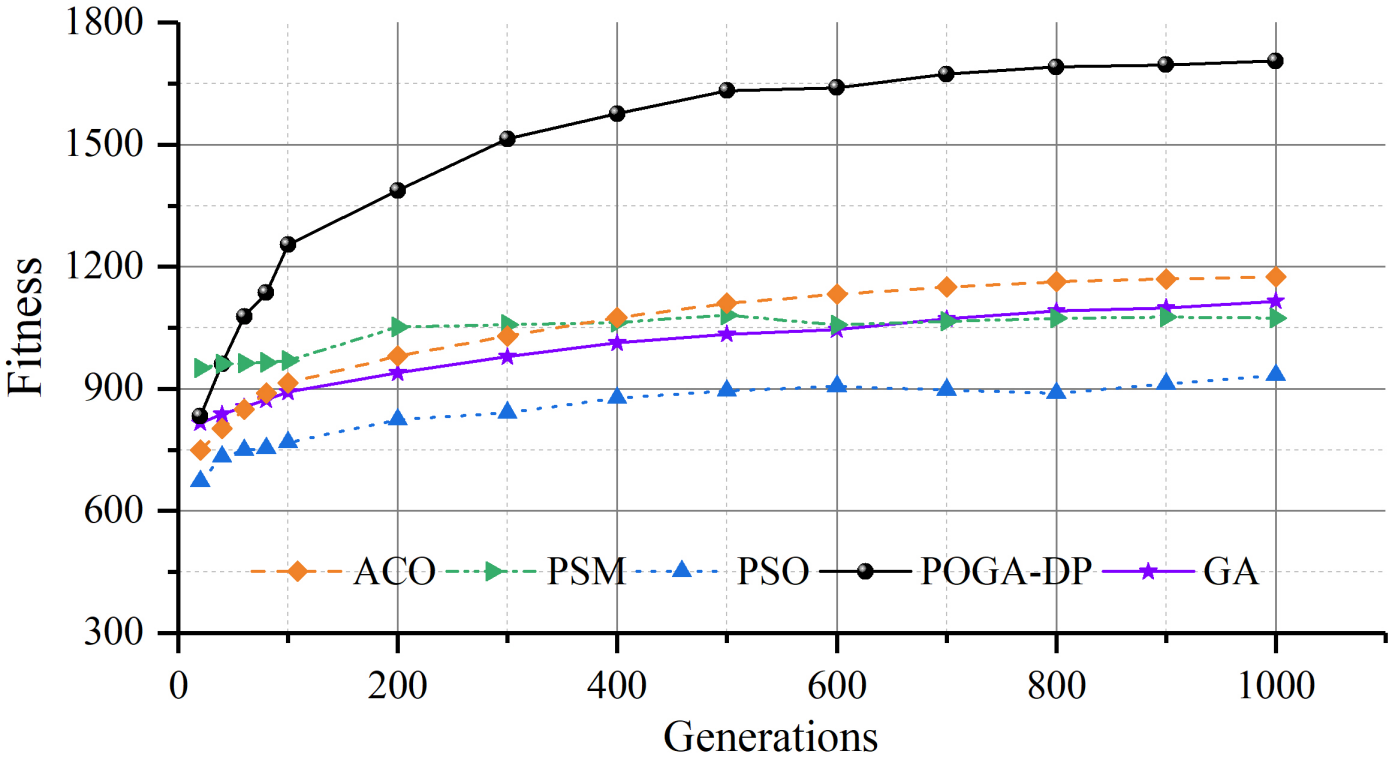

Figure 5. Performance analysis of fitness across different iterations.

To address the aforementioned limitations, future research could introduce more diverse datasets to further validate the performance of POGA-DP, including optimization problems of varying scales and constraint conditions to assess its generality. Additionally, optimizing the parameter configurations of the GA and PSO, such as through adaptive parameter adjustments or hybrid algorithms, may further enhance their performance. Moreover, increasing the number of iterations in the experiments or testing other algorithm combinations (e.g., hybrid algorithms) could also provide more insights for evaluating the algorithms.

Future research could further explore the performance of POGA-DP by increasing the complexity of the experimental conditions, such as varying the number of joint and independent courses or considering more dynamic constraints, like classroom availability and teachers' time limitations. Additionally, comparing POGA-DP with other cutting-edge intelligent optimization algorithms (e.g., combining deep learning with genetic algorithms) could help validate its performance in more complex environments. This would provide the academic community with a more comprehensive evaluation of the algorithm and offer more valuable solutions for complex course scheduling problems in practical applications.

Author Contributions: Conceptualization: X.H. and D.W. Methodology: X.H. Software: X.H. Validation: D.W. Formal analysis: X.H. Investigation: D.W. Resources: D.W. Data curation: X.H. Writing—original draft preparation: X.H. Writing—review and editing: D.W. Visualization: X.H. Supervision: X.H. and D.W. Project administration: X.H. and D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Acknowledgments: The authors are grateful to the anonymous reviewers for their insightful comments, which certainly improved this paper.

Conflicts of Interest: The authors declare that there are no conflicts of interest.

References

1. Khamechian, M.; Petering, M.E. A mathematical modeling approach to university course planning. *Comput. Ind. Eng.* **2022**, *168*, 107855. [CrossRef]
2. Hartmanis, J. Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Rev.* **1982**, *24*, 90. [CrossRef]
3. Wang, Y.Z. Using genetic algorithm methods to solve course scheduling problems. *Expert Syst. Appl.* **2003**, *25*, 39–50. [CrossRef]
4. Morais, R.; Bulhões, T.; Subramanian, A. Exact and heuristic algorithms for minimizing the makespan on a single machine scheduling problem with sequence-dependent setup times and release dates. *Eur. J. Oper. Res.* **2024**, *315*, 442–453. [CrossRef]
5. Liu, J.; Zhang, Z.; Liu, S.; Zhang, Y.; Wu, T. Parallel hyper heuristic algorithm based on reinforcement learning for the corridor allocation problem and parallel row ordering problem. *Adv. Eng. Inform.* **2023**, *56*, 101977. [CrossRef]
6. Tomczak, M.; Jaśkowski, P. Customized particle swarm optimization for harmonizing multi-section construction projects. *Autom. Constr.* **2024**, *162*, 105359. [CrossRef]
7. Fontes, D.B.; Homayouni, S.M.; Gonçalves, J.F. A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *Eur. J. Oper. Res.* **2023**, *306*, 1140–1157. [CrossRef]
8. Wu, H.; Pang, G.K.H.; Choy, K.L.; Lam, H.Y. Dynamic resource allocation for parking lot electric vehicle recharging using heuristic fuzzy particle swarm optimization algorithm. *Appl. Soft Comput.* **2018**, *71*, 538–552. [CrossRef]
9. Muñoz-Díaz, M.L.; Escudero-Santana, A.; Lorenzo-Espejo, A. Solving an Unrelated Parallel Machines Scheduling Problem with machine-and job-dependent setups and precedence constraints considering Support Machines. *Comput. Oper. Res.* **2024**, *163*, 106511. [CrossRef]
10. Bellio, R.; Ceschia, S.; Di Gaspero, L.; Schaerf, A.; Urli, T. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Comput. Oper. Res.* **2016**, *65*, 83–92. [CrossRef]
11. Tamssaouet, K.; Dauzère-Pérès, S.; Knopp, S.; Bitar, A.; Yugma, C. Multiobjective optimization for complex flexible job-shop scheduling problems. *Eur. J. Oper. Res.* **2022**, *296*, 87–100. [CrossRef]
12. De Moraes, M.B.; Coelho, G.P. A diversity preservation method for expensive multi-objective combinatorial optimization problems using Novel-First Tabu Search and MOEA/D. *Expert Syst. Appl.* **2022**, *202*, 117251. [CrossRef]
13. Tosun, U. A new tool for automated transformation of quadratic assignment problem instances to quadratic unconstrained binary optimisation models. *Expert Syst. Appl.* **2022**, *201*, 116953. [CrossRef]
14. Lü, Z.; Hao, J.K. Adaptive tabu search for course timetabling. *Eur. J. Oper. Res.* **2010**, *200*, 235–244. [CrossRef]
15. Ji, B.; Zhang, Z.; Samson, S.Y.; Zhou, S.; Wu, G. Modelling and heuristically solving many-to-many heterogeneous vehicle routing problem with cross-docking and two-dimensional loading constraints. *Eur. J. Oper. Res.* **2023**, *306*, 1219–1235. [CrossRef]
16. Hossain, S.I.; Akhand, M.; Shuvo, M.; Siddique, N.; Adeli, H. Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert Syst. Appl.* **2019**, *127*, 9–24. [CrossRef]
17. Shiau, D.F. A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences. *Expert Syst. Appl.* **2011**, *38*, 235–248. [CrossRef]
18. Ayob, M.; Jaradat, G. Hybrid ant colony systems for course timetabling problems. In Proceedings of the 2009 2nd Conference on Data Mining and Optimization, Selangor, Malaysia, 27–28 October 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 120–126.
19. Hiryanto, L. Incorporating dynamic constraint matching into vertex-based graph coloring approach for university course timetabling problem. In Proceedings of the 2013 International Conference on QiR, Yogyakarta, Indonesia, 25–28 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 68–72.
20. Wen, X.; Zhang, X.; Xing, H.; Ye, G.; Li, H.; Zhang, Y.; Wang, H. An improved genetic algorithm based on reinforcement learning for aircraft assembly scheduling problem. *Comput. Ind. Eng.* **2024**, *193*, 110263. [CrossRef]
21. Squires, M.; Tao, X.; Elangovan, S.; Gururajan, R.; Zhou, X.; Acharya, U.R. A novel genetic algorithm based system for the scheduling of medical treatments. *Expert Syst. Appl.* **2022**, *195*, 116464. [CrossRef]
22. Xiang, K.; Hu, X.; Yu, M.; Wang, X. Exact and heuristic methods for a university course scheduling problem. *Expert Syst. Appl.* **2024**, *248*, 123383. [CrossRef]
23. Domenech, B.; Lusa, A. A MILP model for the teacher assignment problem considering teachers' preferences. *Eur. J. Oper. Res.* **2016**, *249*, 1153–1160. [CrossRef]
24. Esmaeilbeigi, R.; Mak-Hau, V.; Yearwood, J.; Nguyen, V. The multiphase course timetabling problem. *Eur. J. Oper. Res.* **2022**, *300*, 1098–1119. [CrossRef]
25. Ziadlou, G.; Emami, S.; Asadi-Gangraj, E. Network configuration distributed production scheduling problem: A constraint programming approach. *Comput. Ind. Eng.* **2024**, *188*, 109916. [CrossRef]
26. Meng, L.; Zhang, C.; Ren, Y.; Zhang, B.; Lv, C. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Comput. Ind. Eng.* **2020**, *142*, 106347. [CrossRef]

27. Qin, T.; Du, Y.; Chen, J.H.; Sha, M. Combining mixed integer programming and constraint programming to solve the integrated scheduling problem of container handling operations of a single vessel. *Eur. J. Oper. Res.* **2020**, *285*, 884–901. [CrossRef]
28. Dvořák, P.; Eiben, E.; Ganian, R.; Knop, D.; Ordyniak, S. The complexity landscape of decompositional parameters for ILP: Programs with few global variables and constraints. *Artif. Intell.* **2021**, *300*, 103561. [CrossRef]
29. Klein, N.; Gnägi, M.; Trautmann, N. Mixed-integer linear programming for project scheduling under various resource constraints. *Eur. J. Oper. Res.* **2024**, *319*, 79–88. [CrossRef]
30. Caselli, G.; Delorme, M.; Iori, M. Integer linear programming for the tutor allocation problem: A practical case in a British university. *Expert Syst. Appl.* **2022**, *187*, 115967. [CrossRef]
31. Mallari, C.B.; San Juan, J.L.; Li, R. The university coursework timetabling problem: An optimization approach to synchronizing course calendars. *Comput. Ind. Eng.* **2023**, *184*, 109561. [CrossRef]
32. Akhand, M.A.H.; Hossain, F. Producer-Scrounger Method to Solve Traveling Salesman Problem. *Int. J. Intell. Syst. Appl.* **2015**, *7*, 29–36. [CrossRef]
33. Daskalaki, S.; Birbas, T.; Housos, E. An integer programming formulation for a case study in university timetabling. *Eur. J. Oper. Res.* **2004**, *153*, 117–135. [CrossRef]
34. Thepphakorn, T.; Pongcharoen, P. Performance improvement strategies on Cuckoo Search algorithms for solving the university course timetabling problem. *Expert Syst. Appl.* **2020**, *161*, 113732. [CrossRef]
35. Akkan, C.; Gülcü, A. A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. *Comput. Oper. Res.* **2018**, *90*, 22–32. [CrossRef]
36. Thompson, G.M. Using information on unconstrained student demand to improve university course schedules. *J. Oper. Manag.* **2005**, *23*, 197–208. [CrossRef]
37. Zhang, H.; Liu, F.; Zhou, Y.; Zhang, Z. A hybrid method integrating an elite genetic algorithm with tabu search for the quadratic assignment problem. *Inf. Sci.* **2020**, *539*, 347–374. [CrossRef]
38. Qiao, Y.; Wu, N.; He, Y.; Li, Z.; Chen, T. Adaptive genetic algorithm for two-stage hybrid flow-shop scheduling with sequence-independent setup time and no-interruption requirement. *Expert Syst. Appl.* **2022**, *208*, 118068. [CrossRef]
39. Bacalhau, E.T.; Casacio, L.; de Azevedo, A.T. New hybrid genetic algorithms to solve dynamic berth allocation problem. *Expert Syst. Appl.* **2021**, *167*, 114198. [CrossRef]
40. Li, A.D.; Xue, B.; Zhang, M. Multi-objective feature selection using hybridization of a genetic algorithm and direct multisearch for key quality characteristic selection. *Inf. Sci.* **2020**, *523*, 245–265. [CrossRef]
41. Kadri, R.L.; Boctor, F.F. An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *Eur. J. Oper. Res.* **2018**, *265*, 454–462. [CrossRef]
42. Ferreira, F.A.B.S.; LeitãO, H.A.S.; Lopes, W.T.A.; Madeiro, F. Hybrid firefly-linde-buzo-gray algorithm for Channel-Optimized Vector Quantization codebook design. *Integr. Comput. Aided Eng.* **2017**, *24*, 1–18. [CrossRef]
43. Iglesias, A.; Gálvez, A.; Collantes, M. Multilayer embedded bat algorithm for B-spline curve reconstruction. *Integr. Comput.-Aided Eng.* **2017**, *24*, 385–399. [CrossRef]