Genetic and Deep Reinforcement Learning-Based Intelligent Course Scheduling For Smart Education

Sami Ahmed Haider
Electrical Electronic & Computer Engineering School of Engineering and Physical Sciences
Heriot-Watt University
Edinburgh, Scotland, United Kingdom
s.haider@hw.ac.uk

Khwaja Mutahir Ahmad
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
mutahir@std.uestc.edu.cn

Adnan Zahid
Electrical Electronic & Computer Engineering School of Engineering and Physical Sciences
Heriot-Watt University
Edinburgh, Scotland, United Kingdom
a.zahid@hw.ac.uk

Azzah AlGhamdi
Computer Information Systems Department College of Computer Science and Information Technology
Imam Abdalrhman Bin Faisal University
Khobar, Saudi Arabia
azghamdi@iau.edu.sa

Ismail Keshta
Division of Research and Computer Science and Information Systems Department College of Applied Sciences
AlMaarefa University
Riyadh, Saudi Arabia
imohamed@um.edu.sa

Mukesh Soni
Development
Lovely Professional University
Phagwara, India
mukesh.research24@gmail.com

Abstract

Scheduling courses is a routine yet important task in Smart educational activities. Traditional manual scheduling is time-consuming and labor-intensive, prone to errors, and unable to meet the demands of large-scale scheduling. Classic genetic algorithms for scheduling have issues such as rapid convergence and decreased scheduling efficiency as constraint factors increase. To address these problems in existing scheduling genetic algorithms, we propose a self-learning genetic algorithm for scheduling based on deep reinforcement learning (GAGDRL). The GAGDRL algorithm utilizes the Q-learning algorithm to achieve adaptive adjustment of crossover and mutation parameters, enhancing the search capability of the genetic algorithm. Establishing a parameter dynamic adjustment model for the Markov decision process (MDP) analyzes the state set of the population fitness function to achieve a comprehensive evaluation of the overall performance of the population. Additionally, the deep Q-network (DQN) algorithm is introduced into the scheduling problem to address issues related to the large number of population states and the volume of Q-table data in scheduling. Experimental results show that, compared to classical and improved genetic algorithms for scheduling, the GAGDRL algorithm improves accuracy and optimization capability. The proposed algorithm can also be applied to problems such as exam scheduling, cinema seating arrangements, and airline route planning.

This work is licensed under a Creative Commons Attribution International 4.0 License.

ICAITE 2024, Beijing, China
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1013-1/24/10
https://doi.org/10.1145/3702386.3702398

CCS Concepts

• **Computing methodologies: Reinforcement learning**; • **Theory of Computation: Genetic algorithm**; • **Applied computing: Interactive learning environment**;

Keywords

Genetic, Deep Reinforcement Learning, Course Scheduling, Smart Education, Markov Decision Process

ACM Reference Format:
Sami Ahmed Haider, Khwaja Mutahir Ahmad, Adnan Zahid, Azzah AlGhamdi, Ismail Keshta, and Mukesh Soni. 2024. Genetic and Deep Reinforcement Learning-Based Intelligent Course Scheduling For Smart Education. In *2024 the International Conference on Artificial Intelligence and Teacher Education (ICAITE) (ICAITE 2024), October 12–14, 2024, Beijing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3702386.3702398

1 INTRODUCTION

Scheduling classes is a routine and important task in school teaching activities. How to scientifically and efficiently schedule classes has always been an important issue in the arrangement of teaching activities. With the continuous expansion of the scale of school education in my country and the gradual change of teaching methods, the problems of time-consuming, inefficient and unscientific traditional manual scheduling methods have become increasingly prominent and can no longer meet the needs of modern education.

In recent years, scheduling algorithm research has garnered attention and yielded rich findings. Simulated annealing, genetic, and integer programming methods are common [1]. As intelligent algorithm research advances, some studies have used heuristic algorithms to handle scheduling problems. Intelligent scheduling based on genetic algorithm (GA) has the best effect [2]. Two issues remain with this clever scheduling algorithm: 1) The convergence speed of this algorithm is sometimes excessively quick, which impacts scheduling accuracy; 2) Increasing scheduling constraints

considerably reduces its execution efficiency [3-4]. These two issues stem from the evolutionary algorithm's arbitrarily determined or randomly generated parameters. Inability to adapt to computing advancements. Thus, intelligent scheduling algorithm parameter optimisation need additional study.

Reinforcement learning is a form of machine learning that makes decisions and takes actions depending on feedback received from the environment. It refines its strategy through iterative environmental involvement and trial and error to maximise cumulative reward. Reinforcement learning emphasises rewards. Research suggests that using reinforcement learning in genetic algorithms might identify optimal parameters by revealing population composition [5-6]. Some academics have used reinforcement learning to study genetic algorithm parameters with specific results. This paper [7] introduced a multi-strategy parallel genetic method using machine learning. Researchers used reinforcement learning to independently perceive environments and optimise crossover probability. They next tested the modified genetic algorithm's stability and superiority using function instance testing. Deep reinforcement learning combines deep and reinforcement learning. We use reinforcement learning to design the task and optimise the goal function, and deep learning for state representation. In reinforcement learning, a deep neural network models the value function and strategy to solve strategy expression problems. Optimisation of the objective function follows using the error backpropagation algorithm [8]. The structure, similar to reinforcement learning, simplifies addressing excessively vast or continuous state-action space. The article [9] used deep reinforcement learning to optimise CNC milling parameters. Optimisation parameters for complex CNC milling processes is the focus of this algorithm. Experimental verification shows that the suggested technique optimises parameters and improves processing efficiency relative to experience.

To address the issue of automatic class scheduling, a self-learning genetic algorithm based on deep reinforcement learning (GA-DRL) is developed, taking into account the features of deep reinforcement learning and genetic algorithm. The GA-DRL algorithm leverages the principles of reinforcement learning to adjust the parameters of the optimization algorithm dynamically. This adjustment specifically focuses on selecting the crossover and mutation probability parameters during the iterative process of the genetic algorithm. The goal is to maintain population diversity, prevent the algorithm from getting stuck in local optima, and optimize the genetic algorithm's ability to address the issue of accuracy and efficiency sensitivity in intelligent scheduling algorithms.

2 RELATED WORK

Class scheduling is a problem of optimal allocation of teaching resources. It is essentially a constrained 0-1 planning problem [10] and belongs to the NP problem. At present, this problem is often solved based on heuristic algorithms. Typical algorithms include simulated annealing algorithm [11], taboo search algorithm [12], genetic algorithm [13], ant colony algorithm [14], particle swarm algorithm [15], grey wolf optimization algorithm [16], etc. The genetic algorithm is an improved global optimal search algorithm. Compared with other improved search algorithms, it has the characteristics of global and parallel search and is suitable for solving class scheduling problems. This work [17] used a genetic algorithm to design a class scheduling system. By comprehensively analyzing student choices, the optimal Intelligent course schedule was generated, and automatic class scheduling was achieved, ensuring the rationality of the course schedule. However, there is a lack of research on the convergence problem of the optimal solution for class scheduling. This research [18] designed an improved genetic algorithm to solve the class scheduling problem under the high school class system. By formulating a targeted fitness function and performing two genetic operations, Finally, a more reasonable scheduling result is obtained, but there is also a lack of research on the convergence of the algorithm. Due to the complexity of the scheduling problem and the defects of the genetic algorithm itself, it is necessary to optimize the genetic algorithm, such as premature maturity and small processing scale [19]. The research work [20] proposed an adaptive crossover probability and mutation probability design scheme to effectively prevent being trapped in the local optimal solution and improve the global search ability. The performance of genetic algorithms is closely related to the selection of key parameters, but there is relatively little research in the academic community on how to determine the parameters of genetic algorithms to improve computing performance and problem-solving efficiency. In recent years, with the extensive and in-depth development of machine learning research, many researchers have tried to apply reinforcement learning to the dynamic adjustment of parameters of heuristic algorithms to solve optimization problems. For genetic algorithms, some studies have found that reinforcement learning has the advantage of discovering the internal structure of the group and thus finding the optimal parameters [21]. In addition, some studies have also applied reinforcement learning to adjust the parameters of the optimal algorithm and used it in processes such as product manufacturing [22], workshop scheduling [23], and elevator scheduling [24] to improve the accuracy and efficiency of heuristic algorithms in solving problems. However, there is currently a lack of research on the application of reinforcement learning to intelligent class scheduling algorithms.

3 INTELLIGENT CLASS SCHEDULING ALGORITHM BASED ON DEEP REINFORCEMENT LEARNING

3.1 PROBLEM DESCRIPTION

Scheduling is the process of arranging non-conflicting class times and classrooms for multiple teaching tasks, where a teaching task is a triple consisting of a teacher, a student, and a course. There may be multiple conflicts in scheduling, so some hard constraints must be met to avoid conflicts in scheduling. Common hard constraints are: 1) In the same period, the same classroom can offer at most one course; 2) In the same period, the same teacher can offer at most one course; 3) In the same period, the same class can offer at most one course. In addition to meeting hard constraints, scheduling should also consider the needs of schools, teachers, and students to achieve the goal of maximum resource utilization and optimal teaching effect. For example, teachers can adjust Intelligent course arrangements according to students' learning progress to ensure teaching effectiveness. Another example is that scheduling should take into account teachers' willingness to teach to ensure that

teachers can teach on time. Therefore, scheduling also needs to meet some soft constraints. Typical soft constraints include: 1) Courses should be evenly distributed in all opening periods of a week; 2) Special courses are scheduled in special periods, for example, it is best to arrange a self-study class in the first period in the afternoon; 3) Do not arrange the same course in a class on the same day; 4) The teacher's teaching time requirements.

3.2 MATHEMATICAL MODEL

The course schedule is a five-tuple (course, class, teacher, classroom, timeperiod), represented by L, C, H, R, T respectively. Each tuple is a set, as follows:

$$\begin{aligned} L &= \{l_1, l_2, l_3, \cdots, l_L\}; \\ C &= \{c_1, c_2, c_3, \cdots, c_C\}; \\ H &= \{h_1, h_2, h_3, \cdots, h_H\}; \\ R &= \{r_1, r_2, r_3, \cdots, r_R\}; \\ T &= \{t_1, t_2, t_3, \cdots, t_T\}; \end{aligned}$$

$T^i = \{t_1^i, t_2^i, t_3^i, \cdots, t_T^i\}$ represents the time period set of the i-th day.

Therefore, the common hard constraints of the scheduling problem can be expressed as follows:

In the same classroom in time period i, at most one course can be offered. The mathematical model is as follows:

$$\forall i \in T, \forall j \in R, \left(\bigcap_{k=1}^{R} C_{k(i,j)}\right) \leqslant 1 \tag{1}$$

In time period i, a teacher can only teach one course in a classroom in R_j. The mathematical model is as follows:

$$\forall i \in T, \bigcap_{j=1}^{R} H\left(i, R_j\right) = \emptyset \tag{2}$$

In time period i, a class can only have one course arranged in a classroom in R_j. The mathematical model is as follows:

$$\forall i \in T, \bigcap_{j=1}^{R} C\left(i, R_j\right) = \emptyset \tag{3}$$

The common soft constraints of the scheduling problem can be expressed as follows:

In any time period i on a certain day, a class cannot have the same course. Let k represent the day of the week. The mathematical model is as follows:

$$\forall i \in L, \bigcup_{k=1}^{5} T^k\left(i, C_j\right) = \emptyset \tag{4}$$

Therefore, the class scheduling problem can be described as a single-objective optimization model.

The decision variables are: X(l, c, h, r, t), which means that in time period t, teacher h teaches class c in classroom r and course l.

The objective function is:

$$\min = \sum_{i=1}^{5} \sum_{l=1}^{L} \operatorname{card}\left(\bigcap_{c=1}^{c} X_{t \in T'}\ (l, c, h, r, t)\right) \tag{5}$$

The constraints are equations (1)-(3).

3.3 CLASS SCHEDULING GENETIC ALGORITHM DESIGN

At present, genetic algorithm is an effective algorithm for solving constrained 0-1 programming problems [25]. Genetic algorithm is a heuristic search algorithm based on the biological evolution theory of "survival of the fittest" proposed by the biologist Darwin. It can automatically obtain and accumulate relevant knowledge about the search space during the search, and make the problem develop in a better direction by performing selection, crossover and mutation operations to obtain the optimal solution. Because class scheduling is a type of constrained 0-1 programming problem, genetic algorithm can be used to solve it.

3.3.1 CHROMOSOME REPRESENTATION AND POPULATION INITIALIZATION. Chromosomes are composed of various genes, each of which represents a course arrangement. For example, a gene can represent a teaching task arranged in a classroom, and a chromosome can represent the optimal solution to the problem. Before scheduling, the course number, class number and teacher number are scheduled using the decimal method, and the class number and class time number are automatically generated by the computer. The time number is coded according to the weekly and daily time series. The algorithm in this paper sets a total of five days of courses per week and divides them into six times a day, two classes each time. For example, 16 represents the 11th to 12th classes on Monday, and so on. The coding is expressed as "course unit code + class code + teacher code + classroom code + time code", as listed in Table 1. For example, the teacher's work number is 11, and the course is "Computer Network" (code 100044). If this course is scheduled to the 1st and 2nd classes on Tuesday, the class is 3 (coded as 3), and the class is in classroom 105 (coded as 5), then this gene code is 100044311521, where the last 21 code represents the 1st to 2nd class on Tuesday.

3.3.2 GENETIC OPERATION. This paper selects the conflict value of the scheduling plan as the adaptive function, and the conflict involves many aspects, such as classroom conflict, class conflict, teacher conflict, and the harmony of course allocation. This paper designs the adaptive function as the result of weighting the value of each conflict. This paper adopts a selection strategy based on random traversal method to select a group of individuals with high fitness (small conflict value) from the parent generation and use them as offspring. Then, new superior individuals are obtained by hybridization and mutation of the selected superior individuals. The adaptive function is defined as:

$$f = \sum_{i=1}^{n} w_i f_i \tag{6}$$

Where, w_i is the weight of each conflict factor, its value can be determined according to the actual situation, f_i is each conflict factor, and f is the objective function of the scheduling problem.

1) Classroom conflict: At the same time, in the same classroom, at most one course can be offered. If there is a conflict, $f_1 = f_1 + 1$.
2) Class conflict: At the same time, a teacher can only appear in a class to teach a course. If there is a conflict, $f_2 = f_2 + 1$.
3) Teacher conflict: At the same time, a class can only offer one course at most. If there is a conflict, $f_3 = f_3 + 1$.
4) Course time discrete uniformity: A class cannot arrange the same course on the same day. If there is a conflict, $f_4 = f_4 + 1$.

Here, "conflict" is used to describe adaptability. When the "conflict" is large, the corresponding adaptability is low; conversely, when the "conflict" is small, the adaptability is high.

Table 1: Genetic Coding

Course Number	Class Number	Teacher Number	Classroom	Time
6 Digits	1 Digits	2 Digits	1 Digits	2 Digits

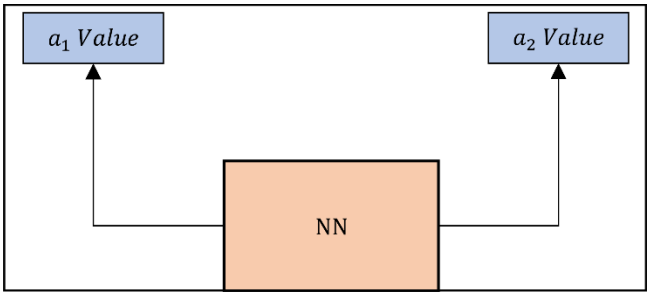


Figure 1: Artificial Neural Networks

3.4 Q-LEARNING ALGORITHM DESIGN

The Q-learning algorithm combines the ideas of the Monte Carlo algorithm and the dynamic programming algorithm and is therefore considered to be the most effective reinforcement learning algorithm. In the Q-learning algorithm, the construction of the Markov decision model (MDP) is a key issue. The MDP model generally consists of four elements, namely the state set, the reward function, the action set, and the policy function. The Q-learning algorithm uses a Q-table table to record the behavior value of each operation, so the larger the operation or state space, the more storage is required. Since the state set selected for the class scheduling problem is too large and uncertain, this paper introduces the DQN algorithm. The key to the DQN algorithm is that an artificial neural network $q(s, a; \omega), s \in S, a \in A$ is used to replace the Q-table, that is, the action value function. The input of the network is state information, and the output is the value of each action, as shown in Figure 1. The DQN algorithm has strong robustness and can handle continuous and uncertain state set problems well. The DQN algorithm combines three factors, namely, the previous experience state, the choice behavior, and the rewards given by the environment, to update the NN table. The update expression of the NN table is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max Q\left(s^{'}, a^{'}\right) - Q(s,a)\right] \tag{7}$$

Among them, Q(s, a) is the action value when the agent is in state s and action a; $Q(s^{'}, a^{'})$ is the action value when the agent is in state $s^{'}$ and action $a^{'}$; r is the immediate reward function; α is the learning rate; γ is the discount rate.

3.4.1 STATE SET DESIGN. In the class scheduling optimization problem, the objective function is to make the conflict value 0. In this algorithm, the conflict value is the fitness function of the group. In order to make a comprehensive evaluation of the performance of the group, this paper selects two indicators, namely the average fitness of the group and the best individual fitness, to divide the state set. Let $G(x_i^t)$ be the fitness function of the ith individual at the tth iteration, then at the tth iteration, the average fitness function G^t of the group can be determined as:

$$G^t = \frac{\sum_{i=1}^{N} G\left(x_i^t\right)}{N} \tag{8}$$

Among them, x_i^t represents the i-th individual at the t-th iteration, and N is the number of individuals in the population.

The fitness function M^t of the smallest individual in the population at the t-th generation is defined as:

$$M^t = \min G\left(x_i^t\right) \tag{9}$$

Then the population state can be represented by the vector S as:

$$S = \left(G^t, M^t\right) \tag{10}$$

3.4.2 SETTING THE ACTION SET AND REWARD FUNCTION. In each iteration, the agent will choose different actions to obtain the appropriate crossover probability P_c and mutation probability P_m. In terms of crossover probability, the interval of 0.4~0.9 is generally selected, and it is divided into 10 intervals with an interval of 0.05; in terms of mutation probability, the value is generally between 0.01 ~ 0.21. This paper divides it into 4 intervals with an interval of 0.05. Then the action set in the MDP model proposed in this paper is the Cartesian product of the crossover probability set and the mutation probability set, with a total of 40 intervals. The genetic algorithm determines the values of P_c and P_m according to the action value m, as shown in Figure 2.

Next, select an appropriate reward function to evaluate the crossover probability and mutation probability selection. This paper defines the reward function r as:

$$r = \frac{\sum_{i=1}^{N} G\left(x_i^t\right) - \sum_{i=1}^{N} G\left(x_i^{t-1}\right)}{\sum_{i=1}^{N} G\left(x_i^{t-1}\right)} \tag{11}$$

Among them, $G(x_i^{t-1})$ represents the fitness function of the i-th individual in the previous generation. It can be seen from the calculation formula that if the average fitness function of the current individual is better than that of the previous generation, that is, when r is a positive value, the agent is rewarded, indicating that the current P_c and P_m are valid, otherwise they are invalid.

3.4.3 ACTION SELECTION STRATEGY. Selecting reinforcement learning is the most basic work. The most frequently used ϵ greedy strategy is the action selection strategy. The ε − greedy strategy achieves a balance between use and search, in which the behaviour corresponding to the largest behaviour value function is selected, and other non-optimal behaviours are selected with a certain probability.

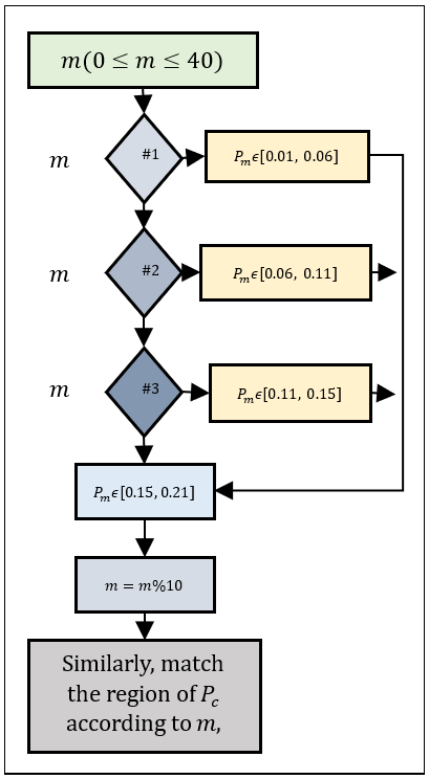


Figure 2: Flowchart for Determination of P_c and P_m

3.5 DESIGN OF SELF-LEARNING CLASS SCHEDULING GENETIC

The core of the basic genetic algorithm is the crossover and mutation operation, which has a very critical impact, and the main parameters of the crossover and mutation operation are P_c and P_m. In the process of genetic algorithm genetic calculation, if the valuesof P_c and P_m are too large, it will lead to slow calculation speed and difficulty in finding the optimal solution; if the values of P_c and P_m are too small, it will be difficult to obtain new individuals. The advantage of reinforcement learning is that it can effectively discover the internal structure of the population, deeply explore the interactive information between individuals, and use the changes in environmental states to determine the optimal parameters and achieve the optimal decision. In addition, the parameters can be adjusted continuously through evaluation feedback to obtain better results. At the same time, in order to solve the problem of too large an operation state set, this paper introduces the DQN algorithm to dynamically adjust the P_c and P_m values in the genetic algorithm in the hope of improving the global optimization ability of the algorithm.

DQN deep reinforcement learning adjustment of Pc and Pm can be roughly divided into four steps: First, the agent collects the state's in the environment, evaluates it, and calculates the fitness to determine the current state of the environment; secondly, using the selected strategy ε – greedy, the agent selects action a according to the corresponding value calculated by the neural network; then, according to the current state s and the strategy The action selected by ε – greedy is executed once for genetic evolution operation. At this time, the state of the population changes to s', and feedback is given to it, and reward is given. If the reward is positive, the action selection of the genetic algorithm will become stronger, and if the reward is negative, its selection will become weaker; finally, the operation s, the selected action a, the reward r and the operation s' are stored in the memory pool, and then DQN learning is selectively performed, and then the agent reselects the action to execute. On this basis, according to the previous training results, the deep reinforcement process is activated, so P_c and P_m The action selection will be optimized. Based on the above series of analysis, the pseudo code of the GA-DRL algorithm is shown in Algorithm 1.

Algorithm 1 Self-learning genetic algorithm based on deep reinforcement learning

```
5) Input: teachers.csv, courses.csv, training plans.csv
6) Output: course arrangement for each class.xls
1 schedules, courses, teachers & –
init (''teachers. csv'','' courses. csv'','' training plans. csv'');
2. init GeneticPopulation, DQN;
3. while n < MaxEvolution
4    a – choose_action(s);
5    s', r, result < step (s, a);
6    NN& – store_transition (s, a, s', r);
7   if n < 200 and n%5 = 0 then
8     DQN. learn( );
9      s = s';
10      if result ≠ None then
11    end;
12 end while
```

In the process of GA-DRL algorithm, in each iteration of genetic algorithm, the agent will select actions according to the strategy ϵ-greedy, that is, select the crossover probability P_c and mutation probability P_m. The algorithm in this paper is based on the internal state of genetic algorithm, and dynamically adjusts the mutation probability and crossover probability parameters of genetic algorithm through deep reinforcement learning, that is, realizes the process of self-learning of genetic algorithm.

The computational complexity of the GA-DRL algorithm mainly comes from the selection, crossover, mutation and other operations in the genetic algorithm and the update operation of the learning neural network. Therefore, the total computational time complexity (T) of GA-DRL can be expressed as:

$$T = O((T * N) + (B * E * N)) \tag{12}$$

Wherein, T is the number of iterations, N is the population size, B is the number of samples each time the NN table is updated, E is the number of training iterations, and N is the number of parameters of the neural network.

4 EXPERIMENTAL SETUP

4.1 EXPERIMENTAL DESIGN

The experiment utilises the class scheduling data, which is configured as follows: There are a total of 5 classes, each taught by 23

teachers, and held in 10 classrooms. The curriculum consists of 15 courses, which are scheduled from Monday to Friday, with 6 classes per day. The GA-DRL algorithm and the classic genetic algorithm are utilised as reference points and contrasted with the enhanced genetic algorithm mentioned in the literature [26].This study proposes various assessment measures to evaluate the correctness and efficiency of the class scheduling algorithm: feasibility, number of iterations needed for success, success probability, optimal fitness, convergence speed, and stability. Whether the algorithm's solution fits all restrictions is feasibility. Class scheduling requires the same number of iterations as the algorithm to find the optimum answer. Given the same test sample, the chance of success is class scheduling success divided by total tests. The conflict value—the individual with the lowest fitness in the algorithm-generated population—is optimal fitness. Quality of the algorithm is shown by this indicator. Algorithm quality depends on optimal solution fitness. Higher fitness suggests lesser quality, while lower fitness indicates higher quality. How quickly optimal fitness declines with each evolutionary generation is convergence speed. Stability is whether the method yields the same results in numerous runs with the same test sample.

Given that the feasibility conditions are met, when the work grows, the number of iterations and class scheduling method success must be assessed. Increasing the number of iterations improves algorithm accuracy, whereas decreasing it may decrease it. Success is positively correlated with algorithm accuracy. Accuracy might drop when success chances drop. The convergence speed and stability of the Intelligent course scheduling algorithm should also be considered. Higher convergence speeds indicate better performance, whereas lower speeds may indicate worse performance. Algorithm stability is its consistency and reliability. Lower stability may decrease algorithm performance, while higher stability improves it.

4.2 EXPERIMENTAL ENVIRONMENT

In order to test the effectiveness of the automatic class scheduling algorithm proposed in this paper, a common genetic algorithm (denoted as GA) and an improved genetic algorithm in the literature [27] (denoted as improved genetic algorithm [28]) were selected for comparison. The three algorithms use the same chromosome encoding scheme and have the same fitness function. The experiment was conducted on a computer with a 2.50GHz processor and 16GB RAM, and the operating system is Win11. The programming tool is PyCharm Community Edition 2022, and the programming language is Python. In addition, the relevant parameters in the GA-DRL algorithm are set as follows: learning rate $\alpha = 0.01$, discount rate $\gamma = 0.9$, greed rate $\varepsilon = 0.9$, memory capacity MS = 500, the step of updating the Q reality network parameters RTI = 200, the number of samples taken from the memory each time BS = 32, and the crossover and mutation probabilities are dynamically adjusted using DQN.

4.3 EXPERIMENTAL RESULTS

In order to test and evaluate the accuracy of the GA-DRL algorithm, 10 simulation tests were performed for each of the five cases with the number of training tasks being 45, 50, 55, 60, and 65, and their average values were taken as the experimental data results. In addition, in order to detect the performance of the GA-DRL algorithm, this paper compares the GA-DRL algorithm with the GA algorithm in terms of class scheduling time, number of class scheduling iterations, and success rate. The results are listed in Table 2.

4.3.1 ALGORITHM ACCURACY. First, we can find that no class offers two courses at the same time; second, we can find that the same classroom is not used at the same time; finally, we can find that the same teacher is not teaching at the same time. In addition, we tested the results of more than ten successful runs of the GA-DRL algorithm and found that they are all correct. Therefore, we can conclude that the results of the GA-DRL algorithm are reliable.

Table 2 indicates that the GA-DRL algorithm schedules classes slower than the GA and enhanced genetic algorithms [29] when training task load is minimal. GA algorithm solution time grows with training task count, to the point where no result can be acquired within a specific timeframe. Compared to the GA method, the enhanced genetic algorithm [30] and the GA-DRL algorithm schedule classes faster as training tasks rise. This greatly improves class scheduling, with GA-DRL being the most efficient. It takes longer to refresh the memory pool and choose mutation probability and crossover probability parameters from the NN table per iteration. Thus, the technique takes longer than the upgraded genetic algorithm [31]. With more training tasks, the GA-DRL algorithm needs fewer iterations to schedule. As training tasks increase, the GA-DRL algorithm scheduling test succeeds consistently.

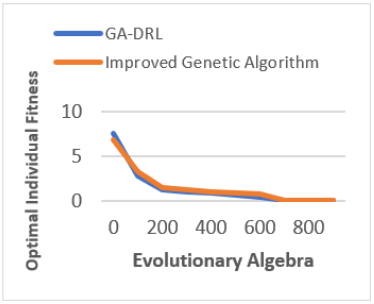
4.3.2 ALGORITHM PERFORMANCE. To further verify the performance of GA-DRL, this paper selected the optimal fitness value and the number of iterations required for successful operation of some training task scales, and statistically analyzed the data of the successful tests of the improved genetic algorithm [32] and GA-DRL. Their average values were used as the experimental data results, and the optimal fitness value of the population was recorded once every 100 generations of evolution. Finally, the average values of these data were used as the experimental results of fitness value and the number of iterations required for operation. Under different scales (S = 55, 60, 65), the results of the relationship between the optimal individual fitness value and the number of iterations of the two algorithms are shown in Figure 3.

5 CONCLUSION

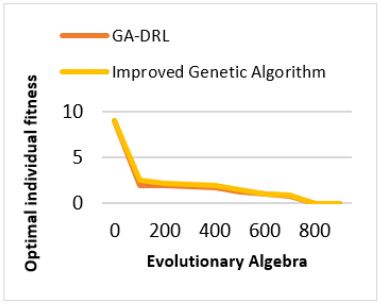
This paper presents a genetic algorithm for class scheduling called DRL-GA that uses deep reinforcement learning to learn how to schedule classes automatically. It then goes on to do experiments on training assignments of varying sizes. The algorithm's strong search performance and ability to automatically Once the Intelligent course scheduling procedure is finished, the results outperform those of the standard genetic algorithm in terms of optimisation performance and accuracy. It can guarantee that the results of Intelligent course scheduling are conflict-free while drastically reducing the number of iterations, which is useful for the teaching job done by colleges and institutions. More trustworthy and precise data backing. Topics including aircraft route planning, ship scheduling, and examination room organisation are also within the algorithm's purview. Also,

Table 2: Results of the Two Algorithms

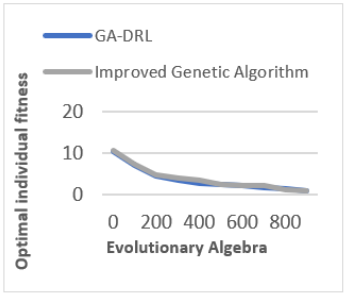
TrainingTask/pc	GA			GA-DRL		
	Time/s	Iterations	Success Rate	Time/s	Iterations	Success Rate
45	0.449	7	1	1.929	7	1
50	20.446	384	0.7	19.286	133	1
55	31.244	509	0.4	43.093	275	1
60	107.08	1530	0.1	66.205	347	0.9
65			0	166.59	853	0.8



(a) Convergence Curves of the Algorithm when the Number of Training Tasks S = 55



(b)Convergence Curves of the Algorithm when the Number of Training Tasks S = 60



(c) Convergence Curves of the Algorithm when the Number of Training Tasks S = 65

Figure 3: Convergence Curves of the Algorithm

there is just one soft constraint and three hard constraints that are necessary to obtain the algorithm design. Optimisation of indicators of curriculum satisfaction is missing from satisfactory solutions. Research will go on to the next phase after adding further soft restrictions, satisfying additional soft constraints while maintaining curriculum correctness, and identifying solutions that better match the curriculum as it actually exists.

References

[1] S. B. Prathiba, G. Raja, K. Dev, N. Kumar and M. Guizani, "A Hybrid Deep Reinforcement Learning For Autonomous Vehicles Smart-Platooning," in IEEE Transactions on Vehicular Technology, vol. 70, no. 12, pp. 13340-13350, Dec. 2021, doi: 10.1109/TVT.2021.3122257.

[2] W. Liu *et al.*, "Hybridization of Evolutionary Algorithm and Deep Reinforcement Learning for Multiobjective Orienteering Optimization," in IEEE Transactions on Evolutionary Computation, vol. 27, no. 5, pp. 1260-1274, Oct. 2023, doi: 10.1109/TEVC.2022.3199045.

[3] A. Zolfagharian, M. Abdellatif, L. C. Briand, M. Bagherzadeh and R. S, "A Search-Based Testing Approach for Deep Reinforcement Learning Agents," in IEEE Transactions on Software Engineering, vol. 49, no. 7, pp. 3715-3735, July 2023, doi: 10.1109/TSE.2023.3269804.

[4] E. Y. Keat *et al.*, "Multiobjective Deep Reinforcement Learning for Recommendation Systems," in IEEE Access, vol. 10, pp. 65011-65027, 2022, doi: 10.1109/ACCESS.2022.3181164.

[5] Bayi Cheng, Tao Xie, Lingjun Wang, Qi Tan, Xiongfei Cao, Deep reinforcement learning driven cost minimization for batch order scheduling in robotic mobile fulfillment systems Expert Systems with Applications, Volume 255, Part C, 2024, 124589, ISSN 0957-4174.

[6] H. Yu and X. Zhao, "Deep Reinforcement Learning With Reward Design for Quantum Control," in IEEE Transactions on Artificial Intelligence, vol. 5, no. 3, pp. 1087-1101, March 2024, doi: 10.1109/TAI.2022.3225256.

[7] L. Shi, S. Li, Q. Zheng, M. Yao and G. Pan, "Efficient Novelty Search Through Deep Reinforcement Learning," in IEEE Access, vol. 8, pp. 128809-128818, 2020, doi: 10.1109/ACCESS.2020.3008735.

[8] S. Y. Luis, D. G. Reina and S. L. T. Marín, "A Deep Reinforcement Learning Approach for the Patrolling Problem of Water Resources Through Autonomous Surface Vehicles: The Ypacarai Lake Case," in IEEE Access, vol. 8, pp. 204076-204093, 2020, doi: 10.1109/ACCESS.2020.3036938.

[9] Z. Ling, Y. Zhang and X. Chen, "A Deep Reinforcement Learning Based Real-Time Solution Policy for the Traveling Salesman Problem," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 6, pp. 5871-5882, June 2023, doi: 10.1109/TITS.2023.3256563.

[10] Zhiwei Liu, Zhaogang Shu, Shuwu Chen, Yiwen Zhong, Jiaxiang Lin, Low-latency Virtual Network function Scheduling Algorithm Based on Deep Reinforcement Learning, Computer Networks, Volume 246, 2024, 110418, ISSN 1389-1286.

[11] B. Gašperov and Z. Kostanjčar, "Market Making With Signals Through Deep Reinforcement Learning," in IEEE Access, vol. 9, pp. 61611-61622, 2021, doi: 10.1109/ACCESS.2021.3074782.

[12] Q. Wei, L. Wang, Y. Liu and M. M. Polycarpou, "Optimal Elevator Group Control via Deep Asynchronous Actor–Critic Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 12, pp. 5245-5256, Dec. 2020, doi: 10.1109/TNNLS.2020.2965208.

[13] N. K. Akrasi-Mensah *et al.*, "Adaptive Storage Optimization Scheme for Blockchain-IIoT Applications Using Deep Reinforcement Learning," in IEEE Access, vol. 11, pp. 1372-1385, 2023, doi: 10.1109/ACCESS.2022.3233474.

[14] X. Qu, Y. -S. Ong and A. Gupta, "Frame-Correlation Transfers Trigger Economical Attacks on Deep Reinforcement Learning Policies," in IEEE Transactions on Cybernetics, vol. 52, no. 8, pp. 7577-7590, Aug. 2022, doi: 10.1109/TCYB.2020.3041265.

[15] H. Yan, Z. Cui, X. Chen and X. Ma, "Distributed Multiagent Deep Reinforcement Learning for Multiline Dynamic Bus Timetable Optimization," in IEEE Transactions on Industrial Informatics, vol. 19, no. 1, pp. 469-479, Jan. 2023, doi: 10.1109/TII.2022.3158651.

[16] C. Wu *et al.*, "UAV Autonomous Target Search Based on Deep Reinforcement Learning in Complex Disaster Scene," in IEEE Access, vol. 7, pp. 117227-117245, 2019, doi: 10.1109/ACCESS.2019.2933002.

[17] Q. Wei, Y. Yan, J. Zhang, J. Xiao and C. Wang, "A Self-Attention-Based Deep Reinforcement Learning Approach for AGV Dispatching Systems," in IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 6, pp. 7911-7922, June 2024, doi: 10.1109/TNNLS.2022.3222206.

[18] M. Moran and G. Gordon, "Deep Curious Feature Selection: A Recurrent, Intrinsic-Reward Reinforcement Learning Approach to Feature Selection," in IEEE Transactions on Artificial Intelligence, vol. 5, no. 3, pp. 1174-1184, March 2024, doi: 10.1109/TAI.2023.3282564.

[19] N. Di Cicco *et al.*, "On Deep Reinforcement Learning for Static Routing and Wavelength Assignment," in IEEE Journal of Selected Topics in Quantum Electronics, vol. 28, no. 4: Mach. Learn. in Photon. Commun. and Meas. Syst., pp.

1-12, July-Aug. 2022, Art no. 3600112, doi: 10.1109/JSTQE.2022.3151323.
[20] V. A. R, M. G, A. I, M. Guru Brahmam, B. Balusamy and F. Benedetto, "Quantum Channel Optimization: Integrating Quantum-Inspired Machine Learning With Genetic Adaptive Strategies," in IEEE Access, vol. 12, pp. 80397-80417, 2024, doi: 10.1109/ACCESS.2024.3410147.
[21] Jiliang Luo, Sijia Yi, Zexuan Lin, Hongbin Zhang, Jiazhong Zhou, Petri-net-based deep reinforcement learning for real-time scheduling of automated manufacturing systems, Journal of Manufacturing Systems, Volume 74, 2024, Pages 995-1008, ISSN 0278-6125.
[22] F. Grumbach, N. E. A. Badr, P. Reusch and S. Trojahn, "A Memetic Algorithm With Reinforcement Learning for Sociotechnical Production Scheduling," in IEEE Access, vol. 11, pp. 68760-68775, 2023, doi: 10.1109/ACCESS.2023.3292548.
[23] X. Hu, S. Liu, R. Chen, W. Wang and C. Wang, "A Deep Reinforcement Learning-Based Framework for Dynamic Resource Allocation in Multibeam Satellite Systems," in IEEE Communications Letters, vol. 22, no. 8, pp. 1612-1615, Aug. 2018, doi: 10.1109/LCOMM.2018.2844243.
[24] J. Liao, G. Yang, S. Zhang, F. Zhang and C. Gong, "A Deep Reinforcement Learning Approach for the Energy-Aimed Train Timetable Rescheduling Problem Under Disturbances," in IEEE Transactions on Transportation Electrification, vol. 7, no. 4, pp. 3096-3109, Dec. 2021, doi: 10.1109/TTE.2021.3075462.
[25] H. Ma, D. Dong, S. X. Ding and C. Chen, "Curriculum-Based Deep Reinforcement Learning for Quantum Control," in IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 11, pp. 8852-8865, Nov. 2023, doi: 10.1109/TNNLS.2022.3153502.
[26] S. Luo, L. Zhang and Y. Fan, "Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning," in IEEE Transactions on Automation Science and Engineering, vol. 19, no. 4, pp. 3020-3038, Oct. 2022, doi: 10.1109/TASE.2021.3104716.
[27] A. Khan, C. Midha and D. Lowther, "Reinforcement Learning for Topology Optimization of a Synchronous Reluctance Motor," in IEEE Transactions on Magnetics, vol. 58, no. 9, pp. 1-4, Sept. 2022, Art no. 8206204, doi: 10.1109/TMAG.2022.3184246.
[28] L. Huang, C. -D. Wang, H. -Y. Chao, J. -H. Lai and P. S. Yu, "A Score Prediction Approach for Optional Course Recommendation via Cross-User-Domain Collaborative Filtering," in IEEE Access, vol. 7, pp. 19550-19563, 2019, doi: 10.1109/ACCESS.2019.2897979.
[29] Lorenzo Tiacci, Andrea Rossi, A discrete event simulator to implement deep reinforcement learning for the dynamic flexible job shop scheduling problem, Simulation Modelling Practice and Theory, Volume 134, 2024, 102948, ISSN 1569-190X.
[30] Tao Liang, Lulu Chai, Jianxin Tan, Yanwei Jing, LiangnianLv, Dynamic optimization of an integrated energy system with carbon capture and power-to-gas interconnection: A deep reinforcement learning-based scheduling strategy, Applied Energy, Volume 367, 2024, 123390, ISSN 0306-2619.
[31] Chunlin Hu, Donghe Li, Weichun Zhao, Huan Xi, Deep reinforcement learning-based scheduling for integrated energy system utilizing retired electric vehicle battery energy storage, Journal of Energy Storage, Volume 97, Part A, 2024, 112774, ISSN 2352-152X.
[32] Chen-Fu Chien, Yu-Bin Lan, Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for Industry 3.5 smart production, Computers & Industrial Engineering, Volume 162, 2021, 107782, ISSN 0360-8352.