Received: 14 March 2024 | Revised: 13 May 2024 | Accepted: 14 May 2024

The Journal of Engineering

DOI: 10.1049/tje2.12393

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Combining reinforcement learning algorithm and genetic algorithm to solve the traveling salesman problem

Yaqi Ruan | **Weihong Cai** | **Jiaying Wang**

Department of Computer Science, Shantou University, Shantou, Guangdong, China

Correspondence
Weihong Cai, Department of Computer Science, Shantou University, Shantou, Guangdong, China.
Email: whcai@stu.edu.cn

Funding information
Science and Technology Planning Project of Guangdong Province, Grant/Award Number: 2019B010116001

Abstract

With the growing recognition of the unique advantages of reinforcement learning and genetic algorithms in addressing combinatorial optimization problems, this study aims to integrate these two methods to collectively tackle the classic combinatorial optimization challenge of the travelling salesman problem (TSP). The TSP stands as a quintessential combinatorial optimization challenge, tasked with determining the shortest path among designated cities. This paper introduces an innovative approach by amalgamating reinforcement learning's path selection prowess with genetic algorithms' global search strategy, aiming to uncover superior solutions in TSP. Specifically, the experiment employs a dual Q-learning algorithm within reinforcement learning to identify multiple optimal paths, serving as progenitors for the genetic algorithm to further enhance performance. The paper meticulously outlines the problem modelling process, elucidating TSP instance definitions, descriptions, and precise objective function definitions. Experimental findings underscore the substantial enhancements achievable in TSP optimization through this comprehensive approach, offering a fresh perspective and methodology for tackling combinatorial optimization challenges.

1 | INTRODUCTION

The travelling salesman problem (TSP) is a classic combinatorial optimization problem, the complexity and practical value of which have made finding efficient solutions a focal point of research in both academia and industry [1, 2]. TSP entails finding the shortest path that visits each city in a given set exactly once. This problem finds widespread applications in fields such as logistics planning [5], circuit design, and traffic route optimization [4, 6]. Moreover, its variants extend beyond the realm of the traveling salesman's route planning, spanning several domains including mathematics, computer science, operations research, genetics, engineering, and electronics [7]. Therefore, the quality of its solutions directly impacts the efficiency and cost-effectiveness of real-world problems.

Traditional approaches to solving the TSP involve various heuristic algorithms and exact solution methods, such as genetic algorithms, simulated annealing [8], dynamic programming [4], tabu search [3], and ant colony optimization [9], among others. However, as the problem scales up, these methods face challenges in computational efficiency and solution quality. To overcome these challenges, researchers have been exploring more innovative and efficient solutions. In recent years, reinforcement learning [10–12] and genetic algorithms [13, 14] have emerged as two distinct optimization methods that have shown excellent performance in solving combinatorial optimization problems.

Reinforcement learning effectively handles the uncertainty and dynamics of problems by learning through decision-making and interaction with the environment [15]. Meanwhile, genetic algorithms have garnered attention for their characteristics of global search and optimization [16]. However, research combining these two approaches is relatively scarce. In this context, this study aims to explore a novel method that integrates reinforcement learning and genetic algorithms to enhance the solution effectiveness of the TSP.

The fundamental concept of the approach outlined in the article is to employ reinforcement learning for acquiring a strategy for path selection, and then leveraging multiple optimal paths identified by reinforcement learning as the progenitors of

a genetic algorithm to refine the global solution further. This amalgamated method effectively circumvents the constraints of reinforcement learning in handling high-dimensional, nonlinear problems while furnishing a superior initial population for the genetic algorithm. This innovative strategy yields substantial performance enhancements on TSP and akin combinatorial optimization challenges, resulting in augmented solution quality and heightened efficiency. The main contributions of this paper are:

- Proposal of an innovative new algorithm: the fusion of reinforcement learning and genetic algorithms, leveraging the double Q learning algorithm for reinforcement learning and employing its outcomes as the genetic algorithm's initial population.
- Demonstration through comparative experimental results that the amalgamation of these two algorithms exhibits favourable efficiency and stability in resolving the TSP, surpassing the efficacy of using reinforcement learning or genetic algorithms individually.

In the subsequent sections, the article delves into the mathematical modelling of the problem, the rationale behind the selection of reinforcement learning and genetic algorithms, the intricacies of experimental design, and a detailed presentation of empirical results. Through this study, a novel approach to solving combinatorial optimization problems is proposed, and empirical evidence is provided to support the research direction of combining reinforcement learning and genetic algorithms.

2 | RELATED WORKS

The TSP has been a classic problem in the field of combinatorial optimization since the 20th century [17]. Traditional solution methods encompass various heuristic algorithms and exact solving techniques. Among them, heuristic algorithms such as ant colony optimization, genetic algorithms, and simulated annealing have achieved notable success in solving TSP [18–20]. These methods typically rely on search and optimization strategies but face challenges of computational complexity when dealing with large-scale problems.

In recent years, the application of reinforcement learning in combinatorial optimization has garnered widespread attention. Reinforcement learning, by learning from the environment and adjusting through reward signals, demonstrates superior performance in decision-making and path selection problems. Some researchers have attempted to use deep reinforcement learning to solve TSP, where neural networks are employed to learn policies for city visitation order [21]. However, due to the combinatorial optimization nature of TSP, deep reinforcement learning faces challenges in handling high-dimensional, nonlinear problems.

The combination of reinforcement learning and evolutionary algorithms has emerged as a novel direction of research in recent years [22, 23]. This fusion approach aims to leverage the strengths of both to enhance overall optimization performance. In some combinatorial optimization problems, reinforcement learning is utilized to provide local search capability while evolutionary algorithms are responsible for global search [24]. This combined method has made breakthrough progress in solving some complex, high-dimensional problems.

The integration of reinforcement learning and genetic algorithms holds potential advantages in solving combinatorial optimization problems. Some researchers propose to use reinforcement learning policy parameters as individuals in genetic algorithms, evolving these parameters through crossover and mutation operations to optimize overall performance [25]. The advantage of this fusion method lies in its ability to utilize both reinforcement learning's local search capability and genetic algorithms' global search capability, providing a more comprehensive perspective for problem-solving [26].

Existing research has indeed shown some success in leveraging the combination of reinforcement learning and evolutionary algorithms to tackle combinatorial optimization problems. However, it is worth noting that there have been relatively few endeavours specifically focusing on the travelling salesman problem (TSP) within this realm of study. This study proposes an innovative approach that combines reinforcement learning's path selection capability with genetic algorithms' global search strategy. Some researchers have compared the performance of three classical reinforcement learning algorithms, Q-learning, double Q-learning, and Sarsa, applied to the traveling salesman problem, concluding that double Q-learning is the superior reinforcement learning algorithm [29]. Therefore, by employing double Q-learning as the reinforcement learning algorithm and utilizing its outcomes as the initial population for genetic algorithms, the aim is to uncover superior solutions in the context of the TSP. This innovative approach holds the potential to inspire and facilitate enhancements in addressing TSP-related challenges.

2.1 | Problem definition

The TSP is a fundamental problem in computer science that aims to find the shortest possible path that visits every city $(C_1, C_2, \cdots, C_n)$ exactly once and returns to the starting city [27]. This problem can be modeled using an undirected weighted graph where the cities are represented as vertices and the roads as edges, with the distance of each road being the weight of the corresponding edge. The objective is to minimize the total weight of the edges in the path while ensuring that each vertex is visited exactly once.

TSPs can be classified into two types: symmetric TSPs and asymmetric TSPs [28].

In the case of the symmetric TSP, the distance between each pair of cities is equal, resulting in an undirected graph. The goal of the problem is to find a Hamiltonian cycle with the minimum weight in a complete weighted undirected graph.

On the other hand, in the asymmetric TSP, the distances between pairs of cities are not equal or there are one-way paths. Thus, the undirected graph transforms into a directed graph. Due to its asymmetry, ATSP is more challenging to solve. In

reality, the majority of TSPs are asymmetric, and therefore, ATSPs have more practical applications [30].

3 | METHOD

3.1 | Reinforcement learning

Double Q-learning was selected as the reinforcement learning algorithm for acquiring the strategy of path selection. Double Q-learning is a variant of Q-learning aimed at addressing the issue of overestimation in traditional Q-learning when estimating action value functions [31, 32]. The algorithm maintains two independent Q-value functions, each used to update the estimation of the other function, thereby enhancing the stability of the estimation.

The updated rules for double Q-learning are as follows:

1) Action selection: Use an ε-greedy policy based on the current state s and Q-value functions to select the next city;
2) Action execution: Move to the next city based on the selected action;
3) Reward calculation: Calculate the immediate reward based on the distance between cities;
4) Update Q-value functions: Use the update rules of double Q-learning to update the estimation of both Q-value functions.

The formulas are as follows:

$$Q_{t+1}^{A} = Q_t^{A}(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q^{B}(s',a') - Q_t^{A}(s,a)] \quad (1)$$

$$Q_{t+1}^{B} = Q_t^{B}(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q^{A}(s',a') - Q_t^{B}(s,a)] \quad (2)$$

There are two functions in double Q-learning, Q^A and Q^B (double Q-learning), and each function updates the next state with the value of the other Q function. When selecting an action, it is determined based on the average of Q^A and Q^B. The pseudo code of double Q-learning algorithm is presented in Algorithm 1.

3.2 | Genetic algorithm

To integrate the outcomes of reinforcement learning into genetic algorithms, the approach was to employ multiple optimal paths identified by reinforcement learning as the progenitors for genetic algorithms [33]. Each path can be represented as an individual, consisting of the order in which cities are visited.

In genetic algorithms, the following operations are employed for evolution:

1) Selection: The selection operation is responsible for choosing a certain number of individuals from the current population to serve as parents for the evolution process.

ALGORITHM 1 Double Q-learning

```
Set the parameters: α, γ and ε
Initialize the matrix Q^A(s, a) = 0, Q^B(s, a) = 0
Observe the state s
repeat
    Take action a(e.g.ε-greedy) based on Q^A, Q^B
    Receive immediate reward r(s, a),Observe the new state s'
    Choose (e.g.random)either UPDATE(A) or UPDATE(B)
    if UPDATE(A) then
        Update Q^A with Eq. (3)
    if UPDATE(B) then
        Update Q^B with Eq. (4)
    s = s'
until end;
```

The roulette wheel selection method is employed, where the probability of selecting an individual is directly proportional to its fitness. Consequently, individuals with higher fitness are more likely to be selected and retained for the next generation. The fitness function F used in the experiment is:

MY: We can try Elitist selection method too

$$F = 1/d_{ij} \quad (3)$$

d_{ij} is the distance between cities i and j.

2) Crossover: The crossover operation is responsible for generating new individuals by combining two parents. Experimental choice of single-point crossover method, where a random crossover point is selected in both parents, and the portion of the chromosome after the crossover point is exchanged between the two parents. This method preserves information from the parents while introducing new genetic variation.
3) Mutation: The mutation operation introduces randomness to increase the diversity of the search. The insertion mutation method is utilized in the approach, where two positions in an individual's chromosome are randomly selected, and the sequence of cities between these two positions is reversed. This mutation method retains existing city visitation information while introducing some randomness.

After the selection, crossover, and mutation operations, new individuals are generated, forming the next generation population. This new population serves as the starting point for the next round of evolution. Over successive generations of evolution, the paths within the genetic algorithm are expected to be gradually optimized, converging towards the global optimum solution (See Algorithm 2).

3.3 | Reinforcement learning algorithm combined with genetic algorithm

In this study an innovative approach combining the double Q-learning algorithm and a genetic algorithm is used to improve the efficiency and accuracy of the problem solution search. First,

ALGORITHM 2 Genetic algorithm

```
Set the parameters: Population Size,Number of Generations,Mutation Probability
Initialize populations based on population size
repeat
    Calculation of fitness based on the fitness function
    Selection operation
    if Apply Crossover operator then
        Perform crossover operations
    if Apply Mutation operator then
        Perform mutation operations
    if current number of generations >target number of generations then
        break;
until end;
```

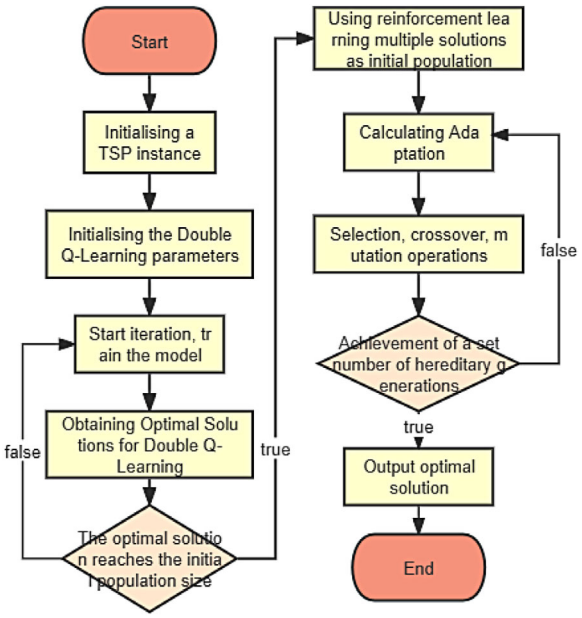

FIGURE 1 Algorithm flow chart.

a series of problem instances are solved using the double *Q*-learning algorithm. This algorithm is able to generate dozens of solutions with certain performance and accuracy through continuous learning and optimization. These solutions are finely trained and optimized to provide a solid foundation for the subsequent optimization process. These solutions optimized by double *Q*-learning are next used as the initial population of the genetic algorithm. The genetic algorithm continuously optimizes and evolves this population through operations such as selection, crossover, and mutation, thus gradually improving the quality and efficiency of the solutions. By combining these two algorithms it is possible to take full advantage of their respective strengths to solve complex optimization problems more effectively.

The specific flow of the combination algorithm is shown in Figure 1.

Through the outlined genetic algorithm steps, the results of reinforcement learning are integrated with evolutionary algorithms, with the objective of enhancing the solution efficacy for the TSP by amalgamating global search and local optimization strategies. In the ensuing sections, detailed descriptions of the experimental design and results analysis will be provided to substantiate the effectiveness of this approach.

TABLE 1 TSP instances.

Instance	*n*	Optimal solution
eil51	51	426
berlin52	52	7542
st70	70	675
eil76	76	538
kroA100	100	21282
pr107	107	44303

4 | EXPERIMENTAL RESULTS AND ANALYSIS

All algorithms were implemented using Python, and the experiments were conducted on an Intel Core i7 2.0 GHz CPU with 16 GB of memory.

4.1 | Experimental setup

4.1.1 | Data set

TSPLIB is a widely used database containing instances of TSP and related problems from various sources and of various types, such as TSP and ATSP. The library provides many instances of different complexity and gives their optimal solutions.

In this experiment, TSP instances from TSPLIB have been chosen for examination, encompassing primarily four TSPs, as delineated in Table 1. The main purpose of our selection of these instances is to observe the effectiveness of the algorithm for TSPs of different sizes. Hence, instances with a small number of cities, such as eil51 and berlin52, were selected, alongside instances featuring a larger number of city nodes, such as kroA100, comprising 100 city nodes.

By choosing instances of different sizes and complexities, the aim was to comprehensively evaluate the performance and effectiveness of the experimental algorithms in solving various types of TSPs. This approach aids in gaining a deeper understanding of the applicability and constraints of the algorithms, while also furnishing a valuable database for future research endeavours.

4.1.2 | Algorithm specifications

For the reinforcement learning (double *Q*-learning) algorithm, several parameters need to be configured to guarantee its effectiveness and stability. These parameters include:

- Learning rate(α): Controls the degree of importance given to historical experiences during each update. In Wang's study [29] it was proposed that the best results were achieved when the learning rate was chosen to be 0.01. So the learning rate for this experiment was set to 0.01.
- Discount factor(γ): Balances the importance of immediate rewards with future rewards. Also in Wang's study [29], it was found that the reinforcement learning algorithm performs best when the discount rate is between 0.01 and 0.3, so the discount rates chosen for this experiment were 0.01, 0.15, and 0.3.
- Exploration rate: Controls the balance between exploration and exploitation in the algorithm. The optimal combination of epsilon and reward function to use when applying reinforcement learning algorithms to different TSPs has been found in past research [29], so in example eil51 $\varepsilon = 1 - i/N$ (N is the total number of trainings, i is the current number of trainings), reward function $= 1/d_{ij}$. (d_{ij}. is the distance between two cities i and j). In example berlin52 $\varepsilon = -(i/N)^6 + 1$, reward function $= -(d_{ij})^2$. In example st70 ε is initially 1 and decreases by 0.1 for every (N/10) trainings, reward function $= 1/d_{ij}$. In example kroA100 $\varepsilon = 1 - i/N$, reward function $= 1/d_{ij}$.

Genetic algorithm parameter settings: For genetic algorithms, again some parameters need to be set to ensure the effectiveness and performance of the algorithm. These parameters include:

- The number of individuals contained in the population of the genetic algorithm is determined. When solving using the genetic algorithm alone, 100 routes are randomly generated and used as the initial population. When solving using a combination of reinforcement learning and genetic algorithms, the reinforcement learning algorithm was used to produce a set of 40 different optimal routes that were used as the initial population. The main reason for choosing different population sizes is that the initial population generated by the reinforcement learning algorithm is of high quality, so 40 populations are already effective and reduce the time cost.
- Number of generations: Specifies how many generations of evolution the algorithm will undergo. This value was set to 2,000 in the experiment in order to more clearly observe the effect of the number of evolutionary generations.
- Crossover probability: Controls the probability that a crossover operation occurs. This experiment does not specifically set the crossover probability, i.e. the crossover rate is 1 and the crossover operation is performed at each iteration.
- Mutation probability: Control the probability of the mutation operation occurring. After reviewing the literature and several experimental attempts, the mutation probability was set to 0.01.

4.1.3 | Experimental procedure

The specific flow of the experiment is as follows:

1) Apply genetic algorithm and reinforcement learning algorithm: solve six TSP instances using double Q-learning algorithm, Q-learning algorithm using improved exploration–exploitation strategy (enhanced Q-learning) [34], and genetic algorithm, respectively, and record the results of the solution and convergence.
2) Apply the combined algorithm of genetic algorithm and reinforcement learning algorithm: use multiple optimal paths selected by reinforcement learning as the initial population of the genetic algorithm, solve six TSP instances using the combined method, and record the solution results and convergence.
3) Evaluation and result analysis: compare the solution effects of different methods, including optimal path length, average path length and convergence speed. Analyse the optimal solution and convergence of the algorithms.
4) Summarize the conclusion: Statistical analysis and visualization of the experimental results, analyse the performance performance and advantages and disadvantages of different algorithms, and draw final conclusions.

4.2 | Experimental results and analysis

Here, the experimental results are compared and analysed in terms of the following indicators:

1) Optimal solution
2) Average solution
3) Convergence of genetic algorithms

4.2.1 | Based on the results of the optimal solution analysis

Initially, a detailed analysis will be conducted on the experimental outcomes derived from the independent application of reinforcement learning and genetic algorithms, followed by a comparison with the results obtained after implementing the combined approach of reinforcement learning and genetic algorithms. The comparative assessment will emphasize the effectiveness of these three algorithms in resolving TSP instances of varying sizes, with a particular emphasis on the optimal solutions attained. Through this comparative analysis, a preliminary evaluation of the strengths and weaknesses of each algorithm will be facilitated, offering insights into their performance across different problem types.

From the results shown in Table 2, it can be seen that the solutions obtained by the method combining reinforcement learning and genetic algorithm are of higher quality and closer to the optimal solutions of the TSP instances. The experimental results show that the average error of the optimal solutions for the six instances obtained by the combined genetic algorithm approach is about 7.1 percent, while the reinforcement learning algorithms alone (including double Q-learning and enhanced Q-learning) and the genetic algorithm alone are obviously less effective than the experimental algorithms. Specifically, the best

TABLE 2 Optimal solutions obtained by different algorithms applied to the TSP.

TSP instances (optimal)	Algorithm	Max	Inaccuracy
eil51(426)	Genetic algorithms+double Q-learning	445	4.3
	Double Q-learning	476	10.5
	Genetic algorithms	496	14.1
	Enhanced Q-learning	480	11.3
berlin52(7542)	Genetic algorithms+double Q-learning	7811	3.4
	Double Q-learning	8141	7.4
	Genetic algorithms	9074	16.9
	Enhanced Q-learning	9210	18.1
st70(675)	Genetic algorithms+double Q-learning	709	4.8
	Double Q-learning	771	12.4
	Genetic algorithms	852	20.8
	Enhanced Q-learning	848	20.4
eil76(538)	Genetic algorithms+double Q-learning	600	10.3
	Double Q-learning	673	20.1
	Genetic algorithms	728	26.1
	Enhanced Q-learning	709	24.1
kroA100(21282)	Genetic algorithms+double Q-learning	23696	10.2
	Double Q-learning	24547	13.3
	Genetic algorithms	25809	17.5
	Enhanced Q-learning	25500	16.5
pr107(44303)	Genetic algorithms+double Q-learning	48968	9.5
	Double Q-learning	51368	13.8
	Genetic algorithms	53581	17.3
	Enhanced Q-learning	52780	16.1

results are found in smaller instances, such as in example eil51, where the error is only 4.3 percent, and in example berlin52, where it is 3.4 percent. as the size of the instances increases, the error increases, from 4.8 percent in example st70 to 10.3 percent in example eil76. in larger instances, the error tends to be lower, and in larger instances, it tends to be lower. In larger instances the error stabilises again, with an error of 10.2 percent in instance kroA100 and decreasing to 9.5 percent in instance pr107.

Taking these factors into account, it can be concluded that the approach combining reinforcement learning with genetic algorithms demonstrates commendable operational efficiency. Although it may require longer runtimes in some cases, considering its capability to obtain higher-quality solutions and its shorter runtime compared to using the double *Q*-learning algorithm alone, this approach remains a viable option worth considering in practical applications.

These findings underscore the efficacy of integrating both optimization algorithms and underscore the superior performance attained through this fusion in tackling TSP instances. Through the integration of reinforcement learning with genetic algorithms, the complementary strengths of each method are leveraged, yielding noteworthy enhancements in solution quality and stability.

4.2.2 | Based on the results of the average solution analysis

Table 3 shows the average solutions of the four different methods for solving the traveller problem after ten runs. From the data in the table, it can be found that the genetic algorithm based on the improved double *Q*-learning algorithm for generating the initial population has a good stability, especially with respect to the genetic algorithm, because it can obtain lower average solutions in all six instances of the traveller's problem, and the larger the size of the instances, the more obvious the results achieved.

In the smallest instance, eil51, the performance of the combined reinforcement learning algorithm and genetic algorithm does not differ much from the other algorithms because of the small number of cities and their proximity; in instance berlin52, the effect starts to become noticeable due to the increase in the distance between its cities; in instance st70, the difference is also noticeable with the increase in the number of cities; and in the following three instances In the following three instances, the city size increases, and the distance between cities also increases, so the average solution of the combination of reinforcement learning algorithm and genetic algorithm performs better than the other three algorithms.

TABLE 3 Average solutions obtained by different algorithms applied to the TSP.

TSP instances	Algorithm	Avg
eil51	Genetic algorithms+double Q-learning	506
	Double Q-learning	512
	Genetic algorithms	561
	Enhanced Q-learning	565
berlin52	Genetic algorithms+double Q-learning	8,333
	Double Q-learning	8,700
	Genetic algorithms	10,100
	Enhanced Q-learning	10,027
st70	Genetic algorithms+double Q-learning	775
	Double Q-learning	865
	Genetic algorithms	944
	Enhanced Q-learning	930
eil76	Genetic algorithms+double Q-learning	619
	Double Q-learning	692
	Genetic algorithms	739
	Enhanced Q-learning	722
kroA100	Genetic algorithms+double Q-learning	26,858
	Double Q-learning	28.726
	Genetic algorithms	34.108
	Enhanced Q-learning	31.066
pr107	Genetic algorithms+double Q-learning	50.126
	Double Q-learning	58.114
	Genetic algorithms	67.759
	Enhanced Q-learning	59.096

The combined algorithm of reinforcement learning algorithm and genetic algorithm will use genetic algorithm to optimize the solution set of reinforcement learning compared to the reinforcement learning algorithm, therefore it can get better solutions. The initial population generated using the reinforcement learning algorithm is of much higher quality than the randomly generated initial population compared to the genetic algorithm, so using the experimental algorithm for the same number of iterations yields better iteration results.

4.2.3 | Analysis based on convergence

In this paper, the convergence of the genetic algorithm with the new algorithm is also analysed experimentally, and the convergence plots of the various algorithms show that when the double Q-learning algorithm + genetic algorithm is used to solve the traveller's problem, it can always produce a quality solution in the initial population of the genetic algorithm. Subsequently, as the number of genetic generations increases, the quality of the solution will further improve. In some instances, the optimal solution of its genetic algorithm may also not be as high quality as the initial solution of double Q-learning algorithm + genetic algorithm.

Figures 2–4 show the convergence curves of the six instances solved using the genetic algorithm and double Q-learning algorithm + genetic algorithm, respectively. As can be seen from the figures, the quality of the randomly generated initial population is very poor, and the quality of the initial population can be significantly improved by using the double Q-learning algorithm, and by iterating from a much lower initial population than that of the genetic algorithm at the very beginning of the iteration, it is possible to converge to the target value by a much shorter number of iterations. The performance is more obvious in the larger scale instances, and in the smallest scale instances eil51 and berlin52, the genetic algorithm can also achieve a solution close to that of double q-learning algorithm + genetic algorithm. And in instance st70, it is still the same situation, but in instance eil76, the gap between the two solution methods starts to become obvious, the difference is so large that the solution obtained using the genetic algorithm is worse than the solution of reinforcement learning, i.e. the quality of the initial population. In the largest instances, kroA100 and pr107, the genetic algorithm converged much less than the double Q-learning algorithm + genetic algorithm, further illustrating the advantages of combining reinforcement learning algorithms and genetic algorithms and their efficient convergence rates.

These results indicate that the combined method of double Q-learning algorithm + genetic algorithm converges significantly faster than the traditional genetic algorithm in solving the traveller problem. The experimental algorithms use reinforcement learning to improve the initial population of the genetic algorithm so that there are better solutions in the initial population, which speeds up the convergence process of the algorithms and enables them to find high-quality solutions with a shorter number of iterations. This provides strong support for the efficiency of the experimental algorithm in practical applications, and also highlights the superiority of the combination of reinforcement learning and genetic algorithm.

Finally, based on the experimental results, the optimal roadmap for each instance of the traveller problem is obtained, as shown in Figures 5–10. These roadmaps show the optimal paths derived by the experimental algorithms in solving traveller problems of different sizes, thus demonstrating the performance and advantages of the algorithms in different instances.

By examining these optimal roadmaps, one can visually assess the effectiveness of the experimental algorithms in solving the traveller's problem and identify the differences between different instances. These visual representations not only provide a clear display of the solution, but also help to gain a deeper understanding of the algorithm's solution process and path planning strategy. These optimization roadmaps provide an important reference for comprehensively evaluating the performance and effectiveness of experimental algorithms, and provide valuable information for further research and development.

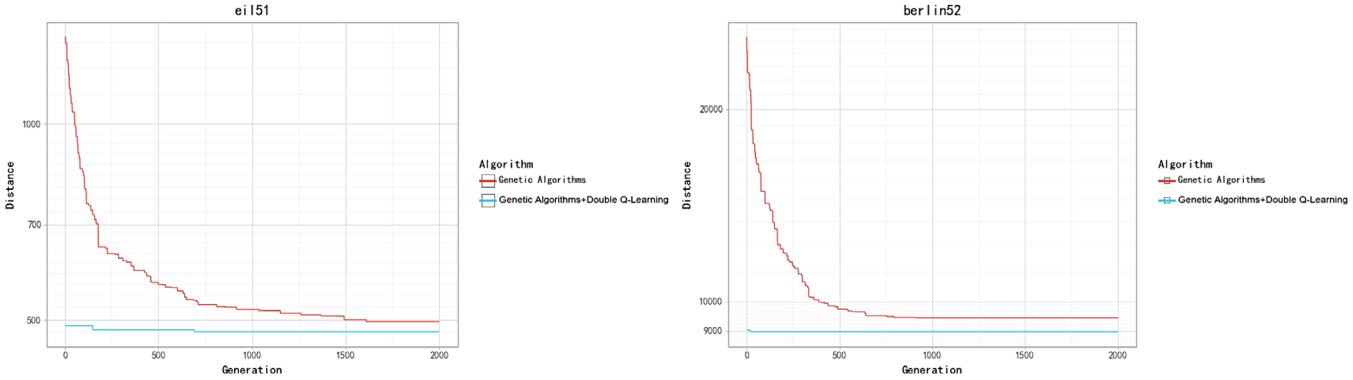

FIGURE 2 Convergence curves for instance eil51 and berlin52.

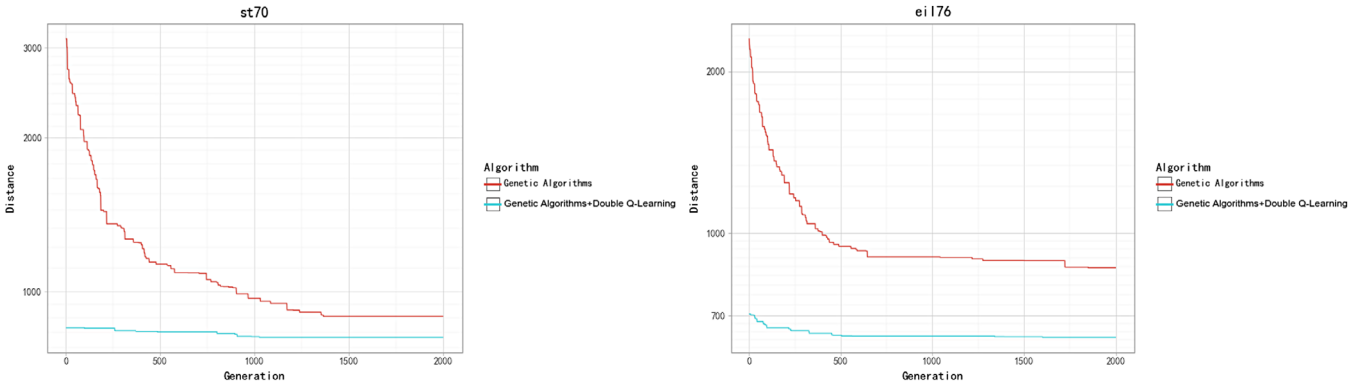

FIGURE 3 Convergence curves for instance st70 and eil76.

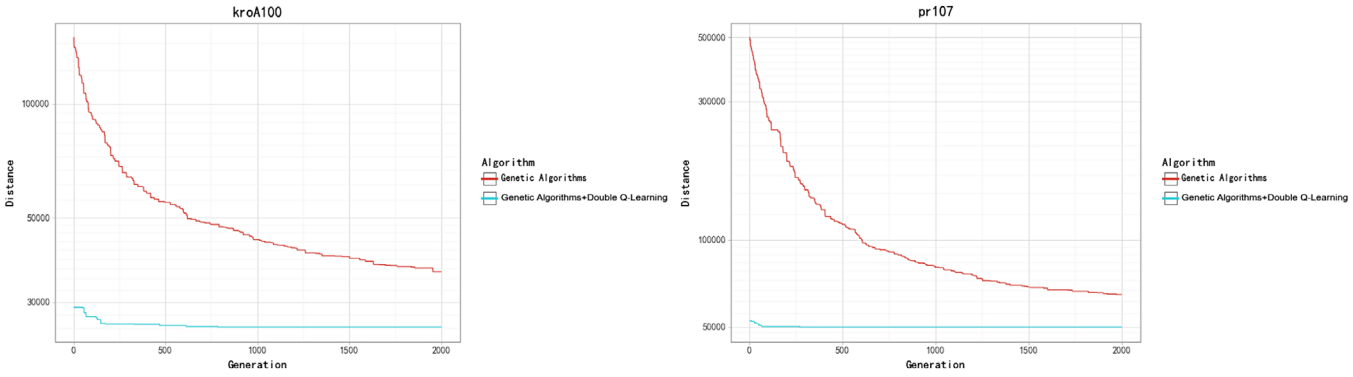

FIGURE 4 Convergence curves for instance kroA100 and pr107.

5 | CONCLUSION

In this paper, a new method combining the improved reinforcement learning algorithm and the genetic algorithm is proposed, and detailed experiments are designed to demonstrate the feasibility and effectiveness of this method, mainly through three aspects: optimal solution, average solution and convergence speed. By observing and analysing the experimental results, it can be obtained that the algorithm performs better than the improved Q-learning algorithm and the genetic algorithm in all six traveller problems, which is reflected in the fact that it can obtain more ideal optimal solutions, and also has a stable average solution and a faster convergence speed.

This finding highlights the potential of combining reinforcement learning algorithms with genetic algorithms, especially when it comes to solving the traveller's problem where it shows commendable solution quality. Although in some cases it may take longer to run and is not as efficient as the genetic algorithm, its stability and overall performance improvement is still satisfactory. Therefore, it can be concluded that combining

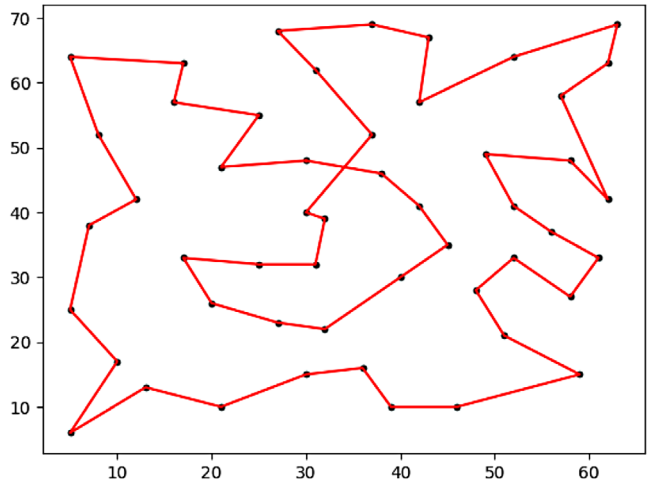

FIGURE 5 The best path to eil51.

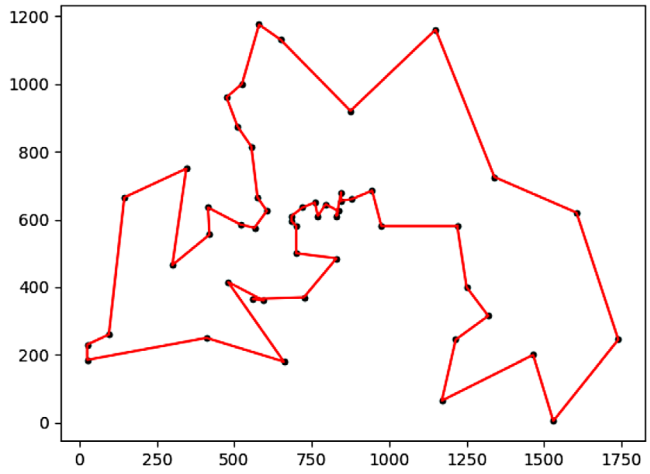

FIGURE 6 The best path to berlin52.

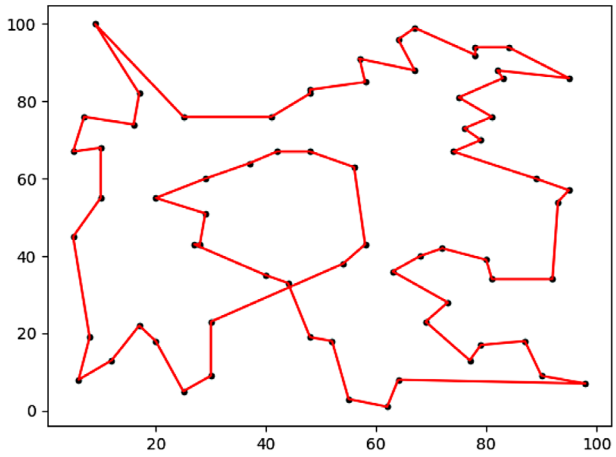

FIGURE 7 The best path to st70.

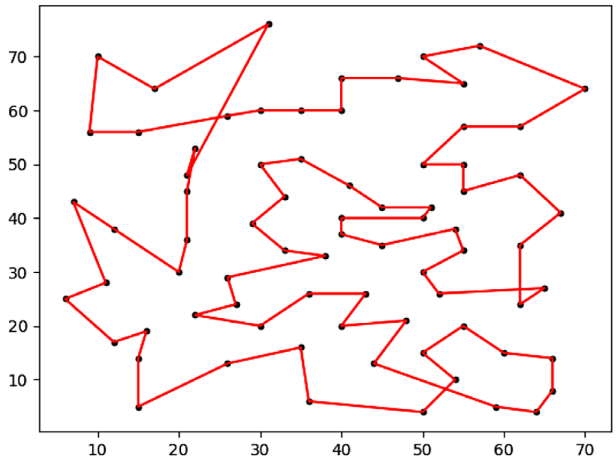

FIGURE 8 The best path to eil76.

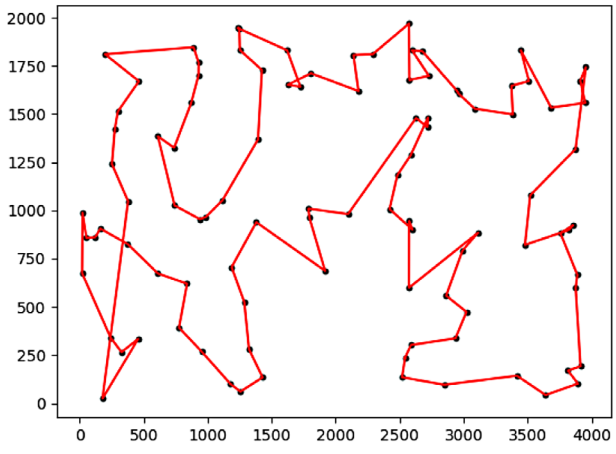

FIGURE 9 The best path to kroA100.

improved reinforcement learning algorithms with genetic algorithms is an area worthy of further research and exploration, and is expected to provide more effective solutions for real-world applications such as the traveller's problem. The findings of this paper emphasize the effectiveness of integrating two optimization algorithms and highlight the excellent performance obtained through this fusion in dealing with the traveller problem instance. By improving the fusion of the reinforcement learning algorithm with the genetic algorithm, the strengths of each approach are complemented, resulting in a significant improvement in the quality and stability of the solution.

Some problems or imperfections were still found in the experimental process, so some directions that can be improved in the follow-up work are proposed here: The two reinforcement learning algorithms chosen in this paper are value-based off-policy algorithms, you can try to use other types of reinforcement learning algorithms such as policy gradient, TD3 etc., and combine them with genetic algorithms, and observe whether their effects are better than the algorithms used in this paper.

In this paper, only six instances of the traveller problem from the TSPLIB library are selected for experimental comparison, and the new algorithms can be subsequently applied to larger scale problems or real-world problems or randomly generated problems.

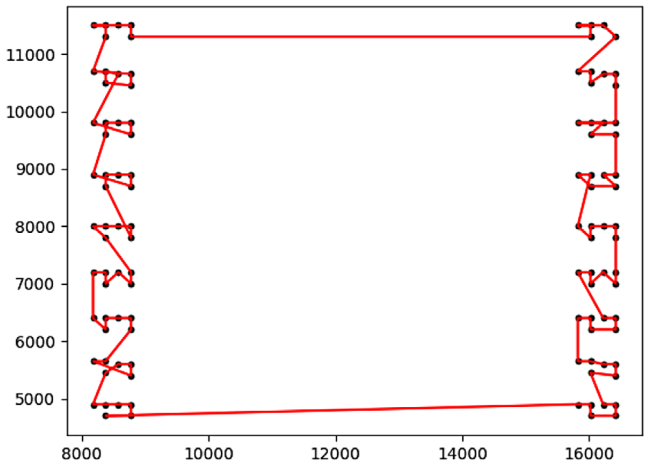

FIGURE 10 The best path to pr107.

The solution efficiency of the two algorithms proposed in the experiment has been improved, but it is still not optimal, and the main reason is that the running time for the reinforcement learning algorithm to generate the optimal initial population is too long, and we can try to improve the solution efficiency of the reinforcement learning algorithm from a different perspective, so that it can generate the initial population of the genetic algorithm faster to achieve the purpose of enhancing the efficiency of the algorithm.

AUTHOR CONTRIBUTIONS

Yaqi Ruan: Conceptualization; formal analysis; software; methodology; writing—original draft. **Weihong Cai**: Funding acquisition; resources; supervision; validation; writing—review and editing. **Jiaying Wang**: Methodology; writing—original draft; validation.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Yaqi Ruan https://orcid.org/0009-0006-3966-990X
Weihong Cai https://orcid.org/0000-0003-4510-341X

REFERENCES

1. Luke, S.: Essentials of Metaheuristics. vol. 2, Lulu, Raleigh (2013)
2. Davendra, D. (ed.): Traveling Salesman Problem: Theory and Applications. Intech Open, London (2010)
3. Osaba, E., et al.: Hybrid quantum computing-tabu search algorithm for partitioning problems: preliminary study on the traveling salesman problem. In: 2021 IEEE Congress on Evolutionary Computation (CEC), pp. 351–358. IEEE, Piscataway, NJ (2021)
4. Gharib, A., Jamal, B., Mohsine, C.: A performance comparison of PSO and GA applied to TSP. Int. J. Comput. Appl. 130(15), 34–39 (2015)
5. Sheng, W., et al.: Robot path planning for dimensional measurement in automotive manufacturing. J. Manuf. Sci. Eng. 127(2), 420–428 (2005)
6. Bertsimas, D.J., David, S.-L.: A new generation of vehicle routing research: robust algorithms, addressing uncertainty. Oper. Res. 44(2), 286–304 (1996)
7. Le, A.V., et al.: Coverage path planning using reinforcement learning-based TSP for hTetran–a polyabolo-inspired self-reconfigurable tiling robot. Sensors 21(8), 2577 (2021)
8. Halim, A.H., Ismail, I.: Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. Arch. Comput. Methods Eng. 26, 367–380 (2019)
9. Haroun, S.A., Benhra, J.: A performance comparison of GA and ACO applied to TSP. Int. J. Comput. Appl. 117(20), 28–35 (2015)
10. Zhang, Z., et al.: Solving dynamic traveling salesman problems with deep reinforcement learning. IEEE Trans. Neural Networks Learn. Syst. 34(4), 2119–2132 (2021)
11. Parasteh, S., et al.: A deep averaged reinforcement learning approach for the traveling salesman problem. In: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2514–2519. IEEE, Piscataway, NJ (2022)
12. Liu, W., et al.: Hybridization of evolutionary algorithm and deep reinforcement learning for multi-objective orienteering optimization. IEEE Trans. Evol. Comput. 27(5), 1260–1274 (2022)
13. Bye, R.T., et al.: A comparison of ga crossover and mutation methods for the traveling salesman problem. In: Innovations in Computational Intelligence and Computer Vision: Proceedings of ICICV 2020, pp. 529–542. Springer, Singapore (2021)
14. George, T., Amudha, T.: Genetic algorithm based multi-objective optimization framework to solve traveling salesman problem. In: Proceedings of the Advances in Computing and Intelligent Systems: ICACM 2019, pp. 141–151. Springer, Singapore (2020)
15. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Boston, MA (2018)
16. Rostami, M., Berahmand, K., Forouzandeh, S.: A novel community detection based genetic algorithm for feature selection. J. Big Data 8(1), 2 (2021)
17. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: a case study. Local Search Comb. Optim. 1997, 215–310 (1997)
18. Petr, S., et al.: Hybrid algorithm based on ant colony optimization and simulated annealing applied to the dynamic traveling salesman problem. Entropy 22(8), 884 (2020)
19. Halim, A.H., Ismail, I.: Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. Arch. Comput. Methods Eng. 26, 367–380 (2019)
20. Ezugwu, A.E.-S., Adewumi, A.O., Frîncu, M.E.: Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. Expert Syst. Appl. 77, 189–210 (2017)
21. Zhengxuan, L., Zhang, Y., Chen, X.: A deep reinforcement learning based real-time solution policy for the traveling salesman problem. IEEE Trans. Intell. Transp. Syst. 24(3), 5871–5882 (2023)
22. Drugan, M.M.: Reinforcement learning versus evolutionary computation: a survey on hybrid algorithms. Swarm Evol. Comput. 44, 228–246 (2019)
23. Radaideh, M.I., Shirvan, K.: Rule-based reinforcement learning methodology to inform evolutionary algorithms for constrained optimization of engineering applications. Knowl. Based Syst. 217, 106836 (2021)
24. Alipour, M.M., et al.: A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. Neural Comput. Appl. 30, 2935–2951 (2018)
25. Liu, H., et al.: NeuroCrossover: an intelligent genetic locus selection scheme for genetic algorithm using reinforcement learning. Appl. Soft Comput. 146, 110680 (2023)
26. Chen, R., et al.: A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Comput. Ind. Eng. 149, 106778 (2020)
27. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1(1), 53–66 (1997)
28. Matai, R., Singh, S.P., Mittal, M.L.: Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches. Intech Open, London (2010).

29. Wang, J., et al.: Reinforcement learning for the traveling salesman problem: performance comparison of three algorithms. J. Eng. 2023(9), e12303 (2023)
30. Hoffman, K.L., Padberg, M., Rinaldi, G.: Traveling salesman problem. Encycl. Oper. Res. Manage. Sci. 1, 1573–1578 (2013)
31. Hasselt, H.: Double Q-Learning. In: Advances in Neural Information Processing Systems, vol. 23, pp. 2613–2621. ACM, New York (2010).
32. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2094–2100. ACM, New York (2016)
33. Andrade, C.E., et al.: The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. Eur. J. Oper. Res. 289(1), 17–30 (2021)
34. El Jaghaoui, S., Elmiad, A.K., Lmah, A.B.: Enhancing the traveling salesman problem solutions with reinforcement learning: a variant exploration-exploitation approach beyond ε-greedy. In: 2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA), pp. 1–6. IEEE, Piscataway, NJ (2023)

How to cite this article: Ruan, Y., Cai, W., Wang, J.: Combining reinforcement learning algorithm and genetic algorithm to solve the traveling salesman problem. J. Eng. 2024, e12393 (2024).
https://doi.org/10.1049/tje2.12393