Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 21(1): 1058–1081.
DOI: 10.3934/mbe.2024044
Received: 29 October 2023
Revised: 26 November 2023
Accepted: 26 November 2023
Published: 22 December 2022

Research article

Design of a reinforcement learning-based intelligent car transfer planning system for parking lots

Feng Guo[1], Haiyu Xu[2], Peng Xu[2] and Zhiwei Guo[2,*]

[1] School of Management Science and Engineering, Chongqing Technology and Business University, Chongqing 400067, China

[2] School of Artificial Intelligence, Chongqing Technology and Business University, Chongqing 400067, China

* **Correspondence:** Email: zwguo@ctbu.edu.cn.

Abstract: In this study, a car transfer planning system for parking lots was designed based on reinforcement learning. The car transfer planning system for parking lots is an intelligent parking management system that is designed by using reinforcement learning techniques. The system features autonomous decision-making, intelligent path planning and efficient resource utilization. And the problem is solved by constructing a Markov decision process and using a dynamic planning-based reinforcement learning algorithm. The system has the advantage of looking to the future and using reinforcement learning to maximize its expected returns. And this is in contrast to manual transfer planning which relies on traditional thinking. In the context of this paper on parking lots, the states of the two locations form a finite set. The system ultimately seeks to find a strategy that is beneficial to the long-term development of the operation. It aims to prioritize strategies that have positive impacts in the future, rather than those that are focused solely on short-term benefits. To evaluate strategies, as its basis the system relies on the expected return of a state from now to the future. This approach allows for a more comprehensive assessment of the potential outcomes and ensures the selection of strategies that align with long-term goals. Experimental results show that the system has high performance and robustness in the area of car transfer planning for parking lots. By using reinforcement learning techniques, parking lot management systems can make autonomous decisions and plan optimal paths to achieve efficient resource utilization and reduce parking time.

Keywords: Reinforcement learning; automated planning system; Markov decision process; dynamic programming

1. Introduction

As China's standard of living rises, the proportion of people with a driving license continues to increase year by year [1]. At the same time, parking difficulties have become a common problem plaguing the public [2]. In order to solve this problem, the rational planning and management of car parks is particularly important [3]. In practical parking planning, traffic flow, vehicle types, surrounding road conditions, parking demand and how to maximize efficiency should be taken into account [4]. Comparatively speaking, traditional manual scheduling methods are deficient in their analysis of the problem and ability to yield decisions [5]. They cannot satisfy the need for fast and efficient computation and are somewhat blind. In terms of transferring cars, it is a different and more problematic scenario than the usual one. This is because it has an infinite state space. While using reinforcement learning the car park is viewed as an intelligent system, which learns by trial and error and guides its behavior through the rewards obtained from interacting with the environment. The car planning system is a multi-objective planning problem with infinite state space. Reinforcement learning can be used to optimize the performance of the system by calculating the optimal combination of strategies through its powerful search strategy [6].

Therefore, designing a reinforcement learning based car transfer planning system for parking lots can effectively solve this problem. Reinforcement learning is a type of machine learning, and it is widely used in the field of artificial intelligence [7]. It allows an intelligent body (agent) to learn by trial and error in an environment to obtain a strategy that maximizes the desired reward. And it is suitable for solving for the sequential decision-making class of problems [8]. The use of this system can enhance the efficiency and accuracy of managers' decision-making process. As a result, it reduces the incidence of risk and error in decision-making. Additionally, utilizing this system can save time and cost in practical applications [9]. Considering the non-negligible environmental factors, it becomes necessary for intelligent bodies to continuously adjust their behaviors and strategies [10]. This adjustment is essential for the requirement of continuous optimization and improvement in performance and effectiveness [11]. Only in this way can the requirement to continuously optimize and improve one's performance and effectiveness be met. Thus, the goal of maximizing the subsequent desired return is achieved. In addition, the development of machine learning provides solutions for optimal control of life's work. In the literature [12], scholars have introduced a variational data assimilation model as a way to deal with sparse, unstructured and time-varying sensor data. In another study [13], a new algorithmic model was built by combining data assimilation with a machine learning model. The authors used this model to implement real-time fire prediction. In a study by Zhong et al. [14], a digital twin fire model was developed for an interactive fire and emissions algorithm for natural environments. Reduced-order modeling and deep learning predictive models are utilized to enhance accuracy and effectiveness in simulations of fire behavior and emissions.

Among the many problems related to car transfer planning, one of the goals pursued is maximization of the expected return. This is achieved by calculating the optimal transfer scheme in scenarios in which vehicles are moved from one parking space to another. The challenge lies in determining the most efficient way to allocate and relocate vehicles to maximize the overall expected return. In parking lots, transfer planning can improve the efficiency of vehicle scheduling and management and increase economic efficiency. It is therefore a research topic of interest in both academia and industry. Wang et al. stated that vehicle rental yard systems are most widely used due to their flexibility, but they may

encounter imbalance, particularly in the forms of saturation and exhaustion, and this imbalance may lead to loss of revenue [15]. Huang et al. considered the vehicle management problem with uncertain demand and proposed a solution for vehicle allocation. In their paper, they stated that the number of vehicles an operator needs to move in or out for each site is related to the vehicle inventory. And the inventory depends on the current inventory, vehicle pickup demand and vehicle return demand [16]. Oliveira et al. suggested that how to divide the existing fleet between parking lots is a key aspect. The number of vehicles at each site is constantly changing due to rentals and returns. So the cost of moving vehicles can be reduced by employing appropriate planning methods [17]. Similarly, for the vehicle allocation scheduling problem, Wang et al. analyzed the vehicle scheduling management problem for buses. They also proposed to establish a theoretical model for automated calculation of the optimal driver and bus scheduling scheme, as well as the use of a dynamic programming algorithm for its storage [18]. Wang et al. proposed that the demand for electric car sharing can be predicted by a hidden Markov model with the goal of maximizing corporate profits. Finally, a regional level electric car sharing optimization relocation model was developed [19]. Hao et al. used a novel distributionally robust optimization method. The method can use covariate information and demand moment information to construct scenario-dependent fuzzy sets to solve the problem of pre-allocation of idle taxi vehicles [20].

Regarding research in reinforcement learning, in work by Wang et al. [21], the tracking of an unknown unmanned surface vehicle in a complex system is optimized by using a reinforcement learning control algorithm. This algorithm is applied to enhance the performance and accuracy of tracking for an unmanned surface vehicle system. Alternatively [22], a self-learning model-free solution was designed to optimally control an unmanned surface vehicle. Wang et al. [23]developed a model that combines behaviorally critical augmented learning mechanisms with finite time control techniques. In this way, the tracking of an unmanned surface vehicle is optimized. Liu et al. proposed a method for achieving human-level control by using deep reinforcement learning. Their work on achieving best-in-class performance on multiple Atari games by using directly trained SNNs [24]. Peng et al. proposed an imitation learning system based on reinforcement learning that enables legged robots to learn agility-type motor skills by imitating real-world animals [25]. Zhang et al. proposed a deep reinforcement learning-based approach to allow unmanned aerial vehicles to perform navigation tasks in multi-obstacle environments with randomness and dynamics [26]. Oh et al. proposed a novel experience replay method. It employs new component-driven learnable features in model-based reinforcement learning to compute the experience scores [27].Li et al. proposed a novel advanced autonomous driving integration method based on end-to-end multi-agent deep reinforcement learning. It is capable of autonomously learning complex and realistic traffic dynamics [28]. In summary, reinforcement learning is widely used in several fields. At the same time, researchers have continued to push the development of reinforcement learning algorithms by proposing new methods and techniques, such as deep reinforcement learning, prioritized experience playback and multi-intelligent body reinforcement learning. These studies provide new perspectives and approaches to the application and theory of reinforcement learning and promote its further application and development in practice.

2. Theoretical methods

In this work, reinforcement learning was applied to a car transfer planning system for parking lots to find out the state, action, policy and reward of the problem based on the established Markov decision process model. In context, this sequential decision-making problem is suitable for the model-based dynamic planning approach. Thus, this problem has been divided into several subproblems and solved by using either strategy iteration or value iteration. The problem of how to choose a suitable iterative method is also addressed in this paper. A value matrix and an action matrix have been created. And an iterative approach is used to keep the two matrices updated until the values in the matrices reach convergence. For the Poisson distribution probability problem involved in this system, a module has been designed to find the parameters by inputting the parking lot and return data from two parking lots at a certain time period. And the optimal Poisson distribution parameters are obtained via a computational solution for the calculation of the state transfer probability of the Markov decision process.

2.1. Overview of technology pathways

In order to solve the problem of maximizing the expected return based on transfer planning, a car transfer planning system has been designed based on reinforcement learning. And, the problem is solved by constructing a Markov decision process and using a dynamic planning-based reinforcement learning algorithm. The system consists of two modules, i.e., a Poisson distribution parameterization module and dynamic programming solution module. The former calculates the imported data and finds a suitable parameter as a basis for the latter's solution. In the context of this paper on parking lots, the states of the two locations form a finite set. And each state corresponds to an action and a value, respectively. The dynamic programming solution module is based on the Markov decision process and dynamic programming ideas. And Poisson distribution is used as the basis of state transfer probability. The dynamic planning solution module is based on the Markov decision process and dynamic planning idea, and it adopts the Poisson distribution as the basis of state transfer probability. After that, the optimal strategy and optimal state value of each state are obtained via the dynamic planning iterative strategy. Finally, the computed data are stored in the database, so as to realize the fast query of the corresponding strategy and value of each state.

The specific business scenario diagram is shown in Figure 1.

2.2. Markov decision process

2.2.1. Markov decision process principle

The Markov decision process is a commonly used mathematical model in the field of artificial intelligence. It can be used to solve sequential decision-making problems, and ultimately the optimal strategy can be obtained via algorithms such as strategy iteration and value iteration [29]. At this stage it already has a wide range of applications in artificial intelligence, operations research, cybernetics, economics and other fields [30].

The first one introduced is the Markov process. It is applied to a temporal process in which the state at moment $t + 1$ depends only on the state S_t at moment t, independent of any previous state [31]. And the sequence of states is obtained via sampling operations through the use of the state transfer probability matrix given by the Markov process. The Markov reward process was introduced on the

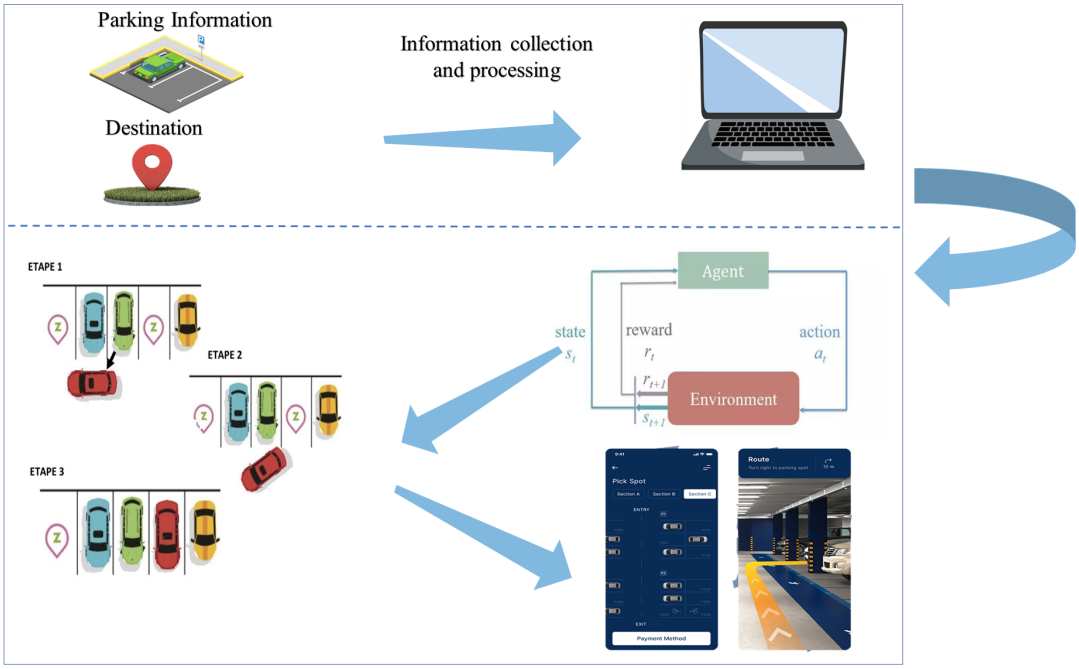


Figure 1. Business scenario diagram.

basis of the Markov process, the components of which are represented by tuples as $< S, P, R, \gamma >$. S is a finite set of states; P is the state transfer probability matrix; R is the reward function; and γ is the decay factor with a range interval of (0, 1] [32]. The cumulative reward for completing a series of states is expressed in terms of a return. The mathematical expression is

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

In order to describe the importance of the current state, the Markov reward process introduces value, which represents the expected reward. The mathematical expression is

$$V_s = E[G_t \mid S_t = s] \tag{2.2}$$

The value V_t of a state S_t is represented by the harvest expectation of that state. The state is sampled through the use of a Markov probability transfer matrix. Thus, a collection of corresponding state sequences is generated. The harvest of each state sequence in the set of state sequences is calculated by means of a discount function. The average harvest of the state is then obtained by performing a weighted summation of all harvests.

The mapping between values and states can be established by using the value function. Since it is unrealistic to calculate the harvest of all state sequences of a state in a practical situation, is modified to obtain the following formula:

$$\begin{aligned} V_s &= E[G_t \mid S_t = s] \\ &= E\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\ &= E\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \end{aligned} \tag{2.3}$$

According to the above equation, the modification can be continued to obtain the Bellman equation for Markov processes. Its mathematical expression is

$$V_s = R_s + \gamma \sum P_{ss'} \cdot V_{s'} \tag{2.4}$$

$S^{'}$denotes any state at the next moment of state s. In this case, the value V_s is determined by R_s of the current state, the state transfer probability of the current state, $V(S_t + 1)$ at the moment $t + 1$ and γ.

The problem of this paper involves the behavior (action) of the intelligent systems themselves, so a Markov decision process is introduced here. This is a mathematical model applied to provide a description of stochastic decision-making processes, and it is widely used to find the optimal strategy [33]. Its composition can be represented by the tuple $< S, A, P, R, \gamma >$. A is the set of actions of the intelligent body and the set is finite. P is the state transfer probability. The Markov decision process introduces the notion of policy and denotes by π the law of probability distribution of an action performed by an intelligent body in a certain state. It can be expressed as

$$\pi(a \mid s) = P[A_t = a \mid S_t = s] \tag{2.5}$$

The equation describes the probability of performing action a in state s. A is the set of actions. When an intelligent body introduces an action, the value function will be different from that of the Markov reward process, and the selection of the action will change the current environmental state. And the intelligent bodies in different states will produce different actions in response to them; also, the actions occur according to the probability distribution law π. In this regard, the Markov decision-making process introduces the value function $Q_\pi(s, a)$ of the action based on the policy π. It considers the action factor on the basis of the original. At this point the state value function $V_\pi(s)$ and the action value function $Q_\pi(s, a)$ are expressed in terms of the Bellman equation as follows:

$$\begin{aligned} V_\pi(s) &= E[G_t \mid S_t = s] \\ &= E[R_{t+1} + \gamma V_\pi(s') \mid S_t = s, A_t = a] \end{aligned} \tag{2.6}$$

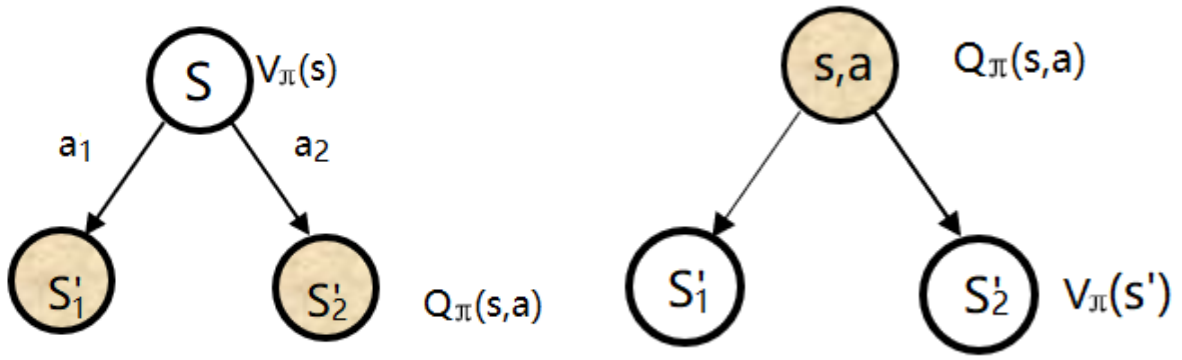
$$\begin{aligned} Q_\pi(s, a) &= E[G_t \mid S_t = s, A_t = a] \\ &= E[R_{t+1} + Q_\pi(s', a') \mid S_t = s, A_t = a] \end{aligned} \tag{2.7}$$

The state value function and the action value function are interrelated and can be expressed in terms of each other. The value of a state can be expressed by using the value of all actions in that state multiplied by the corresponding probability distribution. Similarly, the value of an action can be expressed by multiplying the values of the successor states to that state by the corresponding probability distribution. Their respective relationships are shown in Figure 2(a) and 2(b).

The mathematical equation for the mutual representation between the state value function and the action value function is as follows:

$$V_\pi(s) = \sum_{a \in A} \pi(a \mid s) Q_\pi(s, a) \tag{2.8}$$

$$Q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss}^a \cdot V_\pi(s') \tag{2.9}$$



(a) Plot of state value function to action value function (b) Plot of action value function to state value function

Figure 2. Plots of state value function as related to the action value function.

Combining the above mathematical equations with each other gives the following mathematical equation:

$$V_\pi(s) = \sum_{a \in A} \pi(a \mid s)\left(R_s^a + \gamma \sum_{s' \in S} P_{ss}^a{}' V_\pi(s')\right) \tag{2.10}$$

$$Q_\pi(s, a) = R_{ss}^a + \gamma \sum_{s' \in S} P_s^a \sum_{a \in A} \pi(a' \mid s') Q_\pi(s', a') \tag{2.11}$$

2.2.2. Markov decision process construction process

According to the Markov decision process element composition tuple $< S, A, P, R, \gamma >$, the process is constructed as follows.

First, a finite set of states (S) is determined: it contains all of the possible states of the system. In the problem of this paper, the set of states is the current number of all possible vehicles in the two parking lots.

Determine a finite set of intelligent body actions (A): the elements of this set are the actions that the intelligent body can perform in each possible state. In the problem of this paper, the set of actions is the number of cars that can be moved between two parking lots.

Determine the state transfer probability (P): based on a certain law, determine the probability that, in each possible state, the intelligent body will cause the system to transition to the next state after performing a certain action. The law can be obtained statistically from several experiments or as based on theoretical analysis. In the problem of this paper, the Poisson distribution is used to determine the probability of taking each action in each state.

Determine the state-based and action-based reward function (R): establish a function such that the value of the reward obtained by the intelligent body after performing an action in each state is mapped to the action and state. In the problem of this paper, the correct relationship is established by relating the realities of the problem, such as the cost required to move a car.

Determine the appropriate attenuation factor (γ): the role of the attenuation factor γ in the reward function is to measure the importance of the future reward. The more γ tends to 1, where the higher the

value, the greater the importance of the future reward. For example, in the game of Go, the ultimate goal is to win, not to train the intelligent body to keep capture the opponent's pieces. At this time, the importance of future rewards is high, and γ will be set more inclined to 1. In the problem of this paper, the benefits of parking lots should be considered in the long run, so γ should be set as relatively high.

Determine the optimal policy: an optimal policy π is finally obtained by iterating until the value converges; it is the maximum desired reward value that can be obtained by taking $\pi(s)$ actions in any state. The optimal strategy can be solved by performing either value iteration or strategy iteration [34].

The π value of the optimal policy at this point is not the π value of the Markov decision process, and the optimal policy π can be called π^*, $\pi^* \geq any\pi$. At this point it is not a probability distribution law, but a definite array of 0s and 1s. It can be expressed as follows:

$$\pi^*(\mathrm{a} \mid \mathrm{s}) = \begin{cases} 1 & \text{if a} = \underset{\mathrm{a=A}}{\operatorname{argmax}} Q^*(\mathrm{s}, \mathrm{a}) \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

The V^* parameter denotes an optimal state value function corresponding to π^* , Q^* denotes the optimal action value function, and the mathematical expression is as follows:

$$\mathrm{V}^*(\mathrm{s}) = \max_{\mathrm{a}} \left[\mathrm{R}_{\mathrm{s}}^{\mathrm{a}} + \gamma \sum_{\mathrm{s}' \in \mathrm{s}} \mathrm{P}_{\mathrm{ss}'}^{\mathrm{a}} \cdot \mathrm{V}^*(\mathrm{s}') \right] \tag{2.13}$$

$$\mathrm{Q}^*(\mathrm{s}, \mathrm{a}) = \mathrm{R}_{\mathrm{s}}^{\mathrm{a}} + \gamma \sum_{\mathrm{s}' \in \mathrm{s}} \mathrm{P}_{\mathrm{ss}}^{\mathrm{a}} \max_{\mathrm{a}'} \mathrm{Q}^*(\mathrm{s}', \mathrm{a}') \tag{2.14}$$

The construction process of the Markov decision process model includes determining the basic elements such as the state set, action set, state transfer probability, reward function and decay factor. Then the commonly used strategy iteration or value iteration method is adopted to find the optimal strategy. When building the Markov decision process model, attention needs to be paid to the actual situation of the problem to ensure the reliability and validity of the model.

2.3. Dynamic programming

2.3.1. Overview of dynamic programming

Dynamic programming is often used to obtain the optimization results, and its main idea is to divide a large problem into small problems to obtain the solution and reuse the obtained results in the small problems [35]. Dynamic programming has a very wide range of applications, covering fields such as computer science, operations research, economics and biology. The idea can be described as decomposing the problem, defining the state and solving the problem [36]. These processes will be described below:

Decomposition of the problem: The original problem is split into subproblems and the links between them are defined. Often, the subproblems will have some remaining features that are consistent with the original problem. For example, in a path planning problem, given a start point and an end point, it is necessary to find the shortest path that connects them. This problem can be decomposed into several subproblems. Each subproblem is the shortest path from the starting point to a node on the current path. And this node can be the end point or an intermediate node. Each sub-problem needs to take into account the information that has been previously obtained. That is, the shortest path is known.

Defining states: For each subproblem, a corresponding state needs to be defined to reflect the characteristics and known information of the subproblem. In the path planning problem described above, it is possible to define each subproblem as the shortest path from the starting point to the current node. The connection between states can be described by using some transfer equations.

2.3.2. Dynamic programming strategies for strategy iteration

Strategy iteration is a dynamic programming strategy that belongs to the class of iterative algorithms. It optimizes the strategy by performing two steps over and over again, i.e., strategy evaluation and strategy improvement [37]. In the strategy evaluation phase, the performance of the currently used strategy is evaluated by calculating the value function of the strategy. The value function represents the expected value of the long-term return that can be obtained from the beginning to the end of the computation for each state while in the current strategy. This process can be done by taking the solution of the Bellman equation. The Bellman equation is a recursive equation. It expresses the value function of a state in terms of the weighted average of the value functions of the states adjacent to that state.

In the strategy optimization phase, strategies are optimized and improved based on the value function of the currently selected strategy. Specifically, an optimal action is chosen for each state. This action maximizes the value function of that state. This process can be achieved by implementing a greedy algorithm. A greedy algorithm is one that selects the action that maximizes the value function in each state [38]. The strategy iteration algorithm implements strategy evaluation and strategy improvement over and over again. It ensures that the strategy does not change anymore. When the strategies converge, the optimal strategy with the optimal value function is obtained. The advantage of the policy iteration algorithm is that it ensures convergence to the optimal policy. However, it is computationally intensive to perform policy evaluation and policy improvement for each iteration.

2.3.3. Dynamic programming strategies for value iteration

Value iteration is a dynamic planning strategy. It belongs to a class of iterative algorithms. Unlike strategy iteration, it optimizes the strategy by performing one step over and over again: iterative value update [39]. In the value iteration update phase, the value function of the current state is updated based on the value function of that state. Specifically, for each state an action is chosen that maximizes the value function of that state. This action is still obtained by solving the Bellman optimality equation. The value iteration algorithm performs the value iteration update step repeatedly and stops iterating when the value function converges. At this point the optimal value function is obtained. The optimal policy can be obtained by choosing the action that maximizes the value function in each state.

The advantage of the value iteration algorithm is that it is less computationally expensive. This is because only one value iteration update is required per iteration [40]. However, it is not guaranteed to converge to the optimal policy because in some cases, the optimal policy may not be a greedy policy. In addition, the computational cost of the value iteration algorithm becomes high when the state space is large.

3. Design of car transfer planning system

3.1. Overview of the car transfer planning system

This car transfer planning system contains two modules: They are Poisson distribution parameter calculation and dynamic planning solution module. The Poisson distribution parameter calculation module is mainly purposed to classify and count the data of the uploaded files and find the best-fitting Poisson distribution parameters as the basis for the subsequent probability calculation. The uploaded data should be the number of car transfers and car returns in each of the two places in a longer period of time. The dynamic programming solver module focuses on finding the optimal strategy by passing the set data parameters into the model as states, actions, rewards, penalties, etc. of the system and training it until convergence. And the results of training are presented in the form of a 3D scatter plot and heat map.

The specific technical methodology process is as follows:

Algorithm 1 Reinforcement learning-based car transferring planning method

1: Initialize parking lot environment:
    Define state space S
    Define available action space A
    Initialize Q-value function $Q(S, A)$ as 0
2: Define car relocation reward function $R(S', A, S)$:
    Define the reward function based on the specific problem
3: Define reinforcement learning algorithm parameters:
    Learning rate α
    Discount factor γ
    Exploration and exploitation trade-off parameter ϵ
4: Training iteration:
    For each episode:
        Initialize starting state s
        Repeat for each step in the episode:
            Use epsilon-greedy policy to choose action a
            Execute action a, observe new state s' and reward r
            Update Q-value function:
                $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
5: Model evaluation:
    Test and evaluate the trained Q-value function
    Different evaluation metrics can be used
6: Model improvement:
    Based on evaluation results, improve and optimize the model
7: Output optimal car relocation policy:
    Based on the trained Q-value function, obtain the optimal car relocation policy

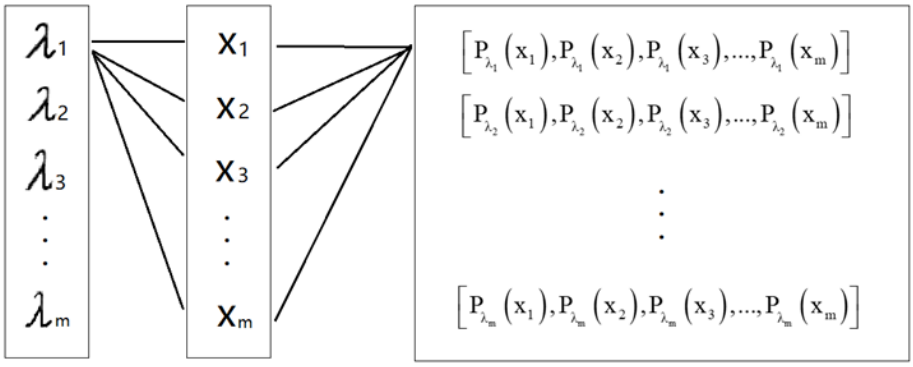


Figure 3. Each λ performs a probabilistic operation on all data points in this column.

3.2. Design of poisson distribution parameter calculation module

In the system designed in this paper, the Poisson distribution probability plays a crucial role in the global picture as the basis of the state transfer probability of the Markov decision process. Therefore, it is necessary to find a Poisson distribution parameter that conforms to the data law. In this work, the Poisson distribution parameter calculation module has been designed for the implementation of this function. When setting the Poisson distribution parameters on the interactive page, one can upload an Excel table by uploading a file; one can then take out the data saved in the table and put it into an array. This array can be used as a parameter for classification statistics. As shown in Table 1, the Excel sheet was designed with four columns, in order of the number of cars requested to be transferred and the number of cars requested to be returned for site 1, and the number of cars requested to be transferred and the number of cars requested to be returned for site 2. And the data for the same day is shown in one row.

The calculation of Poisson distribution is shown in Eq (3.1). Let the number of data points for a certain number x in a certain column obtained after the counting operation be n and the total number of days be N. Then the probability of x under this request at this site can be expressed as $P(x) = \frac{n}{N}$. The parameter λ is obtained by computing x as k and $P(x)$ as $P(X = k)$ of the Poisson distribution.

$$\mathrm{P(X = k)} = \frac{\lambda^{\mathrm{k}}}{\mathrm{k!}}\mathrm{e}^{-\lambda} \tag{3.1}$$

Table 1. Formatting of forms for upload to the system.

Days	Transferred cars (vehicles) I	Returned cars (vehicles) I	Transferred cars (vehicles) II	Returned cars (vehicles)II
1	8	10	7	3
2	8	9	11	0
3	3	4	0	11
...	...	...	...	...

Combining the actual situation and the formula for Poisson distribution, the final output of the

model should be an integer corresponding to each column in the table. Since each data point in each column will generate a Poisson distribution parameter, the data point in each column will yield a list consisting of the Poisson distribution parameter λ after calculation. In order to find a unique result for each column, the idea of a greedy algorithm is used in this work to achieve optimization. That is, for each column, the probability of every λ is calculated for all the data within that column. Subsequently, N probability lists corresponding to different λ value are generated for each column, as shown in Figure 3.The difference between the obtained probability and the true probability is calculated as its absolute value. In the list of differences obtained for each λ, the largest absolute value is taken. After that, find the minimum value among all the maximum absolute values. The λ corresponding to the absolute value of the obtained difference is the optimal result.

3.2.1. Design of Dynamic Programming Solution Module

In the system detailed in this paper, the Poisson distribution parameters as well as various conditional data will be passed as parameters to the dynamic programming solution module. The required parameters are the maximum vehicle capacities of site 1 and site 2, respectively, and the Poisson distribution parameters for site 1 and site 2 for transferring and returning a vehicle. Additionally, the cost of moving a vehicle (cost), the maximum number of request ceilings and the maximum number of vehicles to be moved are also crucial parameters to be considered. The maximum number of vehicles to be moved was designed as the upper limit of the absolute value of the elements in a set of action integers, naming the set A. A positive number is used in set A to represent the number of vehicles moving from site 1 to site 2. Negative numbers represent the number of vehicles moving from site 2 to site 1. In this module, it is necessary to define a value matrix, $V(s)$ for storing the values that have been iterated to convergence, and of size [(maximum number of vehicles at site 1 + 1) (maximum number of vehicles at site 2 + 1)]. It is also necessary to define an action matrix π^* for storing the optimal policy; it is of size [(maximum number of vehicles at site 1 + 1) (maximum number of vehicles at site 2 + 1)]. Among them, the action is the number of vehicles that the agent chooses to move to meet the requirement of not exceeding the number of vehicles held at the site. The reward is calculated as the difference between the earning from renting out the vehicle and the cost of moving the vehicle after executing this action. The environment is the customer group and other factors, such as the request to rent and return a vehicle.

Based on the importance of future rewards, it is considered that the benefits should be considered in the long run. Therefore the decay factor γ is defined as 0.9. This is used to ensure that the final convergence and V_π are unique. In this problem, actions and states should be considered with constraints. That is, the number of vehicles moved must not exceed the number of vehicles held at the site. And the sum of the number of vehicles moved to the target site must not exceed the maximum capacity value. And, the total number of vehicles at both sites before and after the move is kept constant. If the number of transferred requests is greater than the number of vehicles held, all vehicles at the location are transferred at most. If the sum of the number of change requests and the number of vehicles held is greater than the upper capacity limit, the vehicle is moved to another location. Based on such constraints, a summation function is designed for the purpose of finding the expected value of $V(s)$ obtained by using a certain action in a certain state. The flow is shown in Figure 4.

The core strategy evaluation formula for this value function is as follows:

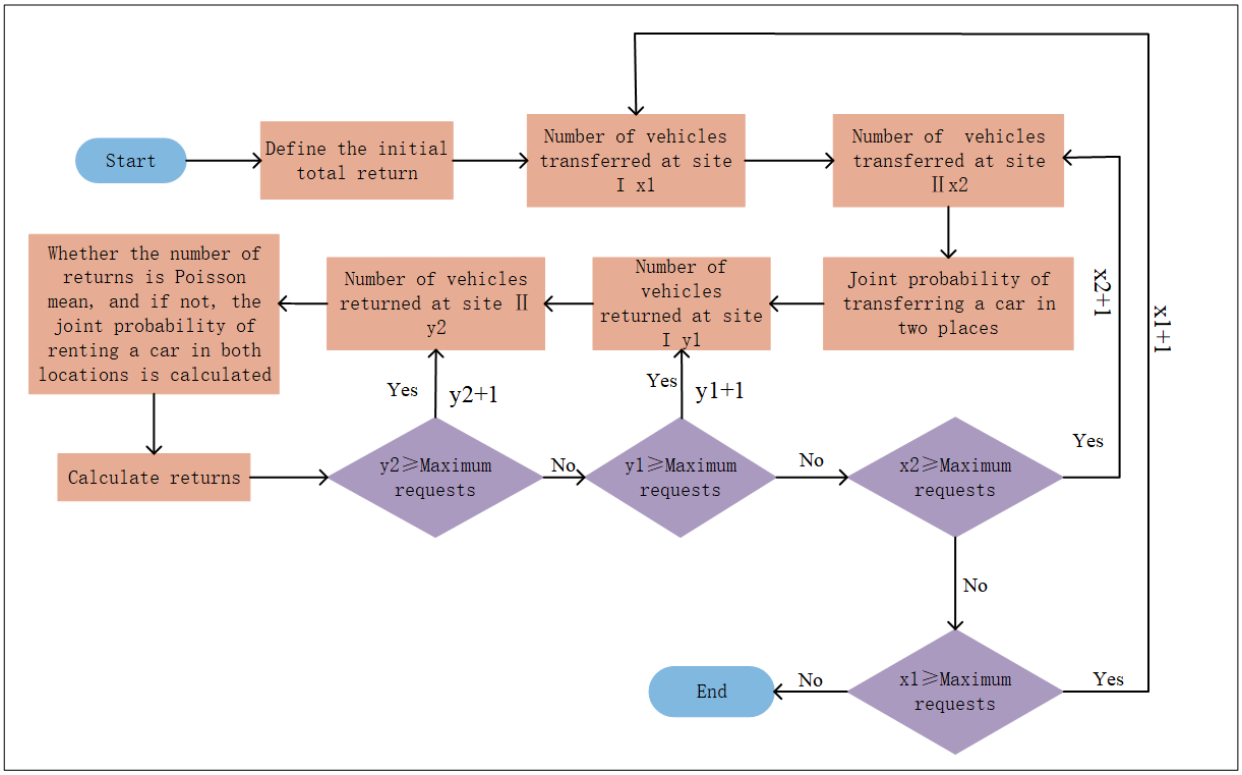


Figure 4. Flowchart for the value function.

$$V(s) = p(s', r \mid s, \pi(s))[r + \gamma V(s')] \tag{3.2}$$

Based on the above value function, the module provides two methods i.e., strategy iteration and value iteration. The first step is to randomly assign the initial value, named policy 0. In this work, the initial action matrix is set as the policy, and the initial value matrix is set as the state value. The second step is to call the value function to calculate the state value according to the current policy. The third step is to compare the size of the new state value with the old state value according to the principle of expectation maximization, take the larger value and update the corresponding policy. The second and third steps are repeated until the state values and strategies converge to the optimal state values and optimal strategies.

The value iteration is one step less than the strategy iteration. In this paper, we will find the optimal state value first, as well as determine the optimal strategy based on the state value. Therefore, in the first step, we set the initial action matrix as the strategy and the initial value matrix as the state value. In the second step, the value function is invoked sequentially on the actions to obtain the corresponding state values, and the maximum value of the state values obtained from all actions is taken. This value is replaced with the number in the corresponding position of the initial value matrix. Repeat the second step until the state values converge to the optimal state values.

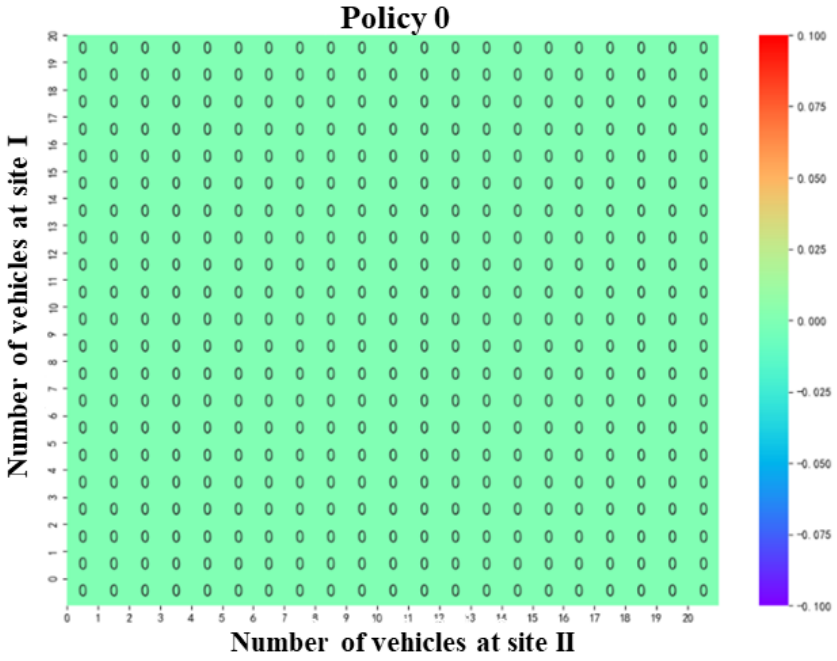


Figure 5. Heat map of the initial strategy (without any improvements).

4. Results and analysis

The background of this experiment was set as follows: the maximum number of vehicles in two places is 20, the cost of moving a car is 2 yuan and the revenue from transferring a car is 10 yuan. The maximum number of requests for car transfer and return is 11, and the maximum number of cars that can be moved is 5. The car transfer and car return Poisson parameters for the first site are 3, the second site's transfer Poisson parameters are 4 and the car return Poisson parameters are 2. And the number of cars returned to the two places obey the Poisson mean.

4.1. Strategy iteration experimental results

As shown in Figure 5, policy 0 represents the initial policy (without any improvement). At this time, all strategies are 0, that is, no movement is made. Positive numbers represent the number of vehicles moving from site 1 to site 2, and negative numbers represent the number of vehicles moving from site 2 to site 1.

As shown in the heat map of Figure 6, policy 1 represents the results after the first strategy improvement. From the figure, it can be seen that, when the number of vehicles held at site one is greater than 7, the policy is optimized to move five vehicles to site 2. And, as the number of vehicles held at site 2 increases, the percentage of moving five vehicles decreases. The policy 1, in general, greatly favors moving vehicles from site 1 to site 2.

As shown in Figure 7, policy 2 represents the results after the second strategy improvement. Compared to polict 1, the percentage of site 1 moving 5 vehicles to site 2 decreases significantly. And when the number of vehicles held by the second site is greater than 7, it tends to move vehicles from the

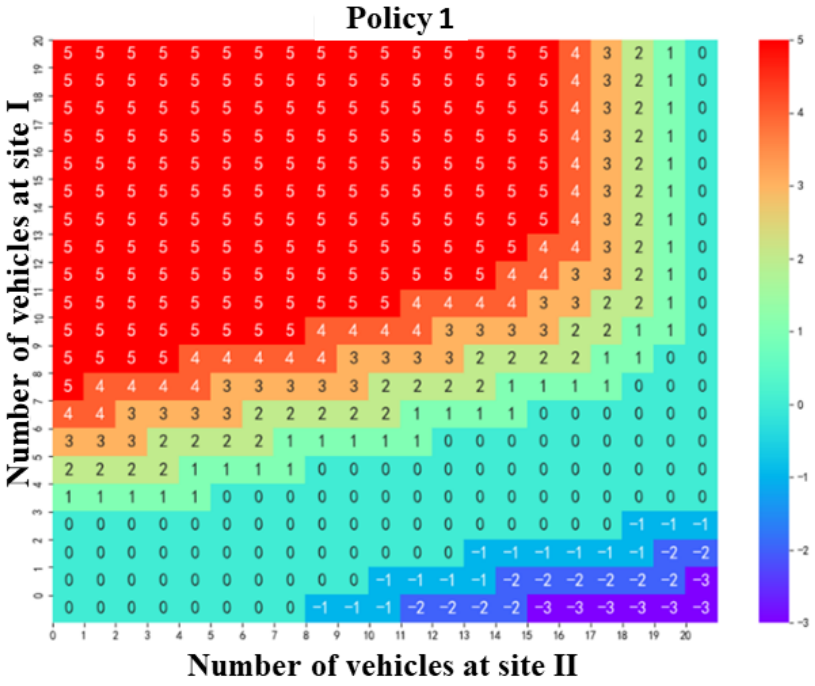


Figure 6. Heat map after the first strategy improvement.

second site to the first site.

As shown in Figure 8, policy 3 represents the results of the third strategy improvement. Compared to policy 2, the percentage of moving vehicles has increased at both sites and the strategies are converging.

As shown in Figure 9, policy 4 represents the results after the fourth policy improvement. It has now converged to the optimal policy.

The state values of the strategy iterations are shown in the heat map of Figure 10.

As shown in the 3D scatter plot of Figure 11, the scatter of the number of vehicles held in the two locations versus the optimal state value forms an approximately curved shape. As the number of vehicles held in the two locations increases, the expected return is higher.

4.2. Value iteration experimental results

As shown in Figure 12 , policy 0 represents the initial policy (without any improvement) and policy 1 demonstrates that the optimal policy has been found.

As shown in the 3D scatter plot of Figure 13, this result is consistent with the strategy iteration method.

4.3. Comparison of the results of the two iterations

Both policy iteration and value iteration can converge to the optimal policy and optimal value state. The running times of the strategy iteration and value iteration are shown in Table 2 respectively. The strategy iteration and value iteration algorithms go through 10 strategy optimizations respectively. From Table 2, it can be seen that the running time gradually converges to a stable value when going

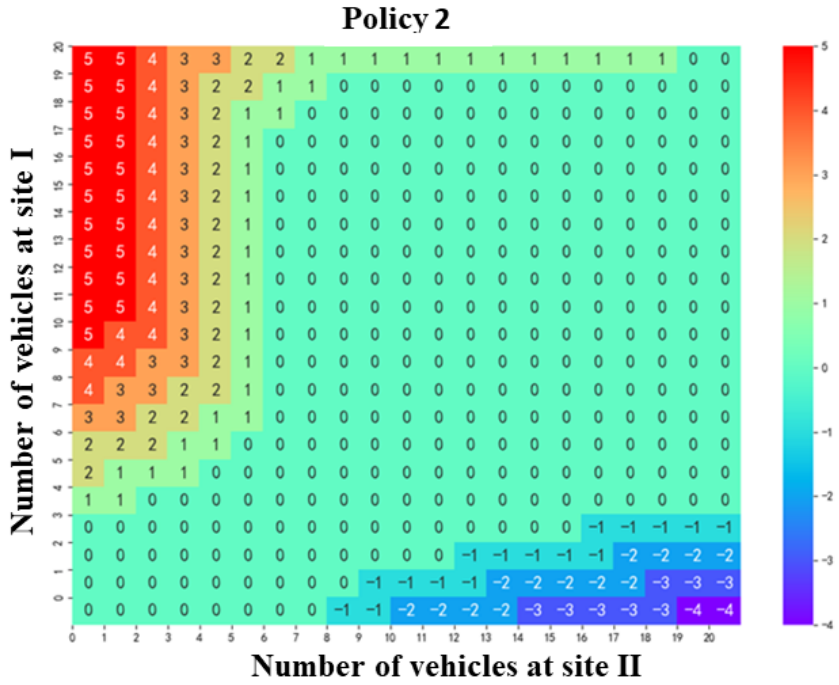


Figure 7. Heat map after second strategy improvement.

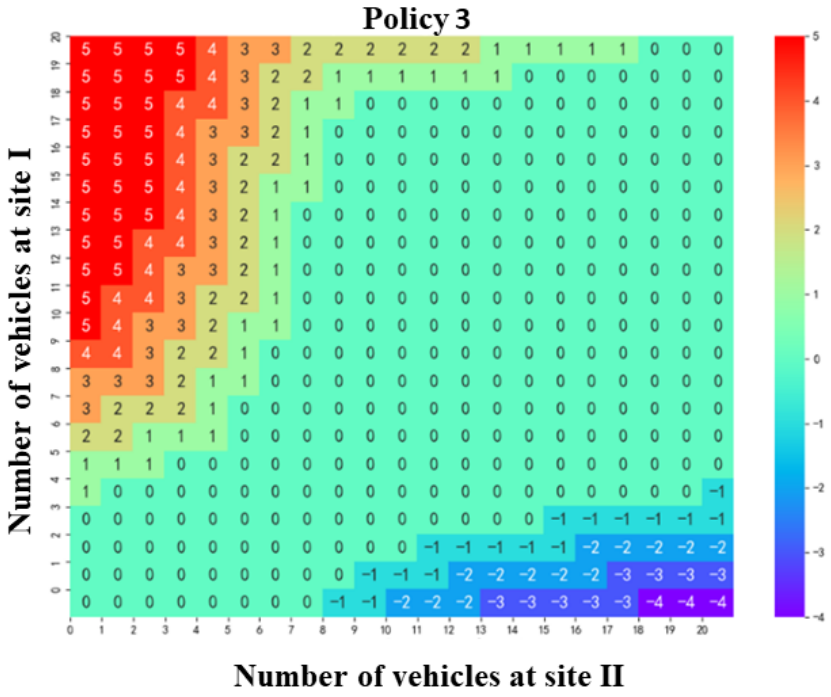


Figure 8. Heat map after third strategy improvement.

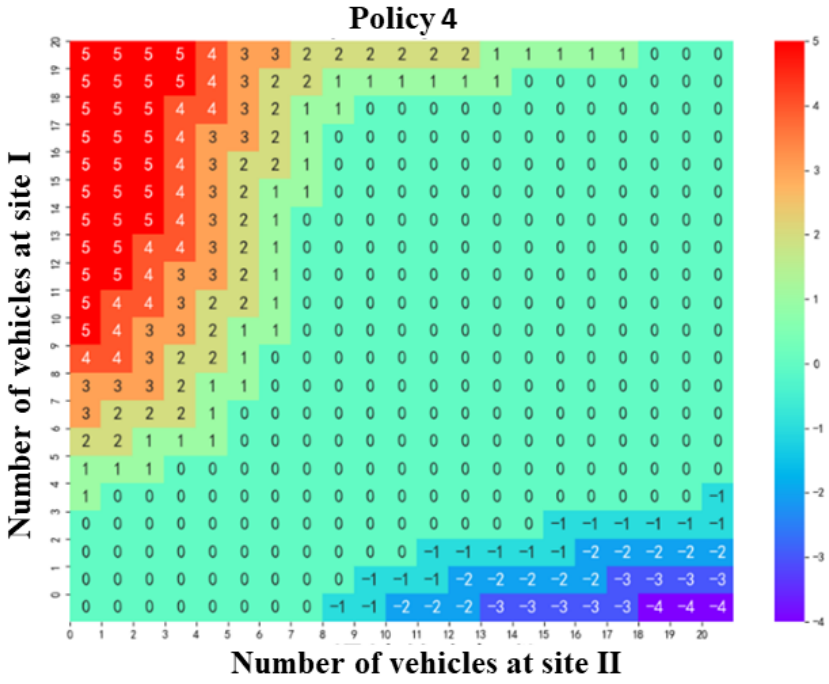


Figure 9. Heat map after fourth strategy improvement.

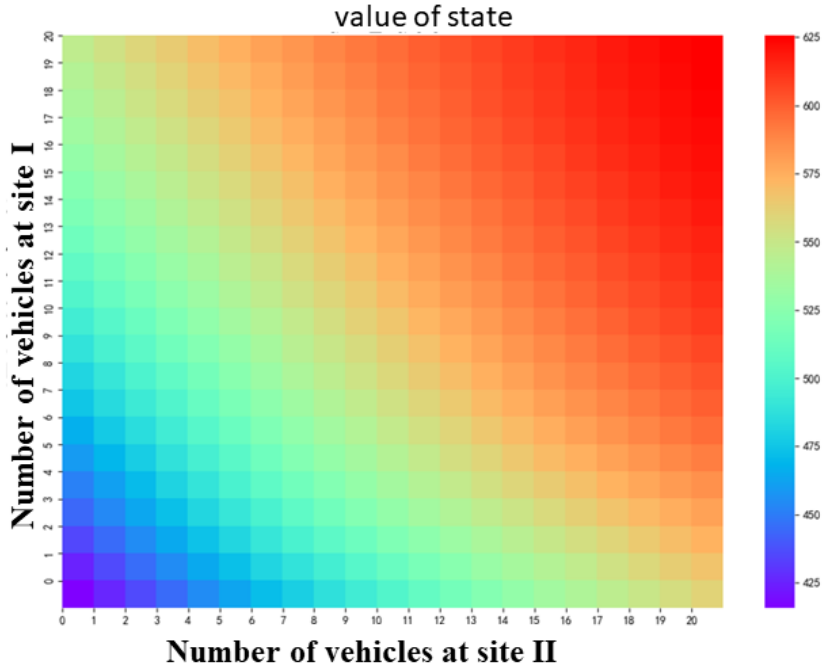


Figure 10. State Value Heat Map for Strategy Iteration.

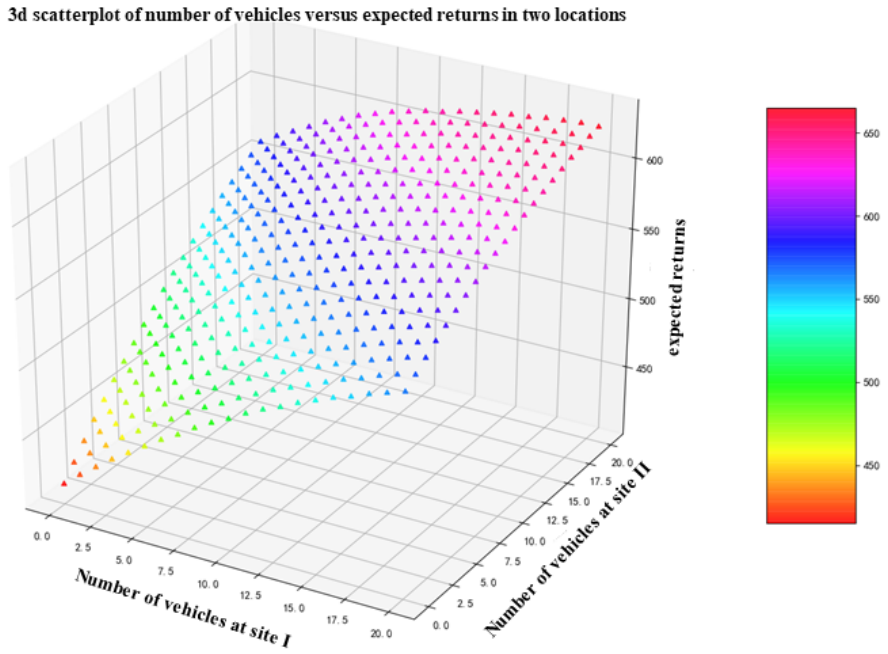


Figure 11. 3D scatterplot for strategy iteration.

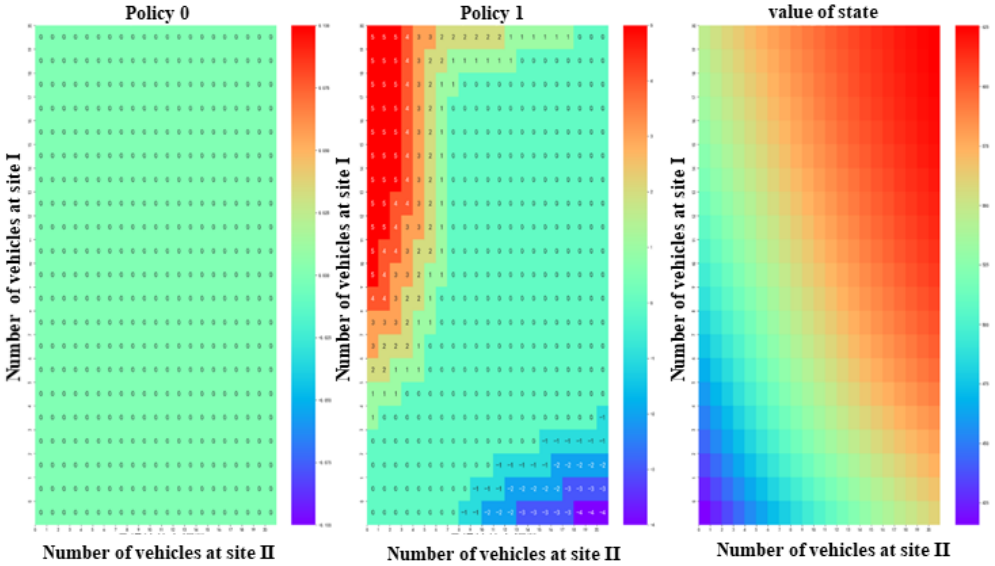


Figure 12. Value iteration strategy Heat maps.

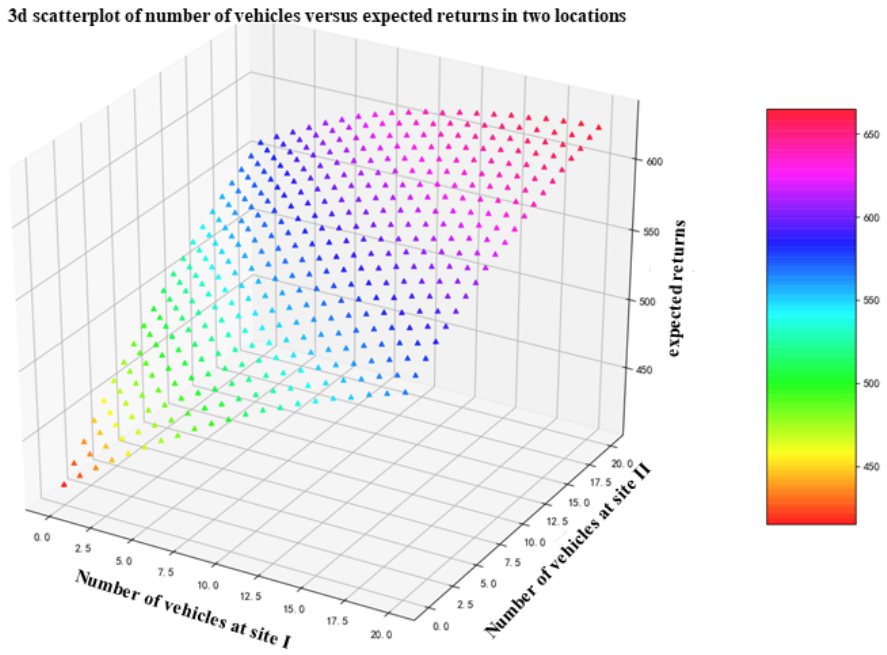


Figure 13. 3D scatterplot for value iteration.

through the eighth iteration. Therefore, considering the high efficiency of the system's operation, the strategy iteration method is the best choice. Compared to traditional planning strategies, this method has more time consumption but better management efficiency. Thus, it can bring more economic benefits. Accordingly, the planning method based on reinforcement learning, as proposed in this paper has higher accuracy.

5. Conclusions

5.1. Conclusions

In this study, reinforcement learning was applied to a car transfer planning system for parking lots. And it is a suitable choice to use reinforcement learning to solve the transferring problem. The vehicle movement problem is an operations research problem. According to the properties of reinforcement learning, it will play a great role in saving cost, improving efficiency and providing a vehicle moving strategy that maximizes benefits. And it is suitable for many application scenarios, especially in the effort to determine out the optimal strategy and maximize the expected value of the return while achieving excellent performance. The system focuses on as many future rewards as possible and has more potential for development than the traditional method of manual transferring planning.

In the experiments discussed in this paper, both the strategy iteration methods and the value iteration converge to the optimal strategy and optimal value state. The former uses the value function of the previous strategy to start the calculation when performing strategy evaluation. This will effectively increase the speed of convergence of strategy evaluation. It consumes a significantly large amount of time on strategy evaluation. In contrast, although value iteration reduces the time spent on strategy

Table 2. Formatting of forms for uploading to the system.

Number of iterations	Strategy iteration runtime (s)	Value iteration run length (s)
1	78.69	184.31
2	77.22	182.11
3	75.92	181.79
4	74.39	181.63
5	69.82	180.55
6	65.43	180.43
7	63.21	179.37
8	60.63	177.91
9	60.55	177.83
10	60.54	177.82

evaluation, it converges slowly. Since strategy iteration ensures the optimization of the strategies, it is better to use strategy iteration for this system.

5.2. Future work

In this paper, a reinforcement learning method based on dynamic planning is detailed. It serves to find the optimal moving strategy and the optimal value return for parking lots. The system now completed in this study has been able to effectively solve this vehicle moving problem. However, due to the existence of objective factors such as the algorithm time complexity being high, the system has a long waiting time for the calculation of large-capacity matrices. Therefore the model needs to be further improved. Moreover, this system solves the operations research problem of how to move vehicles between two locations at the end of a day's business in a parking lot. The scope of the current application is relatively narrow. In the future work, more kinds of situations will be considered, such as how to find the optimal strategy and the optimal state value of the current state at any time during a day's business.

Use of AI tools declaration

The authors declare that they have not used artificial intelligence tools in the creation of this article.

Conflict of interest

The authors declare that there is no conflict of interest.

References

1 J. Yang, F. Lin, C. Chakraborty, K. Yu, Z. Guo, A. T. Nguyen, et al., A Parallel Intelligence-driven Resource Scheduling Scheme for Digital Twins-based Intelligent Vehicular Systems, *IEEE Transact. Intell. Vehicles*, **8** (2023), 2770–2785. https://doi.org/10.1109/TIV.2023.3237960

2 A. Thakur, Car rental system, *Int. J. Res. Appl. Sci. Eng. Technol.*, **9** (2021), 402–412. https://doi.org/10.22214/ijraset.2021.36339

3 X. Zhu, F. Ma, F. Ding, Z. Guo, J. Yang, K. Yu, A Low-latency Edge Computation Offloading Scheme for Trust Evaluation in Finance-Level Artificial Intelligence of Things, *IEEE Int. Things J.*, (2023), 1. https://doi.org/10.1109/JIOT.2023.3297834

4 J. Yang, Z. Guo, J. Luo, Y. Shen, K. Yu, Cloud-Edge-End Collaborative Caching Based on Graph Learning for Cyber-Physical Virtual Reality, *IEEE Systems J.*, (2023), 3262255. https://doi.org/10.1109/JSYST.2023.3262255

5 Z. Shen, F. Ding, Y. Yao, A. Bhardwaj, Z. Guo, K. Yu, A Privacy-Preserving Social Computing Framework for Health Management Using Federated Learning, *IEEE Transact. Comput. Soc. Syst.*, **10** (2023), 1666–1678. https://doi.org/10.1109/TCSS.2022.3212864

6 Z. Guo, Q. Zhang, F. Ding, X. Zhu, K. Yu, A Novel Fake News Detection Model for Context of Mixed Languages Through Multiscale Transformer, *IEEE Transact. Comput. Soc. Syst.*, (2023), 1–11. https://doi.org/10.1109/TCSS.2023.3298480

7 D. Meng, Y. Xiao, Z. Guo, A. Jolfaei, L. Qin, X. Lu, et al., A data-driven intelligent planning model for UAVs routing networks in mobile Internet of Things, *Comput. Commun.*, **179** (2021), 231–241. https://doi.org/10.1016/j.comcom.2021.08.014

8 M. Wen, R. Lin, H. Wang, Y. Yang, Y. Wen, L. Mai, et al., Large sequence models for sequential decision-making: A survey, *Front. Computer Sci.*, **17** (2023), 176–349. https://doi.org/10.1007/s11704-023-2689-5

9 J. Huang, F. Yang, C. Chakraborty, Z. Guo, H. Zhang, L. Zhen, et al., Opportunistic capacity based resource allocation for 6G wireless systems with network slicing, *Future Gener. Comput. Syst.*, **140** (2023), 390–401. https://doi.org/10.1016/j.future.2022.10.032

10 Z. Guo, Y. Shen, A. K. Bashir, M. Imran, N. Kumar, D. Zhang, et al., Robust Spammer Detection Using Collaborative Neural Network in Internet-of-Things Applications, *IEEE Int. Things J.*, **8** (2021), 9549–9558. https://doi.org/10.1109/JIOT.2020.3003802

11 Z. Guo, K. Yu, K. Konstantin, S. Mumtaz, W. Wei, P. Shi, et al., Deep Collaborative Intelligence-driven Traffic Forecasting in Green Internet of Vehicles, *IEEE Transact. Green Commun. Network.*, **7** (2023), 1023–1035. https://doi.org/10.1109/TGCN.2022.3193849

12 S. Cheng, C. Liu, Y. Guo, R. Arcucci, Efficient deep data assimilation with sparse observations and time-varying sensors, *J. Comput. Phys.*, **496** (2024), 112581. https://doi.org/10.1016/j.jcp.2023.112581

13 S. Cheng, I. C. Prentice, Y. Huang, Y. Jin, Y. K. Guo, R. Arcucci, Data-driven surrogate model with latent data assimilation: Application to wildfire forecasting, *J. Comput. Phys.*, **464** (2022), 111302. https://doi.org/10.1016/j.jcp.2022.111302

14 C. Zhang, S. Cheng, M. Kasoar, R. Arcucci, Reduced-order digital twin and latent data assimilation for global wildfire prediction, *Nat. Hazards Earth Syst. Sci.*, **23** (2023), 1755–1768. https://doi.org/10.5194/nhess-23-1755-2023

15 L. Wang, Q. Liu, W. Ma, Optimization of dynamic relocation operations for one-way electric carsharing systems, *Transport. Res. Part C Emerg. Technol.*, **101** (2019), 55–69. https://doi.org/10.1016/j.trc.2019.01.005

16 K. Huang, K. An, G. H. de Almeida Correia, J. Rich, W. Ma, An innovative approach to solve the carsharing demand-supply imbalance problem under demand uncertainty, *Transport. Res. Part C Emerg. Technol.*, **132** (2021), 103369. https://doi.org/10.1016/j.trc.2021.103369

17 B. B. Oliveira, M. A. Carravilla, J. F. Oliveira, Fleet and revenue management in car rental companies: A literature review and an integrated conceptual framework, *Omega*, **71** (2017), 11–26. https://doi.org/10.1016/j.omega.2016.08.011

18 J. Wang, L. Kang, Y. Liu, Optimal scheduling for electric bus fleets based on dynamic programming approach by considering battery capacity fade, *Renewable Sustainable Energy Rev.*, **130** (2020), 109978. https://doi.org/10.1016/j.rser.2020.109978

19 N. Wang, J. Guo, X. Liu, Y. Liang, Electric vehicle car-sharing optimization relocation model combining user relocation and staff relocation, *Transport. Letters*, **13** (2021), 315–326. https://doi.org/10.1080/19427867.2020.1728843

20 Z. Hao, L. He, Z. Hu, J. Jiang, Robust vehicle pre-allocation with uncertain covariates, *Product. Operat. Manag.*, **29** (2022), 955–972. https://doi.org/10.1111/poms.13143

21 N. Wang, Y. Gao, H. Zhao, C. K. Ahn, Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle, *IEEE Transact. Neural Networks Learn. Syst.*, **32** (2022), 3034–3045. https://doi.org/10.1109/TNNLS.2020.3009214

22 N. Wang, Y. Gao, X. Zhang, Data-driven performance-prescribed reinforcement learning control of an unmanned surface vehicle, *IEEE Transact. Neural Networks Learn. Syst.*, **32** (2021), 5456–5467. https://doi.org/10.1109/TNNLS.2021.3056444

23 N. Wang, Y. Gao, C. Yang, X. Zhang, Reinforcement learning-based finite-time tracking control of an unknown unmanned surface vehicle with input constraints, *Neurocomputing*, **189** (2022), 108600. https://doi-org ssl.8611.top/10.1016/j.apacoust.2021.108600

24 G. Liu, W. Deng, X. Xie, L. Huang, H. Tang, Human-Level Control Through Directly Trained Deep Spiking *Q*-Networks, *IEEE Transact. Cybernet.*, (2022). https://doi.org/10.1109/TCYB.2022.3198259

25 X. B. Peng, E. Coumans, T. Zhang, T. W. Lee, J. Tan, S. Levine, Learning agile robotic locomotion skills by imitating animals, *arXiv preprint arXiv:2004.00784*, (2004). https://doi.org/10.48550/arXiv.2004.00784

26 S. Zhang, Y. Li, Q. Dong, Autonomous navigation of UAV in multi-obstacle environments based on a Deep Reinforcement Learning approach, *Appl. Soft Comput.*, **105** (2022), 108194. https://doi.org/10.1016/j.asoc.2021.108194

27 Y. Oh, J. Shin, E. Yang, S. J. Hwang, Model-augmented prioritized experience replay, in *International Conference on Learning Representations*, (2021).

28 T. Li, Z. Wang, G. Yang, Y. Cui, Y. Chen, X. Yu, Semi-selfish mining based on hidden Markov decision process, *Int. J. Intell. Syst.*, **36** (2021), 3596–3612. https://doi.org/10.1002/int.22428

29 G. Kalnoor, G. Subrahmanyam, A review on applications of Markov decision process model and energy efficiency in wireless sensor networks, *Proced.a Computer Sci.*, **167** (2020), 2308–2317. https://doi.org/10.1016/j.procs.2020.03.283

30 G. P. Antonio, C. Maria-Dolores, Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections, *IEEE Transact. Vehicular Technol.*, **71** (2022), 7033–7043. https://doi.org/10.1109/TVT.2022.3169907

31 C. Li, Study on theory of the Grey Markov Chain method and its application, in *The Proceedings of the Multiconference on "Computational Engineering in Systems Applications"*., **72** (2006), 1742–1746. https://doi.org/10.1109/CESA.2006.4281919

32 F. Y. Wang, H. Zhang, D. Liu, Adaptive dynamic programming: An introduction, *IEEE Comput. Intell. Magaz.*, **4** (2009), 39–47. https://doi.org/10.1109/MCI.2009.932261

33 Z. Guo, L. Tang, T. Guo, K. Yu, M. Alazab, A. Shalaginov, Deep Graph neural network-based spammer detection under the perspective of heterogeneous cyberspace, *Future Gener. Comput. Syst.*, **117** (2021), 205–218. https://doi.org/10.1016/j.future.2020.11.028

34 K. K. McDill, C. D. Minchew, Waveform selection for an electrically enhanced seine for use in harvesting channel catfish Ictalurus punctatus from ponds, *J. World Aquaculture Soc.*, **32** (2001), 342–347. https://doi.org/10.1111/j.1749-7345.2001.tb00458.x

35 D. Liu, S. Xue, B. Zhao, B. Luo, Q. Wei, Adaptive dynamic programming for control: A survey and recent advances, *IEEE Transact. Syst. Man Cybernet. Syst.*, **51** (2022), 142–160. https://doi.org/10.1109/TSMC.2020.3042876

36 S. V. Lapensée-Rankine, Dynamic Programming Insights from Programming Contests, *Appalachian State University*, (2021).

37 A. Alla, M. Falcone, D. Kalise, An efficient policy iteration algorithm for dynamic programming equations, *SIAM J. Sci. Comput.*, **37** (2015), A181–A200. https://doi.org/10.1002/pamm.201310226

38 D. Xiang, H. Lin, J. Ouyang, D. Huang, Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot, *Sci. Rep.*, **12** (2022), 13273. https://doi.org/10.1038/s41598-022-17684-0

39 F. Ye, J. Perrett, L. Zhang, Y. Laili, Y Wang, A self-evolving system for robotic disassembly sequence planning under uncertain interference conditions, *Robot. Computer-Integr. Manufact.*, **78** (2022), 102392. https://doi.org/10.1016/j.rcim.2022.102392

40 I. A. Zamfirache, R. E. Precup, R. C. Roman, E. M. Petriu, Policy iteration reinforcement learning-based control using a grey wolf optimizer algorithm, *Inform. Sci.*, **585** (2022), 162–175. https://doi.org/10.1016/j.ins.2021.11.051

AIMS Press