Journal of Artificial Intelligence Practice (2026) **DOI: 10.23977/jaip.2026.090104**
Clausius Scientific Press, Canada **ISSN 2371-8412 Vol. 9 Num. 1**

A Comparative Review of Reinforcement Learning and Traditional Algorithms in Typical Optimization Problems

Suyang Wu

School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China
1615079253@qq.com

Keywords: Reinforcement Learning; Greedy Algorithm; Genetic Algorithm; Traveling Salesman Problem; Job Shop Scheduling; Optimization Comparison

Abstract: Combinatorial optimization problems are widely present in fields such as logistics scheduling and manufacturing, among which the Traveling Salesman Problem (TSP) and Job Shop Scheduling Problem (JSSP) are two highly representative basic problems. As an emerging intelligent optimization method, reinforcement learning (RL) exhibits potential advantages in solving optimization problems due to its characteristic of learning through interaction with the environment; while traditional optimization algorithms such as greedy algorithms and genetic algorithms have formed mature solution frameworks after long-term development. Taking TSP and JSSP as research carriers, this paper systematically sorts out the differences in solution mechanisms and performance between reinforcement learning, greedy algorithms, and genetic algorithms from two core dimensions: solution speed and optimal solution quality. It analyzes the applicable scenarios of various algorithms in combination with existing research results, providing references for algorithm selection in optimization problems. Finally, the shortcomings of current research are summarized, and future research directions are prospected.

1. Introduction

In industrial production and daily life, a large number of problems can be attributed to combinatorial optimization problems. Such problems usually have the characteristics of a large solution space and complex constraints, and most of them are NP-hard problems, making it difficult to find the optimal solution in polynomial time through exact algorithms. The Traveling Salesman Problem (TSP) and Job Shop Scheduling Problem (JSSP) are classic problems in the field of combinatorial optimization, and their solution quality directly affects key indicators such as logistics distribution efficiency and production resource utilization rate.

To efficiently solve such optimization problems, academia and industry have proposed various algorithmic solutions. As a simple and intuitive heuristic algorithm, the greedy algorithm can quickly obtain feasible solutions by virtue of its characteristic of making locally optimal decisions, and is widely used in small-scale optimization problems; genetic algorithms, based on the theory of biological evolution, achieve global optimization through operations such as selection, crossover, and mutation, and have strong exploration capabilities for complex solution spaces; reinforcement learning, on the other hand, continuously interacts with the environment to learn the optimal strategy with the goal of maximizing cumulative rewards, and has the potential to handle dynamically uncertain environments.

In recent years, the application of reinforcement learning in optimization problems has gradually become a research hotspot, but traditional algorithms still dominate in many scenarios[3]. Existing research mostly focuses on the improvement of a single algorithm or the simple comparison between two algorithms, lacking a systematic sorting out of the three algorithms under a unified evaluation dimension. Based on this, this paper takes TSP and JSSP as specific research scenarios, deeply compares the performance differences between reinforcement learning, greedy algorithms, and genetic algorithms from two core dimensions of solution speed and optimal solution quality, clarifies the applicable conditions of various algorithms, and provides theoretical and practical references for algorithm selection in different scenarios.

2. Overview of Core Algorithm Principles

2.1. Greedy Algorithm

The core idea of the greedy algorithm is to select the locally optimal solution under the current state in each step of decision-making, and obtain the global feasible solution by gradually accumulating local optimal solutions. This algorithm does not need to consider the structure of the overall solution space, nor does it require complex iterative calculations, and only makes decisions based on preset greedy criteria (such as the shortest distance and the lowest cost). For example, in TSP, the greedy algorithm often adopts the "nearest neighbor" strategy, that is, starting from the current city, it prioritizes selecting the nearest unvisited city as the next destination until all cities are traversed and returns to the starting point.

The advantages of the greedy algorithm lie in its simple logic, easy implementation, and extremely fast solution speed. Its time complexity is usually polynomial, which can quickly respond to the solution needs of small-scale optimization problems. However, due to its "short-sighted" decision-making characteristic, it ignores the correlation between local optimal solutions and global optimal solutions, and often fails to obtain high-quality global optimal solutions. Even in some complex scenarios, the gap between the feasible solution obtained and the optimal solution is relatively large.

2.2. Genetic Algorithm

Genetic algorithm is a random search algorithm based on natural selection and biological evolution mechanisms. Its core process includes five links: initial population initialization, fitness function evaluation, selection, crossover, and mutation. First, a certain number of initial solutions are randomly generated to form a population, and each initial solution corresponds to a "chromosome"; second, a fitness function is defined to quantify the pros and cons of each chromosome, which serves as the basis for subsequent selection operations; the selection operation retains chromosomes with higher fitness in the population and eliminates individuals with lower fitness; the crossover operation generates new individuals by exchanging part of the genes of two chromosomes to achieve genetic recombination; the mutation operation randomly changes part of the genes of the chromosomes to increase population diversity and avoid the algorithm falling into local optimal solutions. Through multiple generations of iteration, the population gradually converges to the optimal solution or approximate optimal solution.

The core advantage of genetic algorithms is their strong global optimization capability, which can effectively explore complex solution spaces and has good versatility for various combinatorial optimization problems. However, this algorithm has the problem of slow solution speed. As the problem scale expands, the population size and the number of iterations need to be increased accordingly, leading to a significant increase in computational cost; at the same time, the algorithm performance is greatly affected by parameters such as crossover probability and mutation probability, and the parameter tuning process is cumbersome.

2.3. Reinforcement Learning

The core framework of reinforcement learning consists of five parts: Agent, Environment, State, Action, and Reward. The agent perceives the state of the environment, performs specific actions to change the environmental state, and obtains reward signals fed back by the environment. Its core goal is to learn an optimal strategy to maximize long-term cumulative rewards. In optimization problems, the solution space of the problem is usually mapped to the environmental state, the construction steps of the solution are mapped to the actions of the agent, and the quality of the solution is quantified as a reward signal.

Different from traditional optimization algorithms, reinforcement learning does not need to know the complete solution space structure of the problem in advance, and can independently learn the optimal strategy through interaction with the environment, thus having strong adaptive capabilities. For example, in TSP, reinforcement learning can take "the current city and the set of unvisited cities" as the state, "selecting the next city to visit" as the action, and "the negative value of the total path length" as the reward signal, and train the agent to learn the optimal path selection strategy through algorithms such as policy gradient[4]. However, reinforcement learning has problems such as complex training process and low sample efficiency. It requires a large number of interaction samples to converge to a stable strategy, and the training results are easily affected by parameters such as reward function design and learning rate.

3. Comparative Analysis of Algorithms in Typical Optimization Problems

3.1. Algorithm Comparison in Traveling Salesman Problem

The core goal of TSP is to find the shortest path that traverses all cities exactly once and finally returns to the starting point. Its solution space grows exponentially with the increase in the number of cities, making it a typical NP-hard problem. The following compares the performance of the three algorithms from two dimensions of solution speed and optimal solution quality in combination with existing research results.

In terms of solution speed, the greedy algorithm performs the best. Since it does not require iterative calculations and can complete path construction only through local optimal decisions, it can obtain feasible solutions almost instantaneously in scenarios with a small number of cities (such as 10-20 cities). Even when the number of cities increases to about 50, the solution time of the greedy algorithm remains at the millisecond level. The solution speed of the genetic algorithm is the second, but its solution time increases significantly with the increase in the number of cities and population size. For example, in the 50-city TSP problem, the average solution time of the genetic algorithm is about 7.77 seconds, which is much longer than that of the greedy algorithm. The solution process of reinforcement learning is divided into training phase and inference phase. The training phase requires a large number of interaction samples for strategy learning, which takes a long time; but the inference phase after training has extremely fast solution speed. In the 50-city TSP problem, the inference time of the algorithm based on deep reinforcement learning is only 0.022 seconds, which is even better than that of the greedy algorithm.

In terms of optimal solution quality, the genetic algorithm performs better than the greedy algorithm. The "short-sighted" decision-making of the greedy algorithm is likely to lead the path into a local optimum. For example, in the 50-city TSP problem, the average error of the path length obtained by the greedy algorithm can reach 29.55%. Through multi-generational iterative global search, the genetic algorithm can effectively jump out of local optimal solutions, and its average error can be controlled at a low level. If an improved genetic algorithm (such as the greedy genetic algorithm) is adopted, the quality of the solution can be further improved, making the path length about 5% better than that of the traditional genetic algorithm[2]. When fully trained, the optimal solution quality of reinforcement learning is close to that of the genetic algorithm. In the 50-city TSP problem, the average error is about 3.97%, which is slightly higher than the optimal level of the genetic algorithm but significantly better than that of the greedy algorithm; moreover, reinforcement learning has good generalization ability and can handle TSP problems of different scales without retraining. In the 100-city TSP problem, the error can still be controlled within 10%.

It should be noted that the optimal solution quality of reinforcement learning is highly dependent on the sufficiency of the training process. If the training samples are insufficient or the reward function design is unreasonable, the solution quality may be lower than that of the genetic algorithm. Although the solution quality of the greedy algorithm is poor, relying on its extremely fast solution speed, it still has an irreplaceable advantage in scenarios where the requirement for solution quality is not high but the requirement for response speed is extremely high (such as the temporary distribution path planning of emergency supplies).

3.2. Algorithm Comparison in Job Shop Scheduling Problem

The core goal of JSSP is to reasonably arrange the processing sequence and processing machines of workpieces under the constraints of limited production resources, so as to optimize the production objectives (such as minimizing the maximum completion time Makespan and minimizing the total production cost). This problem involves multiple constraint conditions such as workpieces, machines, and processes, and the complexity of the solution space increases significantly with the increase in the number of workpieces and machines. The following compares the performance of the three algorithms from two dimensions of solution speed and optimal solution quality.

In terms of solution speed, the greedy algorithm still has obvious advantages. For small-scale JSSP (such as 5 workpieces and 5 machines), the greedy algorithm can quickly determine the processing sequence through criteria such as "shortest processing time first", and the solution time is usually within seconds. The inference speed of reinforcement learning is the second. After training, the agent can quickly output the scheduling plan without iterative calculation; but the training phase needs to build a complex scheduling environment, which takes a long time. The solution speed of the genetic algorithm is the slowest. Due to the need to optimize the population through multiple generations of iteration, in large-scale JSSP (such as 50 workpieces and 10 machines), the number of iterations often needs to reach hundreds or even thousands of times, leading to a significant increase in solution time.

In terms of optimal solution quality, genetic algorithms and reinforcement learning have their own advantages. Through their global search capabilities, genetic algorithms can effectively explore complex scheduling solution spaces and perform well in static large-scale JSSP. The maximum completion time obtained is 5%-8% lower than that of the greedy algorithm. For example, in the Job-Shop scheduling problem, the improved genetic algorithm can effectively balance global exploration and local exploitation through adaptive crossover and mutation operators, obtaining better scheduling plans[2]. Reinforcement learning, on the other hand, shows unique advantages in dynamic JSSP scenarios (such as emergency order insertion and equipment failure). Through real-time interaction with the dynamic environment, it can quickly adjust the scheduling strategy, controlling the performance attenuation rate of the scheduling plan within 10%, which is better than genetic algorithms and greedy algorithms.

The solution quality of the greedy algorithm in JSSP is poor. Because it only considers the local optimal processing time, it is easy to cause unbalanced machine load, thereby extending the overall production cycle. However, in scenarios where production tasks are urgent and the requirement for the real-time performance of the scheduling plan is extremely high (such as the rapid response to temporary orders), the greedy algorithm can still be used as an effective means to quickly obtain feasible scheduling plans.

4. Summary of Applicable Scenarios of Algorithms

Based on the comparative analysis of the above two typical optimization problems, combined with the two core dimensions of solution speed and optimal solution quality, the applicable scenarios of reinforcement learning, greedy algorithms, and genetic algorithms can be clarified as follows:

Greedy algorithms are suitable for small-scale optimization problems and scenarios where the requirement for solution speed is extremely high but the requirement for solution quality is low. For example, it can be applied to the rapid path planning for small-scale TSP and the emergency scheduling of temporary production tasks. Its advantages lie in simple implementation and fast solution speed, which can quickly respond to real-time requirements; its disadvantage is poor solution quality, making it difficult to apply to complex large-scale optimization problems.

Genetic algorithms are suitable for static large-scale optimization problems and scenarios where the requirement for solution quality is high but the requirement for solution speed is relatively loose. For example, it is applicable to large-scale logistics distribution path planning and static batch production workshop scheduling. Its advantages lie in strong global optimization capabilities and the ability to obtain high-quality approximate optimal solutions; its disadvantages are slow solution speed, complex parameter tuning, and difficulty in adapting to dynamically uncertain environments.

Reinforcement learning is suitable for dynamically uncertain optimization problems and scenarios where the requirements for solution quality and generalization ability are high and the initial training cost is acceptable. For example, it can be applied to dynamic logistics scheduling (such as real-time traffic congestion adjustment), dynamic workshop scheduling with equipment failures, and the unified solution for multi-scale TSP problem. Its advantages lie in strong adaptive capabilities and good generalization, which can handle optimization problems in dynamic environments; its disadvantages are complex training process, low sample efficiency, and high initial investment cost.

5. Research Shortcomings and Future Prospects

5.1. Research Shortcomings

Existing research on the comparison of the three algorithms still has many shortcomings: first, the comparison scenarios are relatively single, mostly focusing on the static scenarios of TSP and JSSP, and the comparative research on dynamically complex scenarios (such as multi-objective optimization and multi-constraint coupling scenarios) is insufficient; second, the evaluation dimensions are not comprehensive enough. Existing research mostly focuses on solution speed and optimal solution quality, and there are few comparisons on dimensions such as algorithm stability, robustness, and computational resource consumption; third, the sample efficiency problem of reinforcement learning has not been effectively solved, and its competitiveness in small-scale optimization problems is still weaker than that of traditional algorithms[3].

5.2. Future Prospects

Future research can be carried out in the following three directions: first, expand the comparison scenarios, conduct in-depth research on the performance of the three algorithms in complex scenarios such as multi-objective optimization and dynamically uncertain environments, and improve the algorithm comparison system; second, build a multi-dimensional evaluation index system, comprehensively considering factors such as solution speed, solution quality, stability, and robustness, to form a more comprehensive algorithm evaluation framework; third, promote algorithm fusion and innovation, combine the adaptive ability of reinforcement learning with the global optimization ability of genetic algorithms, or use reinforcement learning to optimize the decision-making criteria of greedy algorithms, so as to improve the solution performance of algorithms in complex scenarios[1]. In addition, improving the sample efficiency of reinforcement learning and reducing training costs are also key directions for the wide application of reinforcement learning in optimization problems in the future.

6. Conclusion

Taking TSP and JSSP as research carriers, this paper systematically compares the performance differences between reinforcement learning, greedy algorithms, and genetic algorithms from two core dimensions of solution speed and optimal solution quality. The research shows that the greedy algorithm has the core advantage of fast solution speed and is suitable for small-scale optimization scenarios with high real-time requirements; the genetic algorithm has the core advantage of strong global optimization capability and is suitable for static large-scale optimization scenarios with high solution quality requirements; reinforcement learning has the core advantages of strong adaptive capability and good generalization, and is suitable for dynamically uncertain optimization scenarios with high solution quality requirements.

Different algorithms have their own advantages and limitations. In practical applications, algorithms should be reasonably selected according to factors such as problem scale, dynamic characteristics, and real-time requirements. In the future, through algorithm fusion innovation and evaluation system improvement, it is expected to further improve the solution efficiency and quality of optimization problems, and promote the in-depth application of optimization algorithms in more fields.

References

[1] Graves, Alex, et al. "Hybrid computing using a neural network with dynamic external memory." Nature 538.7626 (2016): 471-476.
[2] Jain, Vinod, and Jay Shankar Prasad. "Solving travelling salesman problem using greedy genetic algorithm GGA." Int. J. Eng. Technol 9.2 (2017): 1148-1154.
[3] Yang, Yunhao, and Andrew Whinston. "A survey on reinforcement learning for combinatorial optimization." 2023 IEEE World Conference on Applied Intelligence and Computing (AIC). IEEE, 2023.
[4] Barrett, Thomas, et al. "Exploratory combinatorial optimization with reinforcement learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 34. No. 04. 2020.