IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 September 2023, accepted 6 October 2023, date of publication 10 October 2023, date of current version 16 October 2023.
Digital Object Identifier 10.1109/ACCESS.2023.3323584

RESEARCH ARTICLE

Dynamic Service Composition Method Based on Zero-Sum Game Integrated Inverse Reinforcement Learning

YUAN YUAN, YUHAN GUO, AND WANQING MA
School of Science, Zhejiang University of Science and Technology, Hangzhou 310023, China
Corresponding author: Yuhan Guo (yuhan.guo@zust.edu.cn)

This work was supported by the Liaoning Provincial Social Science Planning Fund Project under Grant L21CGL005.

ABSTRACT Automatically generating service composition solutions that meet user application requirements is one of the hot research topics in the field of service composition in the context of Web service big data. To address the challenges of accurately obtaining reward function values and significant increase in time complexity when dealing with large-scale data in the context of reinforcement learning-based service composition, this paper proposes a novel approach that combines zero-sum game and inverse reinforcement learning. The proposed method models the service composition problem as a Markov Decision Process (MDP) and dynamically adjusts the service composition solution by solving for the optimal policy. By leveraging runtime records of service composition operations, we generate an expert experience dataset and develop a novel inverse reinforcement learning algorithm based on the integration of zero-sum game principles. Experimental results demonstrate that the proposed method effectively reduces the dependence of the inverse reinforcement learning algorithm on the quality of expert experience data and reduces the time cost of service composition.

INDEX TERMS Service composition, inverse reinforcement learning, zero-sum game, quality of service.

I. INTRODUCTION

The rapid proliferation of diverse Web services across various domains on the Internet has led to an explosive growth of Quality of Service (QoS) data due to dynamic changes in network environments and client contexts [1], [2]. This abundance of Web services, along with associated data such as service descriptions, service QoS data, and service composition runtime records, forms what is known as Web service big data. In the context of Web service big data, traditional methods for service discovery and selection face significant time costs in generating service composition solutions. Moreover, these methods lack the capability to re-plan service composition solutions, resulting in uncertainties regarding the successful and efficient execution of composite services [3], [4]. Consequently, there is a pressing need to automatically generate service composition solutions based on user application requirements, which represents a key focus area in the field of service composition [5].

Web services operate in dynamic network environments where factors such as network bandwidth, access point locations, and time can cause fluctuations in service performance, primarily reflected in the dynamic changes in QoS [6], [7]. Even for the same user in the same network environment, uncertainties in server loads from service providers can lead to QoS fluctuations in Web services for different invocation times [8]. Furthermore, abnormal conditions such as gateway errors, network errors, and server errors can easily result in service invocation failures. Reinforcement learning-based service composition methods enable adaptive adjustments of service composition. However, reinforcement learning methods can only yield optimal policies when the ''reward function values'' for each action taken by the agent can be accurately obtained. Due to the dynamic nature of network environments and variations in client contexts, the QoS values of candidate services can change at any moment, making

The associate editor coordinating the review of this manuscript and approving it for publication was Shu Xiao.

it difficult to provide a deterministic reward function value based on QoS data at a specific time or historical QoS data. Therefore, in such scenarios, the agent cannot obtain accurate reward functions and optimal policies through reinforcement learning. Additionally, as the scale of services continues to expand to a massive number of Web services, the increase in dimensions of actions and states leads to an exponential growth in the number of learning parameters. This results in a noticeable performance decline in reinforcement learning-based service composition methods, which lack fast convergence speed and struggle to meet user requirements for both service composition quality and time costs.

The primary objective of this paper is to explore the issue of Web service composition within the realm of Web service big data. In light of the aforementioned problem, we present a novel approach to dynamic service composition by integrating zero-sum game theory and inverse reinforcement learning. Through extensive simulation experiments, we demonstrate the efficacy of our proposed method in mitigating the reliance on expert experience data quality and enhancing the overall quality of composite services. Furthermore, our method proves effective in reducing the time required for service composition replanning, particularly when confronted with a growing number of candidate services and state nodes. The contributions of this paper are shown below:

- First, we model the service composition problem as a Markov Decision Process (MDP) and utilizes observed service invocation data and expert experience to dynamically adjust and generate new service composition solutions using inverse reinforcement learning algorithms, thereby achieving dynamic service composition. In the context of web services big data environment, inverse reinforcement learning is indeed a promising approach to address the challenges of computational complexity and increased parameters in traditional reinforcement learning methods. It can provide more effective solutions to meet user requirements for service composition quality and time cost.
- Second, in order to tackle the challenge of the accuracy of inverse reinforcement learning algorithms being influenced by the quality of expert experience data, we integrate the concept of zero-sum game and propose a zero-sum game integrated inverse reinforcement learning algorithm. This algorithm involves adjusting weight coefficients and iteratively learning from expert experience data to generate an optimal strategy that surpasses the expert policy. By doing so, we effectively reduce the reliance of the inverse reinforcement learning algorithm on the quality of expert experience data, and furthermore, decrease the time required for service composition re-planning.

The remainder of this paper is organized as follows. Section II reviews related works. Section III introduces a service composition framework based on inverse reinforcement learning. Section IV presents a service composition model based on MDP. Section V proposes a zero-sum game integrated inverse reinforcement learning algorithm. Section VI presents experimental results. Finally, Section VII concludes this paper.

II. RELATED WORK

Dynamic service composition problem has been studied extensively. Dynamic service composition methods consider the volatility of QoS attributes of Web services, and dynamically adjust the service composition scheme to meet the global QoS constraints of users during the operation of composite services [9]. Dynamic service composition methods can be divided into strategy-based approaches, replacement-based approaches and reinforcement learning-based approaches.

The strategy-based approach defines maintenance policies in advance to deal with common abnormal situations that may occur during the operation of composite services by analyzing the collected historical data of the operation of composite services [10], [11]. In the actual running process, unknown exceptions may occur at any time, leading to the adjustment strategy defined in advance for known exceptions is far from enough. Therefore, these strategy-based approaches are not universally applicable.

In order to further improve the universality, some researchers adopt the replacement-based approach, that is, to replace one or more similar or better candidate services with the original poorly performing services, the essence of which is to establish a redundancy mechanism for service composition [12], [13], [14]. In [14], the authors pointed out that most of the current studies are adaptive methods for service selection module, that is, they only consider the abnormal situation of service composition runtime, but do not consider the service discovery module, that is, the dynamic change of the candidate service set. In [15], the authors introduced the concept of abstract proxy services in a variability-supporting service composition language, namely VxBPEL, and provides a mechanism to support variation design and dynamic binding for unplanned changes at run time. In [16], the authors constructed a QoS Dependency Graph to capture QoS variations, and achieve adaptive composition with dynamic QoS satisfactions. In [17], the authors used a recurrent neural network to predict the QoS, which can be well adapted to a dynamic network environment. The above replacement-based approach will inevitably increase the extra cost of the system when the redundancy increases. Especially, when the number of Web services with the same or similar functions is large, the cost for the system is often unacceptable.

In recent years, some researchers have applied reinforcement learning to dynamic service composition to meet the application requirements of service composition reprogramming. Reinforcement learning is a machine learning method that learns the optimal strategy by interacting with the external environment, which has been widely used in the field of service composition. In [18], [19], and [20], the authors modeled a trustworthy composite service adaptive control process using Markov Decision Process (MDP), and proposed an adaptive control algorithm based on Q-learning, which supports optimal strategy action selection to achieve adaptive adjustment and dynamic construction of the composite service. Traditional reinforcement learning algorithms are based on a discrete and finite system state space and action space, and usually adopt value table query to learn the value function, which is effective for problems with small data size. However, in practical applications, when facing massive Web service data, as the dimensions of actions and states increase, the number of learning parameters increases exponentially, and traditional reinforcement learning methods do not have fast convergence speed, which cannot meet the efficiency requirements of service composition.

To address the above issues, in [21] and [22], the authors proposed a dynamic service composition method based on automatic hierarchical reinforcement learning technology, which divides the hierarchical structure of the service composition, effectively improving the efficiency of hierarchical division and accelerating the learning speed, compared to manual hierarchical division when encountering new scenarios. In [23], [24], [25], and [26], the authors integrated the multi-agent system mechanism and the virtual action process in the game theory field into the service composition model on the basis of reinforcement learning, which effectively improved the convergence speed of the algorithm when facing large-scale service scenarios. In [27], the authors introduced a novel approach that aimed to achieve greater data efficiency by saving the experience data and using it in aggregate to make updates to the learned strategy. In [28], the authors proposed a deep reinforcement learning-based composition approach based on a parallel flock-based service discovery algorithm. In [29], a WSCMDP model is proposed to solve the large-scale service composition within a dynamically changing environment, by combining RL with skyline computing.

There are still some problems for the above research methods in the current application context, mainly manifested in the following aspects:

(1) The reinforcement learning-based service composition method can only obtain the optimal strategy when the ‘‘reward function value’’ of each step action learned by the agent can be accurately obtained in the environment. In big data environment, the QoS value of candidate services may change at any time, making it difficult to give an accurate reward function value based on the QoS data of a certain moment or historical QoS data alone, thus making it difficult to obtain the optimal strategy.

(2) As the number of action and state dimensions increases, the number of learning parameters increases exponentially. The reinforcement learning-based service composition method will show significant performance degradation and does not have a fast convergence speed, making it difficult to meet the user's requirements for service quality and service composition time.

III. THE FRAMEWORK OF DYNAMIC SERVICE COMPOSITION BASED ON INVERSE REINFORCEMENT LEARNING

The paper models the dynamic service composition problem as an MDP and employs inverse reinforcement learning algorithm to dynamically adjust service composition solutions based on monitored service invocation and expert experience data, thus ensuring the normal operation of composite services. A dynamic service composition framework based on inverse reinforcement learning algorithm is proposed in this paper, as shown in Figure 1. The historical QoS information of service composition, user preferences and global constraints are taken as the runtime records of service composition. Each execution of service composition generates a service composition execution record. The successful runtime records of service composition are used as expert experience data. Based on historical QoS data, user preferences and global constraints, a MDP-SCM is generated to model the dynamic service composition. During the service composition operation, the service composition execution engine sends real-time QoS information of monitored service components to the QoS data analysis module; the QoS data analysis module analyzes the current service composition solution based on user preferences, global QoS constraints and historical QoS information, and determines whether adjustment is needed. Once adjustments are deemed necessary, real-time QoS information, user preferences, and global constraints are sent to the optimal strategy solving module. This module utilizes expert experience data to solve the optimal strategy based on the zero-sum games integrated inverse reinforcement learning algorithm, dynamically adjusting the service composition solution, and sending the new service composition solution to the service composition execution module to ensure the normal operation of composite services.

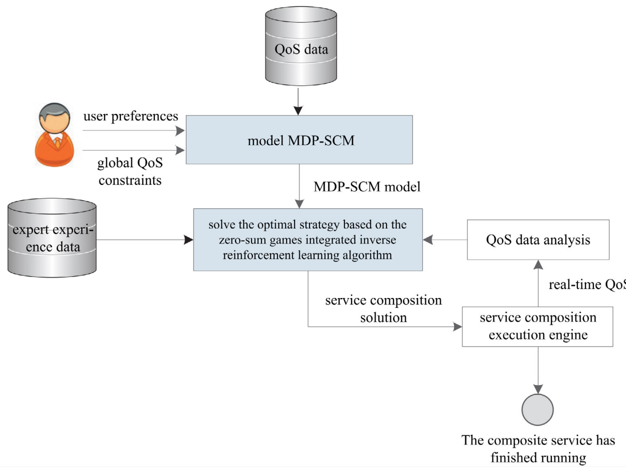
QoS data
user preferences
global QoS constraints
model MDP-SCM
MDP-SCM model
expert experience data
solve the optimal strategy based on the zero-sum games integrated inverse reinforcement learning algorithm
QoS data analysis
real-time QoS
service composition solution
service composition execution engine
The composite service has finished running

FIGURE 1. The framework of inverse reinforcement learning-based dynamic service composition.

IV. SERVICE COMPOSITION MODEL BASED ON MDP

This paper models the service composition problem as a Markov Decision Process (MDP) and proposes an MDP-based service composition model, MDP-SCM. First of all, this paper gives a formal description of MDP-SCM, and gives the calculation method of state values. Then, the calculation method of state values and the solution method of the optimal strategy are given.

A. FORMAL DESCRIPTION OF MDP-SCM

The MDP-SCM model is represented by a five-tuple, MDP-SCM $= (t, S, A, P, R)$, where:

1) t represents the decision stage, $t \in \{1, 2, \cdots, n\}$, where n denotes the number of service classes.
2) S represents the collection of all possible states that a service composition may experience during its execution from the start state to the end state. When we say $s \in S$, we mean that it is the data about the environment obtained from sensors or other perception devices, and it is represented as an element belonging to S. The states are represented in the form of vectors so that computers can understand and process them. s_0 represents the start state, and s_{ni} represents the end state.
3) A represents the set of candidate services, namely all services that may be employed during the service combination process. $A(t)$ represents the set of candidate actions that can be selected at the stage t, i.e., the set of candidate services in service class S_t, where a_{tj} denotes the jth candidate service in service class t.
4) P represents the probability of state transition, $P_a(s, s')$ represents the probability of transitioning from state s to state s' after executing service a under state s.
5) R represents the reward function, where $R_a(s, s')$ denotes the immediate reward that the system receives when executing service a in state s and transitioning to state s$'$, commonly denoted as r.

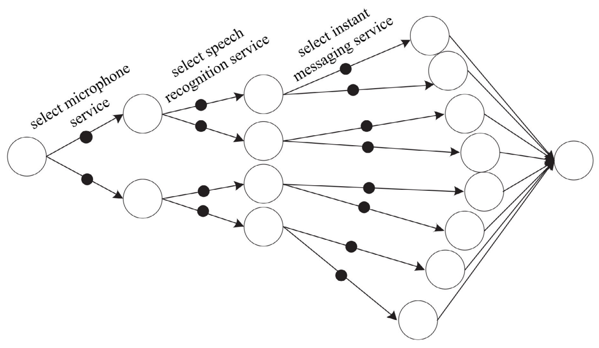
select microphone service
select speech recognition service
select instant messaging service

FIGURE 2. MDP-SCM of a speech communication service composition.

Figure 2 is a schematic diagram of the MDP-SCM for a speech communication service composition. To facilitate description, it is assumed that there are two candidate services for each service class in this service composition. The circles in the diagram represent service composition states, which depict the relevant environment parameter vectors perceived by the system after the execution of a specific service. The solid black dots represent executable actions (i.e., selectable services). Each path from the start state to the end state in the diagram represents a service composition scheme. There are a total of eight paths in the diagram, indicating that there are eight possible service composition schemes for this speech communication service composition. One of these service composition schemes can be represented in the following form:

$$\tau_1 = (s_0, a_{11}, s_{11}, a_{21}, s_{21}, a_{31}, s_{31})$$

In the context of speech communication service composition, τ_1 denotes the transition from state s_0 to state s_{31}. When it is in state s_0, the next step is to execute action a_{11}, which refers to invoking the first candidate service of the first service class. Then the state transitions to s_{11}, which describes the environment data obtained from sensors or other sensing devices after the completion of action a_{11}. Next, action a_{21} is executed, leading to a state transition to s_{21}. After that, action a_{31} is performed, resulting in a state transition to s_{31}. The entire service composition process concludes, reaching state s_{31}.

B. STATE VALUE CALCULATION OF MDP-SCM

To help understand the calculation process of state values, let's take an example of a service composition with three service classes in the MDP-SCM. The action sets are denoted as $A(1)$, $A(2)$, and $A(3)$, respectively. $A(1) = \{a_{11}, a_{12}, a_{13}, a_{14}\}$, $A(2) = \{a_{21}, a_{22}, a_{23}\}$, and $A(3) = \{a_{31}, a_{32}, a_{33}, a_{34}\}$ represent the actions contained in each action set, i.e., the candidate services provided by each service class. The global constraints of the service composition are: the response time $C^r \leq 2000\ ms$, and the service price $C^p \leq 600\ yuan$. The user preferences are 0.6 and 0.4, respectively.

The prices of candidate services in each action set are $A(1) = \{182, 190, 270, 380\}$, $A(2) = \{140, 230, 180\}$, $A(3) = \{120, 130, 125, 122\}$. The response time is dynamically affected by the network environment. To reflect this, we obtained the call records of each candidate service for 10 times and represented them as a matrix. Each element x_{ij} in the matrix represents the response time of the ith candidate service for the jth call, with a unit of milliseconds. If the call fails, the corresponding element is denoted as -1. The response time matrix of action set $A(1)$ is shown below:

$$\begin{pmatrix} 417 & -1 & 429 & 416 & 421 & -1 & 414 & 430 & -1 & 421 \\ 320 & 289 & 307 & 322 & 286 & 295 & 313 & 317 & 321 & 296 \\ 390 & 362 & 371 & 380 & 310 & 373 & 367 & -1 & -1 & -1 \\ 434 & 405 & 421 & -1 & 400 & 403 & 413 & -1 & -1 & -1 \end{pmatrix}$$

The matrix has 4 rows and 10 columns, where the 1st row represents the response time of candidate service a_{11}. Some elements in the matrix may be -1, indicating the failure of the corresponding candidate service in specific calls.

For example, in the response time matrix of action set $A(1)$, $x_{11} = 417$ represents the response time of a_{11} for the 1st call is 417 milliseconds. $x_{37} = -1$ represents the 7th call of a_{13} fails.

The response time matrix of action set $A(2)$ is shown below:

$$\begin{pmatrix} 417 & 239 & -1 & 641 & 521 & -1 & -1 & -1 & -1 & -1 \\ 201 & 277 & 307 & 121 & 186 & 395 & 395 & 417 & 121 & 396 \\ 450 & 242 & 431 & -1 & 240 & 433 & 433 & -1 & -1 & -1 \end{pmatrix}$$

The response time matrix of action set $A(3)$ is shown below:

$$\begin{pmatrix} 173 & 133 & 121 & 146 & -1 & -1 & 119 & 128 & -1 & -1 \\ 90 & 109 & 97 & 112 & 86 & 95 & 113 & 107 & 111 & 106 \\ 112 & 132 & 101 & 130 & -1 & 122 & 109 & 113 & -1 & -1 \\ 174 & 153 & 124 & -1 & 140 & 133 & 113 & -1 & -1 & -1 \end{pmatrix}$$

To begin with, we calculate the median of response times for successful service calls as the response time of each candidate service. For action set $A(1)$, its response time vector is given by $v_1^r = \{421, 307, 371, 408\}$. For action set $A(2)$, its response time vector is $v_2^r = \{417, 277, 431\}$, while for action set $A(3)$, its response time vector is $v_3^r = \{128, 106, 113, 133\}$. Therefore, the average response time of action set $A(1)$ is $\mu_1^r = 376.75$, that of action set $A(2)$ is $\mu_2^r = 375$, and that of action set $A(3)$ is $\mu_3^r = 120$. Next, we decomposed the global constraint on response time C_r into sub-constraints C_i^r, where n denotes the number of action sets. The formula for computing these constraints is as follows:

$$C_i^r = C^r \times \frac{\mu_i^r}{\sum_{i=1}^{n} \mu_i^r} \tag{1}$$

Based on formula (1), we obtain $C_1^r = 864.35$, $C_2^r = 860.43$, $C_3^r = 275.31$, which are the sub-constraints on response time for each action set. The actual response time of action a_{ij} is denoted as v_{ij}^r. The calculation formula for the response time status s_i^r of the service composition after executing action a_{ij} is given by:

$$s_i^r = \begin{cases} 1, & if\ v_{ij}^r \leq 0.85C_i^r \\ 2, & if\ 0.85C_i^r < v_{ij}^r \leq C_i^r \\ 3, & if\ C_i^r < v_{ij}^r \leq C^r \\ 4. & if\ others \end{cases} \tag{2}$$

In the case of service call failure, we set $s_i^r = 4$. The coefficient 0.85 and the state value of service composition response time status are both artificially set based on experience, and can be adjusted according to different actual situations. The calculation method for the service composition price status s_i^p is the same as described above. Finally, we can obtain the calculation formula for service status s_i as follows, where w_r and w_p represent user preferences for response time and service price, respectively.

$$s_i = s_i^r \times W_r + s_i^p \times w_p \tag{3}$$

Suppose that the final selected service for action $A(1)$ in the first decision stage is candidate service a_{13}, with an actual response time of 750ms, thus $s_1^r = 2$. If $C_1^p = 271.98$ and the price of a_{13} is 182, then $s_1^p = 1$. Finally, we can obtain $s_1 = 1.6$. Depending on the application scenario, the number of states that can be partitioned in equation (2) may vary. Furthermore, since user preferences may vary for different service composition plans in equation (3), different state spaces can be obtained for different application scenarios.

C. OPTIMAL STRATEGY OF MDP-SCM

The core problem of MDP-SCM is to find a global optimal strategy π^* that maximizes the cumulative reward. The function $\pi(s)$ describes the action that the system will choose in state s. For MDP-SCM, the selection of the optimal service component depends on global constraints and the historical QoS records of each candidate service. The process of strategy generation for MDP-SCM is illustrated in Figure 3.

As shown in Figure 3, at the start state s_0, the agent selects and executes the optimal service component ws_1 based on the historical QoS information H_1 of each candidate service in the first service class. The state transitions to s_1, the runtime QoS of ws_1 is stored as historical information in H_1, and the immediate reward r_1 is calculated. Next, based on the historical QoS information H_2 of each candidate service in the second service class, the agent selects and executes the optimal service composition component ws_2, and the state transitions to s_2. This process is repeated until the optimal service components for all service classes in the workflow are selected.

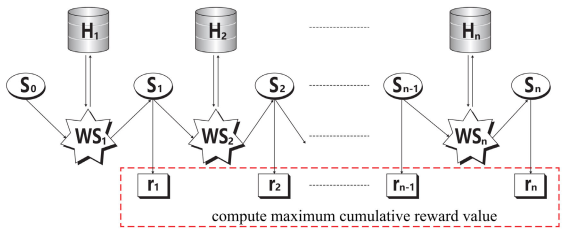

FIGURE 3. The strategy generation process of MDP-SCM.

The fundamental idea of reinforcement learning is to improve the model by perceiving and incorporating feedback information resulting from action execution during interaction with the environment, while considering the current system state to select the next action. When applying the basic concept of reinforcement learning to Web service composition problems, a service is executed at a certain system state, the system transitions to the successor state, and feedback information is obtained. Generally, the better the service quality, the higher the user satisfaction. Therefore, some calculation method based on QoS attribute values can be used to simulate the feedback information after service execution. The formula for calculating the reward function value r_a of the service a after execution can be obtained as follows:

$$r_a = \sum_{k=1}^{r} q_k \times w_k \tag{4}$$

Here, q_k represents the normalized value of the kth QoS attribute of service a, while w_k represents the user preference for the attribute k. A higher reward function value r_a after service execution indicates higher service quality. Therefore, the sequence of service selections that maximizes cumulative reward function value and satisfies user non-functional constraints can be considered as the optimal strategy of MDP-SCM, which generates an optimal Web service composition solution. The specific formula is given as follows:

$$\begin{aligned}\pi^* &= \underset{\pi}{\operatorname{argmax}} \sum_{s\in S} V^{\pi}(s)\\ &= \underset{\pi}{argmax} \sum_{s\in S} P_a(s,s')(R_a(s,s')+\gamma V^{\pi}(s'))\\ &\quad \forall k\in\{1,2,\cdots,r\}\ Cq_k \geq C_k \qquad (5)\end{aligned}$$

C_k represents the user's requirements for QoS attributes of composite services and Cq_k represents the QoS attributes of the composite service. Based on the real-time QoS information obtained during system operation, the current values of different QoS attributes of the composite service can be calculated.

MDP-SCM provides a reliable approach for selecting service components that satisfy user QoS constraints based on real-time QoS information. The selection process proceeds sequentially until all tasks are completed. Historical QoS information is used to estimate the expected return function value after executing each candidate service. At the ith service class of the workflow, the return function values of the preceding i services are updated based on the real-time QoS information of each service component, and the optimal strategy is determined based on the global constraints and the empirical return function values of the candidate services in the unexecuted service classes using equations (5).

V. THE OPTIMAL STRATEGY BASED ON ZSG-IRL

When using the inverse reinforcement learning (IRL) algorithm to find the optimal strategy for MDP-SCM, the saved records of successfully executed service combinations are processed. The state values are calculated based on the runtime QoS values, global constraints, user preferences, and other information in the service composition execution records using the method described in Section IV. An expert experience dataset is generated, which consists of decision trajectory data $\{\tau_1, \tau_2, \cdots, \tau_m\}$. m represents the number of samples, and each trajectory τ_i includes a sequence of states and actions and is represented as follows:

$$\tau_i = \langle s_0, a_1, s_1, a_2, \ldots, s_{n_i}\rangle$$

Here, n_i represents the number of transitions in the ith trajectory, i.e., the number of service calls. The trajectory τ_i represents the execution of service a_1 in state s_0, followed by a sequence of subsequent actions until the state transitions to s_{n_i}. The calculation of the return function value is based on the service composition records and is calculated using the method described in Section IV.

With the increasing Web services, IRL algorithm has shown higher levels of fitting accuracy and optimization precision compared to RL algorithms. However, IRL can only obtain a reward function by blindly learning from expert policies, without improving it. This limitation can impact the degree of approximation between the optimal and expert policies. Therefore, when applying IRL to solve MDP-SCM, the quality of expert experience data directly affects algorithmic performance. The expert experience data extracted from service composition records may contain service compositions that do not meet the constraints or are of low quality, which could reduce the overall quality of the expert data and subsequently affect the effectiveness of the IRL algorithm. To solve this problem, this paper proposes a novel IRL algorithm that applies zero-sum game theory to learn better strategies that minimize the influence of expert experience data's quality on the results.

A. BASIC PRINCIPLE OF ZSG-IRL

Mathematician John von Neumann introduced the concept of zero-sum games in his book "Theory of Games and Economic Behavior", providing a specific definition: if the total winnings of the participants in a game are always distributed in such a way that one participant's earnings are equal to the other participant's losses, the game is called a zero-sum game [30]. To establish a model for a zero-sum game, it is necessary to determine the strategy sets and corresponding payoffs of both players based on the description of the practical problem, and to construct a payoff matrix M to view the two participants as the row player and column player, respectively. In a zero-sum game, the two participants A and B have strategy sets $\Pi_A = \{a_1, a_2, \cdots, a_m\}$ and $\Pi_B = \{b_1, b_2, \cdots, b_n\}$, respectively. c_{ij} represents the payoff for player A when player A adopts strategy a_i and player B adopts strategy b_j (at this time, player B 's payoff is $-c_{ij}$), and the payoff matrix M for player A is shown in Table 1.

TABLE 1. The payoff matrix M of player A.

player B / player A	b_1	b_2	$\cdots$	b_n
a_1	c_{11}	c_{12}	$\cdots$	c_{1n}
a_2	c_{21}	c_{22}	$\cdots$	c_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_m	c_{m1}	c_{m2}	$\cdots$	c_{mn}

A choice of a row or column is typically considered a pure strategy, while many combinations of rows or columns constitute a mixed strategy. P represents the mixed strategy of the row player, Q represents the mixed strategy of the column player, and $M(P,Q) = P^T MQ$ represents the payoff of the row player under two mixed strategies. $M(P,j)$ and $M(i,Q)$ represent the expected payoff value when one player adopts a pure strategy and the other player adopts a mixed strategy. After multiple rounds of gameplay, the payoff of the row player and the loss of the column player will reach a fixed and unchanged state.

In solving zero-sum game problems for two players, the minimax method is often used, based on the core idea that both players assume a pessimistic attitude towards their opportunities for achieving greater gains. Specifically, player A assumes that player B will choose a strategy that maximizes player A's loss, while player B assumes that player A will choose a strategy that maximizes player B's loss. Given a payoff matrix M for the row player, player A hopes to select the position in the matrix with the highest possible payoff, while player B hopes to select the position with the lowest possible payoff. Specifically, for each row strategy that player A can choose, player B chooses the column with the lowest payoff in that row. Therefore, player A will choose the row strategy corresponding to the maximum payoff among the minimum payoffs of each row chosen by player B, which means choosing "maximum" among "minimum" and is represented as maxmin. For each column strategy that player B can choose, player A chooses the column with the highest payoff in that column. Therefore, player B will choose the column strategy corresponding to the minimum payoff among the maximum payoffs of each column chosen by player A, which means choosing "minimum" among "maximum" and is represented as minmax.

In zero-sum two-player games, if the given payoff matrix M has mixed strategies $P = (p_1, p_2, \cdots, p_m)$ and $Q = (q_1, q_2, \cdots, q_n)$, as well as a constant v that satisfies $\sum_{i=1}^{m} c_{ij} p_i \geq v$ for any j, and $\sum_{j=1}^{n} c_{ij} q_j \leq v$ for any i, then the strategy combination (P, Q) is the Nash equilibrium of the zero-sum two-player game [31]. Here, v is the expected payment received by player A in the equilibrium, which is also known as the value of the game. When $v > 0$, the linear programming method is commonly used to solve the Nash equilibrium of the game.

In IRL, the performance of a learned strategy π can be evaluated by the difference between the cumulative reward function value V^{π} and the cumulative reward function value V^{π_E} of an expert strategy π_E, that is, $V^{\pi} - V^{\pi_E}$. The fundamental idea of integrating zero-sum games into IRL is to find an optimal strategy π^* from the strategy set Π, such that the optimal strategy learned by the agent is as superior as possible to the expert strategy. Thus, the objective function can be expressed as follows:

$$U(\pi) = V^{\pi} - V^{\pi_E} \tag{6}$$

Equation (6) can be transformed into the following form:

$$U(\pi) = w^T \mu^{\pi} - w^T \mu^{\pi_E} \tag{7}$$

w^T represents the weight coefficient vector; μ^{π} and μ^{π_E} respectively denote the expected state vector under the strategy π and the expert strategy π_E. The corresponding strategy when maximizing Equation (7) is the optimal strategy π^*, which can be expressed as follows:

$$\pi^* = \underset{\pi}{argmax} \, (w^T \mu^{\pi} - w^T \mu^{\pi_E}) \tag{8}$$

As the weight w is adjustable and affects the result of the objective function, even if the optimal strategy has been obtained, changes in the weight w may cause the objective function to decrease, resulting in the learned strategy being inferior to the expert strategy. Therefore, it is necessary to consider the value of the weight w. The basic principle of ZSG- IRL algorithm is to minimize the objective function by changing the weight w, and then maximize the objective function by selecting the strategy π. This process is similar to the basic idea of two-player zero-sum games in game theory. Therefore, the strategy π determined by the learner and the weight coefficient w determined by the environment are regarded as two players. The game objective of strategy π is to maximize its own benefits, that is, to maximize the value of the objective function, while the game objective of weight coefficients w is to minimize their own losses, that is, to minimize the value of the objective function. Thus, the expression of the optimal strategy for ZSG- IRL can be obtained as follows:

$$\pi^* = arg \underset{\pi}{max} \underset{w \in W}{min} \, (w^T \mu^{\pi} - w^T \mu^{\pi_E}) \tag{9}$$

In the expression, W represents the domain of the weight coefficients w, where $W = \{w : w \in \mathbb{R}, \|w\|_1 = 1, w > 0\}$, and $\|w\|_1$ denotes the L1 norm of the weight vector w.

B. ZSG-IRL ALGORITHM

As shown in Algorithm 1, ZSG_IRL consists of the following steps: (1) calculating the expert strategy's feature expectation vector based on expert experience data, (2) using iterative IRL algorithm to obtain the optimal strategy under different weight coefficients and the corresponding feature expectation vector, and (3) updating the weight coefficients and returning to step (2) until reaching the maximum iteration number.

When the dimension of the game matrix M is large, the computational cost of solving it by linear programming is wasteful due to the large number of iterations generated by the iterative process, which equals the number of policies generated. In multiple repeated games, the multiplicative weight method is used to solve the problem of a large or unknown game matrix. The idea behind the multiplicative weight method is to take one player as the update criterion, select an initial mixed strategy for the row player in the first round of the game, and then calculate the corresponding new mixed strategy for the row player based on certain multiplication rules in each subsequent round. Through this process, the row player is updated and the weight is updated by multiplication rules. Each column of the game matrix M is taken as the column player's strategy choice, and the optimal strategy represents the feature expectation vector $\mu^{\pi_j} - \mu^{\pi_E}$ under the current strategy condition. The new strategy is obtained based on the strategy iteration method. A game process mainly studies its game matrix, where the game matrix in this paper is established based on the gains and losses of the learner. The expression of the element in the ith row and jth column of the

Algorithm 1 The Zero-Sum Game Integrated Inverse Reinforcement Learning Algorithm

```
Input: State space $S$;
       Action space $A$;
       Expert space $\pi_E$;
       Maximum iteration number $T$;
       Parameter $\varepsilon$.
Output: Optimal strategy $\pi^*$.
1. According to $\pi_E$, compute the expected state vector $\mu^{\pi_E}$
2. Set $\beta = \left(1 + \sqrt{\frac{2\ln}{T}}\right)^{-1}$, $d$ denotes the number of states
3. Initialize $w^1$, set $w_i^1 = 1$, where $i = 1, 2, \cdots, d$
4. Initialize $\pi^*$
5. for ($t = 1; t \leq T; t++$) do
6.     Normalize weight $w$, set $w_i^t = \frac{w_i^t}{\sum_{i=1}^d w_i^t}$
7.     $\pi^1$ = randomly generated strategy
8.     $\mu^1$ = feature expectation of $\pi^1$
9.     for ($j = 1, t(j) > \varepsilon, j++$) do
10.        solve $t(j) = max_{w:\|w\|_2 \leq 1} min_{k \in \{1,2,\cdots,j\}} w^T \left(\mu^k - \mu^{\pi_E}\right)$
           s.t. $\|w\|_2 \leq 1$, get $w^j$
11.        $\pi^{j+1}$ = the optimal strategy in $\langle S, A, R(s) = w^{jT} s\rangle$
12.        compute $\mu^{j+1}$
13.    end for
14.    if ($t(j) \geq t$)
15.        $\pi^* = \pi^{j+1}$
16.    end if
17.    compute the new weight $w^{t+1}$:
           $w_i^{t+1} = w_i^t \frac{\beta^{M(i,j)}}{\sum_{i=1}^{d} w_i^t \beta^{M(i,j)}}$
18. end for
```

game matrix M is given by:

$$M(i,j) = \mu^{\pi_j}(i) - \mu^{\pi_E}(i), \quad i = 1, 2, \cdots, d$$

In this equation, $\mu^{\pi_j}(i)$ represents the ith component of the feature expectation vector under strategy π_j, and $\mu^{\pi_E}(i)$ represents the ith component of the feature expectation vector under expert strategy π_E. The dimension of the weight coefficient is denoted by d. Therefore, the expression for the optimal strategy is as follows:

$$\pi^* = \max_{\pi \in \Pi} \min_{w \in W} w^T M \tag{10}$$

The basic multiplicative weight algorithm is used to update the weight vector w of the row player. The main idea of this algorithm is as follows: The row player selects the weight vector w that minimizes the target function, i.e., $w^{min} = argmin_{w \in W} w^T M$. Then, based on this weight vector, the column player selects a strategy π that maximizes the target function, i.e., $\pi^{max} = argmax_{\pi \in \Pi} w^T M$. According to the multiplication rules in this algorithm, the weight vector w is updated using a simple equation, which is given as follows:

$$w_i^{t+1} = w_i^t \frac{\beta^{M(i,j)}}{\sum_{i=1}^d w_i^t \beta^{M(i,j)}} \tag{11}$$

In this equation, t represents the current iteration number, i represents the ith component of the weight vector, β is a parameter of the multiplicative weight algorithm, and $\beta \in [0, 1)$.

VI. EXPERIMENTAL EVALUATION

The main content of this section of the experiment includes two parts:

1) The dataset is divided into good and bad expert experience datasets, and the number of states where the optimal strategy obtained by IRL algorithm and each expert strategy take different actions, as well as the difference in cumulative rewards under different iteration numbers are analyzed to verify the learning ability of the algorithm.
2) The cumulative reward values and convergence speed of several dynamic service composition methods are compared under different numbers of candidate services and state nodes to verify the effectiveness and scalability of the service composition method proposed in the paper.

A. DATA SET AND EXPERIMENT SETUP

1) EXPERIMENTAL DATA SET AND INPUT PARAMETERS

We generated the raw dataset, named API_Data, by scraping 21,867 API service description documents and 6,245 Mashup description documents from the ProgrammableWeb website. QoS data was collected by randomly calling API services from the API_Data dataset at different time intervals, generating the QoS_Data dataset for simulation experiments. To simulate large-scale services and dynamic network environments, we randomly generated an MDP state transition graph with a varying number of state nodes, ranging from 100 to 400, and a different number of services available to each node, ranging from 1,000 to 4,000. The resulting workflow search space ranges from 1000^{100} to 4000^{400}, representing a vast solution space. We assumed that candidate services within each state node were functionally equivalent, but differ in their QoS values. Using the QoS_Data dataset, we randomly assigned QoS data from the dataset to each of the candidate services. We utilized the service selection method proposed in paper [32] to generate the top 10^5 optimal service composition solutions and consider this set of solutions as the expert experience dataset.

2) COMPARISON METHOD

The dynamic service composition proposed in the paper is compared with the methods used in paper [20] and [26]:

Q-DSC: The dynamic service composition method proposed in paper [20] models the service composition problem as an MDP-SCM and uses the Q-learning algorithm to find the optimal strategy. The relevant parameters used in the Q-learning algorithm used in the experiment are: discount factor $\gamma = 0.9$, learning rate $\alpha = 0.2$.

Multi-Q-DSC: The dynamic service composition method proposed in paper [26] models the service composition problem as an MDP-SCM and uses the multi-agent

Q-learning algorithm to find the optimal strategy. The agent is set to be 4.

ZSG-IRL-DSC: The service composition re-planning method proposed in the paper models the service composition problem as an MDP-SCM and uses the inverse reinforcement learning algorithm combining zero-sum game theory to find the optimal strategy.

B. ANALYSIS OF LEARNING EFFECT OF ZSG-IRL

The dataset used in this section is divided into two groups: a good expert experience dataset and a poor expert experience dataset, and two experiments are conducted to verify the learning ability of the ZSG-IRL algorithm. The good expert experience dataset is the same as the one used in the paper, while the poor expert experience dataset randomly replaces 10% of the service composition running records with failed service compositions and low-quality service composition running records from the expert experience dataset used in the paper. The number of state nodes is set to 100, the number of candidate services available on each service class is set to 100, and each experiment runs 50 times, with the average taken as the experimental result. Due to the limited space, Table 2 only includes a portion of the expert experience data. Specifically, it provides three examples of good expert experience data and three examples of bad expert experience data. Each trajectory consists of 100 state nodes, and the table only lists the four state values from s_0 to s_3. In this table, s_i represents the state value when the service composition is in state i, and a_i represents the selected candidate service number from service class i.

In each experiment, the number of states in which the expert strategy and the learned strategy by ZSG-IRL take different actions is recorded at different iteration numbers to verify the similarity between the learned strategy and the expert strategy, i.e., the learner's learning ability, and the difference between the reward function values of the expert strategy and the learned strategy is recorded to verify the similarity between the learned strategy and the expert strategy. The experimental results are shown in Figures 4 and 5, where the horizontal axis represents the iteration number in units of 10^3.

TABLE 2. The examples of both good and bad expert experience data.

	good			bad		
	τ_1	τ_2	τ_3	τ_1	τ_2	τ_3
s_0	0	0	0	0	0	0
a_1	27	81	10	68	88	2
s_1	1	1	1	1	1	1
a_2	45	62	93	40	53	77
s_2	1	1	1	2.2	1.8	2.2
a_3	75	12	3	91	16	51
s_3	1	1	1	2.8	1.8	3

Figure 4 illustrates the number of states where the learned optimal strategy differs from the expert strategy in the good expert experience dataset and the poor expert experience dataset. In the good expert experience dataset, the curve shows a decreasing trend as the number of iterations increases and reaches a stable state after approximately 5×10^3 iterations. In the poor expert experience dataset, the curve also shows a decreasing trend as the number of iterations increases but exhibits a sharp decline within approximately 3500 iterations and then shows fluctuating trends before reaching stability after 15×10^3 iterations. In both datasets, the final results do not converge to 0, indicating that the ZSG-IRL algorithm cannot learn a strategy that is identical to the expert strategy. Additionally, more learning time is required to achieve good performance in the poor expert experience dataset.

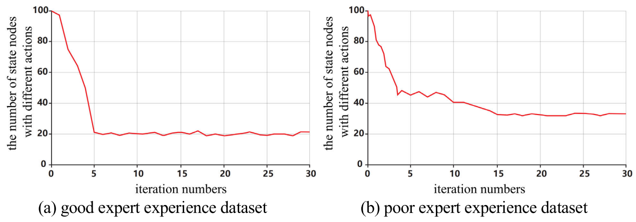

FIGURE 4. The number of states with different actions.

Figure 5 depicts the difference between the cumulative reward values of the expert strategy and the learned strategy in the good expert experience dataset and the poor expert experience dataset. In both datasets, the curve shows a rapid decreasing trend and reaches stability at approximately -2.8 and -3.1, respectively. This indicates that the cumulative reward value of the learned strategy converged to a value greater than the cumulative reward value of the expert strategy, providing evidence of the accuracy of the integrated zero-sum game inverse reinforcement learning algorithm. The algorithm can learn an optimal strategy that outperforms the expert strategy. The trend of the performance curve in both datasets is similar, suggesting that the performance of the integrated zero-sum game inverse reinforcement learning algorithm is less affected by the quality of the expert experience data. This validates the effectiveness of the proposed integrated zero-sum game inverse reinforcement learning algorithm.

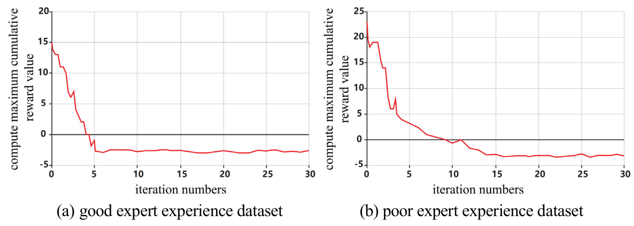

FIGURE 5. The difference of cumulative reward values.

C. COMPARATIVE ANALYSIS OF DYNAMIC SERVICE COMPOSITION METHODS

This section of the experiment is divided into two groups. The first group compares the cumulative reward values and convergence time of Q-DSC, Multi-Q-DSC, and ZSG-IRL-DSC with different candidate service numbers, while the second group compares the convergence time of Q-DSC, Multi-Q-DSC, and ZSG-IRL-DSC with different state node numbers. The purpose is to verify the effectiveness and scalability of the proposed ZSG-IRL-SCR algorithm.

1) CONVERGENCE REWARD VALUE AND CONVERGENCE TIME WITH DIFFERENT CANDIDATE SERVICE NUMBERS

In this experiment, the state node number was set to 100, and the number of candidate services that can be selected for each service category was set to 1000, 2000, 3000, and 4000, respectively. The experimental results of different candidate service numbers are shown in Figure 6. The y-axis represents the cumulative reward value. When the algorithm finally converges, the cumulative reward value will converge to a fixed value, referred to as the convergence reward value. A higher convergence reward value indicates a higher service combination quality closer to the optimal value. The x-axis represents the number of time segments. The earlier the cumulative reward value converges, the faster the algorithm converges, which means that the steeper the curve, the faster the convergence speed, and the flatter the curve, the slower the convergence speed.

The experiment result of the candidate service number being 1000 is shown in Figure 6(a). The cumulative reward value of ZSG-IRL-DSC converges at approximately 61, that of Multi-Q-DSC converges at approximately 68, and that of Q-DSC converges at approximately 63. In terms of convergence time, ZSG-IRL-DSC converges in approximately 4100 time segments, Multi-Q-DSC converges in approximately 4500 time segments, and Q-DSC converges in approximately 5400 time segments. The experiment results show that the convergence reward value of ZSG-IRL-DSC is the lowest, but the convergence speed is the fastest. ZSG-IRL-DSC approaches the convergence reward value of Q-DSC, and the convergence time of ZSG-IRL-DSC is reduced by 8.9% compared to Multi-Q-DSC with the maximum convergence reward value.

The experimental results obtained in Figure 6(a) can be attributed to several factors. Firstly, inverse reinforcement learning is particularly well-suited for the dynamic web service composition problem due to its ability to learn from expert experience data. This allows the system to leverage the knowledge and decision-making processes of human experts when generating the reward function and optimal strategy. The use of IRL enables the system to capture complex patterns and preferences present in the expert experience dataset, leading to effective and efficient service composition. However, it is acknowledged that traditional IRL methods may suffer from limitations in terms of accuracy. The inherent uncertainty and variability of web service quality of service (QoS) data can pose challenges when calculating the reward function. To overcome this limitation, an improved IRL approach ZSG-IRL-DSC was employed in the experiments. This approach aimed to approximate

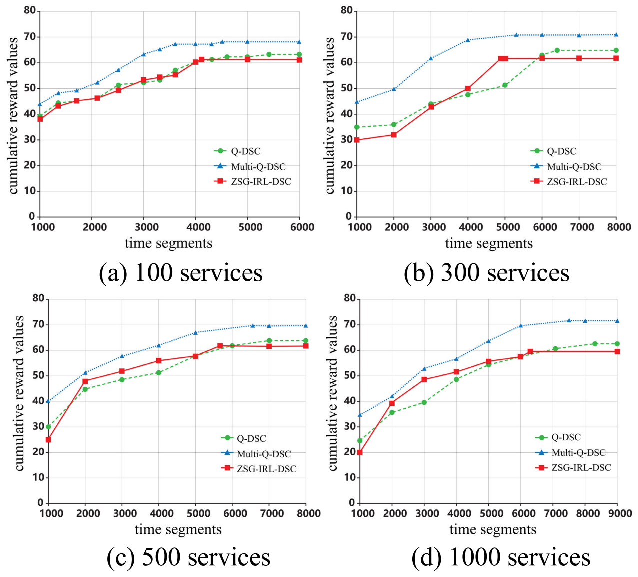

FIGURE 6. Scalability comparison for different numbers of candidate services.

the results of Q-learning, a well-established reinforcement learning algorithm known for its accuracy. By enhancing the IRL methodology, the system was able to achieve results that were close in performance to Q-learning while maintaining faster convergence.

The experimental results shown in Figures 6 (b), (c), and (d) demonstrate that as the number of candidate services increases, it has an impact on the performance of the algorithms. The Q-DSC algorithm converges at approximately 6400 time steps with a cumulative convergent reward of around 65, when the number of candidate services is 2000. The Multi-Q-DSC algorithm converges at approximately 5300 time steps with a cumulative convergent reward of around 71 under the same condition. The ZSG-IRL-SCR algorithm converges at approximately 4900 time steps with a cumulative convergent reward of around 62. When the number of candidate services is 3000, Q-DSC converges at approximately 7100 time steps with a cumulative convergent reward of about 64, Multi-Q-DSC converges at approximately 6600 time steps with a cumulative convergent reward of about 70, and ZSG-IRL-DSC converges at approximately 5700 time steps with a cumulative convergent reward of about 62. When the number of candidate services is 4000, Q-DSC converges at approximately 8300 time steps with a cumulative convergent reward of about 62, Multi-Q-DSC converges at approximately 7500 time steps with a cumulative convergent reward of about 71, and ZSG-IRL-DSC converges at approximately 6300 time steps with a cumulative convergent reward of about 59. These results indicate that the size of the candidate service set affects the performance of the algorithms. With an increase in the number of candidate services, the convergence speed of the algorithms decreases, while the cumulative convergent reward does not show a clear upward trend.

2) THE CONVERGENCE TIME FOR DIFFERENT NUMBERS OF STATE NODES

In this experiment, the number of candidate services for each service class was set to 1000, and the number of state nodes was set to 100, 200, 300, and 400, respectively. The running times of the algorithms to obtain the optimal strategy are shown in Figure 7. As the number of state nodes increases, the running time of all three methods increases to varying degrees. ZSG-IRL-DSC performs the best, followed by Multi-Q-DSC. When the number of state nodes is set to 400, the running time of Q-DSC is approximately 27.8 minutes, that of Multi-Q-DSC is about 22.4 minutes, and ZSG-IRL-DSC only needs 16.1 minutes to converge.

The experimental results demonstrate that the convergence cumulative return of ZSG-IRL-DSC is close to that of Q-DSC and it performs better than Multi-Q-DSC and Q-DSC in terms of convergence speed. This means that ZSG-IRL-DSC can effectively reduce the running time while obtaining a cumulative return value that is superior to the expert's strategy.

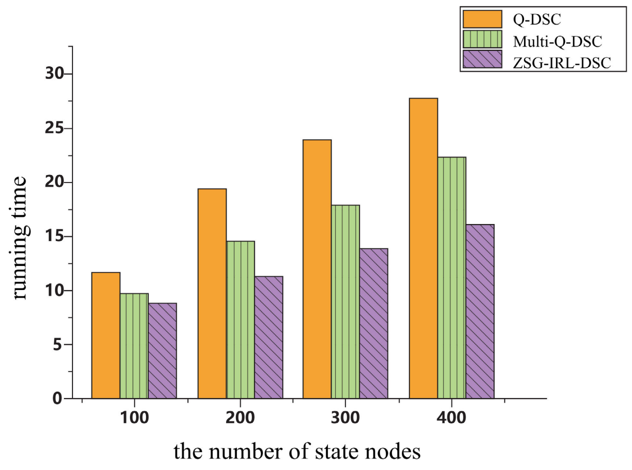

FIGURE 7. Convergence time for different numbers of states.

VII. CONCLUSION

In this paper, a dynamic service composition method based on ZSG-IRL is proposed. To achieve this, we introduce a service composition framework based on inverse reinforcement learning, along with a Markov Decision Process (MDP) service composition model, and present methods for computing state values and optimal strategies within this model. Additionally, we develop an inverse reinforcement learning algorithm based on a combination of the zero-sum game and inverse reinforcement learning concepts to find the optimal strategy for MDP-SCM. Experimental results demonstrate that our proposed dynamic service composition method not only achieves better-than-expert optimal strategies but also reduces the time required for service composition re-planning. Furthermore, this method effectively addresses the problem of increasing numbers of candidate services and state nodes.

In our future work, we plan to use large-scale service composition execution records from the Internet, along with client context information, to predict the service composition environment's status and Web service QoS information, and to develop high-quality service composition solutions that can improve the quality of service composition and success rate of service composition execution.

REFERENCES

[1] A. Moustafa, "On learning adaptive service compositions," *J. Syst. Sci. Syst. Eng.*, vol. 30, no. 4, pp. 465–481, Aug. 2021.

[2] Y.-Y. Fanjiang, Y. Syu, and W.-L. Huang, "Time series QoS forecasting for web services using multi-predictor-based genetic programming," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1423–1435, Jun. 2022.

[3] P. Rajeswari and K. Jayashree, "Hybrid metaheuristics web service composition model for QoS aware services," *Comput. Syst. Sci. Eng.*, vol. 41, no. 2, pp. 511–524, 2022.

[4] A. Palade and S. Clarke, "Collaborative agent communities for resilient service composition in mobile environments," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 876–890, Apr. 2022.

[5] H. Wang, J. Li, Q. Yu, T. Hong, J. Yan, and W. Zhao, "Integrating recurrent neural networks and reinforcement learning for dynamic service composition," *Future Gener. Comput. Syst.*, vol. 107, pp. 551–563, Jun. 2020.

[6] Y. Liu, H. Liang, Y. Xiao, H. Zhang, J. Zhang, L. Zhang, and L. Wang, "Logistics-involved service composition in a dynamic cloud manufacturing environment: A DDPG-based approach," *Robot. Comput.-Integr. Manuf.*, vol. 76, Aug. 2022, Art. no. 102323.

[7] R. Z. Yasmina, H. Fethallah, and L. Fadoua, "Web service selection and composition based on uncertain quality of service," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 1, pp. 1–20, Jan. 2022.

[8] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE Trans. Services Comput.*, vol. 7, no. 1, pp. 32–39, Mar. 2014.

[9] Y. K. Wang, S. L. Wang, S. Gao, and X. X. Guo, "Adaptive multi-objective service composition reconfiguration approach considering dynamic practical constraints in cloud manufacturing," *Knowl.-Based Syst.*, vol. 234, Dec. 2021, Art. no. 107607.

[10] G. H. Alferez and V. Pelechano, "Facing uncertainty in web service compositions," in *Proc. IEEE 20th Int. Conf. Web Services*, Santa Clara, CA, USA, Jun./Jul. 2013, pp. 219–226.

[11] M. Yang and X. Hu, "SVM-based efficient QoS-aware runtime adaptation for service oriented systems," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, San Francisco, CA, USA, Jul. 2016, pp. 396–403.

[12] A. Zisman, G. Spanoudakis, J. Dooley, and I. Siveroni, "Proactive and reactive runtime service discovery: A framework and its evaluation," *IEEE Trans. Softw. Eng.*, vol. 39, no. 7, pp. 954–974, Jul. 2013.

[13] I. Guidara, I. Al Jaouhari, and N. Guermouche, "Dynamic selection for service composition based on temporal and QoS constraints," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, San Francisco, CA, USA, Jun. 2016, pp. 274–276.

[14] A. Yachir, Y. Amirat, A. Chibani, and N. Badache, "Event-aware framework for dynamic services discovery and selection in the context of ambient intelligence and Internet of Things," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 85–102, Jan. 2016.

[15] C.-A. Sun, Z. Wang, Z. Zhang, P. Wang, X. He, and J. Han, "Toward supporting unplanned dynamic changes of service-based business processes," *IEEE Access*, vol. 7, pp. 48982–48997, 2019.

[16] D. Zhao, Z. Zhou, P. C. K. Hung, S. Deng, X. Xue, and W. Gaaloul, "CTL-based adaptive service composition in edge networks," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1051–1065, Mar. 2023.

[17] X. Yu, C. Ye, B. Li, H. Zhou, and M. Huang, "A deep Q-learning network for dynamic constraint-satisfied service composition," *Int. J. Web Services Res.*, vol. 17, no. 4, pp. 55–75, Oct. 2020.

[18] P. Alizadeh, A. Osmani, M. E. Khanouche, A. Chibani, and Y. Amirat, "Reinforcement learning for interactive QoS-aware services composition," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1098–1108, Mar. 2021.

[19] L. Quan, Z.-L. Wang, and X. Liu, "A real-time subtask-assistance strategy for adaptive services composition," *IEICE Trans. Inf. Syst.*, vol. E101.D, no. 5, pp. 1361–1369, 2018.

[20] L. F. Ren, W. J. Wang, and H. Xu, "A reinforcement learning method for constraint-satisfied services composition," *IEEE Trans. Services Comput.*, vol. 7, no. 1, pp. 32–39, Oct. 2020.

[21] H. Wang, G. Huang, and Q. Yu, "Automatic hierarchical reinforcement learning for efficient large-scale service composition," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, San Francisco, CA, USA, Jul. 2016, pp. 57–64.

[22] R. Yang, B. Li, and Z. Liu, "Automatic hierarchical reinforcement learning for reusing service process fragments," *IEEE Access*, vol. 9, pp. 20746–20759, 2021.

[23] C.-S. Ying, A. H. F. Chow, H. T. M. Nguyen, and K.-S. Chin, "Multi-agent deep reinforcement learning for adaptive coordinated metro service operations with flexible train composition," *Transp. Res. B, Methodol.*, vol. 161, pp. 36–59, Jul. 2022.

[24] H. Wang, X. Wang, X. Zhang, Q. Yu, and X. Hu, "Effective service composition using multi-agent reinforcement learning," *Knowl.-Based Syst.*, vol. 92, pp. 151–168, Jan. 2016.

[25] Y. Lei and P. S. Yu, "Multi-agent reinforcement learning for service composition," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, San Francisco, CA, USA, Jul. 2016, pp. 790–793.

[26] H. Wang, X. Chen, Q. Wu, Q. Yu, X. Hu, Z. Zheng, and A. Bouguettaya, "Integrating reinforcement learning with multi-agent techniques for adaptive service composition," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 2, pp. 1–8, 2017.

[27] A. Moustafa, "On learning adaptive service compositions," *J. Syst. Sci. Syst. Eng.*, vol. 30, no. 4, pp. 465–481, Aug. 2021.

[28] A. G. Neiat, A. Bouguettaya, and M. Bahutair, "A deep reinforcement learning approach for composing moving IoT services," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2538–2550, Oct. 2022.

[29] H. Wang, X. Hu, Q. Yu, M. Gu, W. Zhao, J. Yan, and T. Hong, "Integrating reinforcement learning and skyline computing for adaptive service composition," *Inf. Sci.*, vol. 519, pp. 141–160, May 2020.

[30] K. G. Vamvoudakis and F. L. Lewis, "Multi-player non-zero-sum games: Online adaptive learning solution of coupled Hamilton–Jacobi equations," *Automatica*, vol. 47, no. 8, pp. 1556–1569, Aug. 2011.

[31] D. Liu, H. Li, and D. Wang, "Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm," *Neurocomputing*, vol. 110, pp. 92–100, Jun. 2013.

[32] Y. Yuan, W. Zhang, X. Zhang, and H. Zhai, "Dynamic service selection based on adaptive global QoS constraints decomposition," *Symmetry*, vol. 11, no. 3, p. 403, Mar. 2019.

YUAN YUAN was born in Liaoning, China, in 1989. She received the B.S. degree in network engineering, the M.S. degree in computer application technology, and the Ph.D. degree in computer technology from Dalian Maritime University, China, in 2012, 2014, and 2020, respectively. From September 2020 to April 2022, she was a Research Assistant with the Institute of Higher Education, Dalian Maritime University. Since April 2022, she has been a Lecturer with the Zhejiang University of Science and Technology. She is the author of one book and more than ten research articles. Her research interests include service computing, combinatorial optimization, and machine learning.

YUHAN GUO was born in Heilongjiang, China, in 1983. He received the B.S. degree in computer science and technology from the Harbin Institute of Technology, China, in 2005, the M.S. degree in software engineering from the Harbin Institute of Technology, in production engineering from the Bordeaux of University, France, in 2009, and the Ph.D. degree in computer and automation from the University of Artois, France, in 2013. From January 2015 to July 2021, he was an Associate Professor and the Head of the Software Engineering Research Center, Liaoning Technical University, China. Since August 2021, he has been with the Zhejiang University of Science and Technology. He is the author of two books and more than 30 research articles. His research interests include intelligent transportation, combinatorial optimization, machine learning, and distributed computing. He is a member of the China Computer Federation.

WANQING MA was born in Heilongjiang, China, in 1996. She received the B.S. degree in resource exploration engineering from Jilin University, in 2019. She is currently pursuing the M.S. degree with the Zhejiang University of Science and Technology. Her research interests include big data technology and application and education quality evaluation.