Integrating On-policy Reinforcement Learning with Multi-agent Techniques for Adaptive Service Composition

Hongbing Wang[1], Xin Chen[1], Qin Wu[1], Qi Yu[2], Zibin Zheng[3], and Athman Bouguettaya[4]

[1] School of Computer Science and Engineering, Southeast University, China
hbw@seu.edu.cn, {cyceve,bellawu627}@gmail.com
[2] College of Computing and Information Sciences, Rochester Institute of Tech, USA
qi.yu@rit.edu
[3] Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Hong Kong, China
zbzheng@cse.cuhk.edu.hk
[4] School of Computer Science and Information Technology, RMIT, Australia
athman.bouguettaya@rmit.edu.au

Abstract. In service computing, online services and the Internet environment are evolving over time, which poses a challenge to service composition for adaptivity. In addition, high efficiency should be maintained when faced with massive candidate services. Consequently, this paper presents a new model for large-scale and adaptive service composition based on multi-agent reinforcement learning. The model integrates on-policy reinforcement learning and game theory, where the former is to achieve adaptability in a highly dynamic environment with good online performance, and the latter is to enable multiple agents to work for a common task (i.e., composition). In particular, we propose a multi-agent SARSA (State-Action-Reward-State-Action) algorithm which is expected to achieve better performance compared with the single-agent reinforcement learning methods in our composition framework. The features of our approach are demonstrated by an experimental evaluation.

1 Introduction

As the mainstream paradigm of SOC (Service-oriented Computing), the research on theories of service composition and related technologies for seamless integration of business applications is always the core proposition. However, large-scale service composition faces a multitude of thorny issues, such as, accuracy, interoperability, efficiency and adaptability for practical use, if there exist massive services with similar functionality in a highly-dynamic environment.

Under the premise of validity for service composition, efficiency, adaptability and optimality of composition in large-scale and dynamic scenarios are especially significant. First of all, both the complexity of business flows and the quantity of candidate services may affect the efficiency of the service orchestration. Secondly,

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 154–168, 2014.

how to adapt to the services' internal changes and external dynamic environment is a grand challenge. Furthermore, how to achieve the optimal aggregated QoS should also be taken into consideration. Therefore, a novel method should be proposed to obtain a good balance between those objectives.

Previous studies mainly focus on integer programming, graph planning, reinforcement learning (RL) and so on. Ardagna et al. [1] modelled the QoS information of candidate services by a multi-channel framework, and then utilized Mixed Integer Programming (MIP) to obtain the optimal solution. However, this method only performs well for small-scale problems, and the computing resource consumption may become prohibitive when faced with large-scale scenarios. Beauche et al. [2] used a hierarchical planning approach based on graph planning and hierarchical task networks to construct adaptive service composition. However, continuous emergence and demise of services lead to sustained search of viable services for updating the corresponding planning graph, which is not suitable for a highly dynamic environment. Jureta et al. [7] proposed a multi criteria-driven reinforcement learning algorithm to ensure that the system is responsive to the availability changes of Web services. We also [20] proposed an adaptive service composition approach based on reinforcement learning method combined with logic preference. Despite the effectiveness of conventional reinforcement learning in achieving adaptability, such methods can not ensure high efficiency in a large-scale and complex scenario.

As a subdiscipline of distributed artificial intelligence (DAI) [15], multi-agent techniques have arisen as a viable solution for modularity, more computing power, scalability and flexibility required by service composition [16]. Some researchers have already applied multi-agent techniques to service composition. Maamar et al. [11] proposed a web service composition method based on multi agents and context awareness. Gutierrez-Garcia et al. [5] characterized behavior of the services with Colored Petri-net, and exploited multi-agent techniques for services orchestration in the context of cloud computing. Unfortunately, those methods seldom take self-adaptivity into consideration.

In view of superiority from RL and multi-agent technologies, a natural idea to achieve self-adaptability in a dynamic environment and maintain acceptable efficiency when faced with massive candidate services is to combine them together, which has already been discussed in the field of DAI and is called Multi-agent reinforcement learning (MARL) [15]. On the one hand, RL is a commonly used machine learning method for planning and optimization in a dynamic environment [18], which learns by trial-and-error interaction with dynamic environment and thus has good self-adaptability. On the other hand, multi-agent technology can compensate for inefficiencies under large-scale and complex scenarios.

In this paper, we propose a new adaptive model that is built upon MARL. Different from previous work, this new model is based on team Markov Games, which is more mature and generic for service composition in a multi-agent scenario. To tackle the common problems of agent coordination and equilibrium selection emerged in a multi-agent environment, we introduce the coordination equilibrium and fictitious play process to ensure the agents to converge to a

unique equilibrium when faced with multiple equilibriums. Finally, we have proposed the multi-agent Sarsa algorithm for our multi-agent service composition. Our contributions are summarized as follows:

- We introduce a *TMG-WSC* model for service composition with massive candidate services in a highly dynamic and complex environment.
- We propose a multi-agent Sarsa algorithm to adapt to the multi-agent service composition scenarios and achieve a better performance.
- We present the concept of multi-agent service composition that caters for the distributed environment and big data era.

The reminder of this paper is organized as follows. Section 2 compares our approach against some related works. Section 3 introduces the problem formulation and basic definitions. Section 4 presents our approach for service composition based on MARL. In section 5, some experimental results are presented for evaluating the proposed approach. Finally, the paper is concluded in Section 6.

2 Related Work

In this section, we review some existing works that are most relevant to our approach, including RL and agent techniques adopted in service composition.

Moustafa et al. [13] proposed a approach to facilitate the QoS-aware service composition problem using multi-objective reinforcement learning, but the method is not very efficient for large-scale service composition scenarios. Our prior work [20] suffer from the same issue with preceding method.

Xu et al. [22] proposed a multi-agent learning model for service composition, based on the Markov game and Q-learning with a hierarchical goal structure to accelerate the searching of states during the learning process. However, their model may not work well when faced with a complicated goal with more mutual dependencies between each sub-goals as their agents are fixed for certain service classes. We proposed a multi-agent learning model [19] based on MDP and knowledge sharing before, however this can not be regarded as a real multi-agent framework as the MDP is designed for a single agent and does not take the potential collaboration between agents into consideration.

MARL has strong connections with game theory [4], because the relation between agents has a great impact on the design of learning dynamics. According to Claus and Boutilier [4], the MARL can be classified into two forms. The first one is independent learners (ILs), which just apply RL methods (Q-learning, Sarsa etc.) and ignore the existence of other agents. The second one is joint action learners (JALs), which learn their actions in conjunction with others via integration of RL with a certain kind of Nash equilibrium, just like the coordination equilibrium [3,4]. Consequently, agents coordination and equilibrium selection are the key issue in MARL for JALs. Wang et al. [21] proposed an algorithm which can ensure to converge to an optimal equilibrium, but its high computational cost has limited its practical use.

In this paper, we integrate on-policy reinforcement learning with multi-agent techniques for services composition. The proposed approach is fundamentally

different from existing approaches (e.g., [19,22]) as we exploit the coordination equilibrium and fictitious play process to ensure the agents to converge to a unique equilibrium. We also propose a multi-agent Sarsa algorithm to achieve an optimal or suboptimal solution.

3 Problem Formulation

Defnition 1 (Web Service). *A Web service is modeled as a pair WS=< ID, QoS >, where ID is the identifier of the Web service. QoS is a n-tuple $< att_1, ..., att_n >$, where each $att_i (1 \leq i \leq n)$ denotes a QoS attribute value.*

As we use Team Markov Games (TMG) to model multi-agent service composition, we first introduce the basis of TMG-based service composition, that is MDP (Markov Decision Process)-based service composition.

Defnition 2 (MDP-based web service composition (MDP-WSC)). *A MDP-WSC is a 6-tuple MDP-WSC=< $S, S_0, S_\tau, A(.), P, R$ >, where S is a finite set of the world states; $S_0 \in S$ is the initial state from which an execution of the service composition starts; $S_\tau \subset S$ is the set of terminal states, indicating an end of composition execution when reaching one state $S_\tau^i \in S_\tau$; $A(s)$ represents the set of services that can be executed in state $s \in S$; P is the probability distribution. When a web service α is invoked, the world makes a transition from its current state s to a succeeding state s'. The probability for this transition is labeled as $P(s'|s, \alpha)$. R is the immediate reward from the environment after executing an action.*

Fig.1 shows a MDP-WSC graph of a composite service for a vacation plan. It consists of two kinds of nodes, i.e., state nodes and service nodes, which are represented by open circles and solid circles, respectively. s_0 is the initial state node, and nodes with double circles are terminal state nodes, such as s_{10}. A state node can be followed by a number of invoked service nodes, labeled with the transition probability $P(s'|s, \alpha)$. A MDP-WSC transition graph can be created by using some automatic composition approaches, such as an AI planner [14].

With multiple agents in the environment, the fundamental problem of MDP is that the approach treats the other agents as a part of the environment and thus ignores the fact that the decisions of the other agents may influence the environment state. Then how can we extend the single-agent MDP model and adjust it for the multi-agent scenarios? One possible solution is to use the multi-agent Markov decision processes, i.e. Markov games [8].

Defnition 3 (Markov Games). *An n-player involved Markov games is modeled as a 5-tuple MG=< a, S, A, T, R >, where a is the set of agents; S is a finite environment states set; A $(A_1 \times A_2 \times ... \times A_n)$ is the joint action, $A_i (i = 1, ..., n)$ is a discrete available action set of the ith agent; T: $S \times A \rightarrow \prod(S)$ is the transition function, giving for each state and one action from each agent. A probability distribution $T(s, a_{1,...,}a_n, s')$ is the probability of state transition*

from joint state s to s', and each agent $i(1 \leq i \leq n)$ choose action $a_i \in A_i$; $R_i : S \times A \to \Re$ is the ith agent's reward function, giving the immediate reward gained by the ith agent for each set of available actions.

Markov Games is so called team Markov Games when all agents strive for a common goal and thus share a common reward function. Here we adopt team Markov Games as all agents work for a common service workflow.

However, Markov Games can not directly replace the MDP model for multi-agent service composition, because some differences arises when trying to transfer some concepts in the MDP-WSC model [20] to the new multi-agent environment.

For example, in MDP-WSC, there is only one learning agent, which always starts from the initial state. If it finally reaches the terminal state, it can get a full path from the initial state to the terminal state according to its trajectory.

Unfortunately, it is much more complicated in the multi-agent scenario, as there are a group of learning agents and each one starts from one of the states randomly instead of a fixed initial state in MDP-WSC model. So even someone has reached one of the terminal states, we can not claim that they have completed current learning episode and got the full path, because this "lucky" one may not start from the initial state, and consequently what it has marched is just part of the whole path. In order to handle this problem, we need to introduce some new concepts to fit in the new multi-agent scenario.

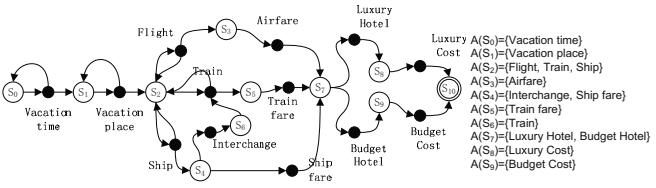


Fig. 1. The MDP-WSC of a Composite Service

Defnition 4 (Candidate Initial State). *The joint state $\overline{S_0} = s_1 \times ... \times s_n(s_i, i = 1, ..., n,$ is the state of the ith agent in the team) is a candidate initial state iff $s_i = s_0(1 \leq i \leq n)$, where s_0 is the initial state of the MDP-WSC transition graph.*

Suppose that a 3-agent group is wandering in Fig.1. If agent 1 starts from s_0, agent 2 starts from s_2, and agent 3 starts from s_5, the joint state $\overline{s} = s_0 \times s_2 \times s_5$ is a so-called Candidate Initial State, because it contains the initial state node s_0, which is the initial state in MDP-WSC transition graph. In contrast, a joint state $\overline{s} = s_1 \times s_2 \times s_5$ for Fig.1 can not be regarded as a candidate initial state, because it does not contain any initial state node. Since Candidate Initial State represents the starting points in the Multi-agent scenario, the question is what is the ending state. Hence we introduce the concept of Possible Terminal State.

Defnition 5 (Possible Terminal State). *The joint state* $\overline{S_x} = s_1 \times ... \times s_n(s_i, i = 1, ..., n$, *is the state of the ith agent) is a possible terminal state iff* $s_i = s_\tau (1 \leq i \leq n)$, *where* s_τ *is among terminal states of the MDP-WSC transition graph.*

More specifically, considering Fig. 1 in a 3-agent setting, where agent 1 starts from s_0, agent 2 starts from s_2, and agent 3 starts from s_9. After some steps, agent 3 may reach the terminal state node s_{10}, while agent 1 reaches s_1 and agent 2 reaches s_4. The joint state $\overline{s} = s_1 \times s_4 \times s_{10}$ at this time is obviously a possible terminal state. But it is not a true terminal state for the multi-agent environment, because the three sub-path $s_0 \rightarrow s_1$, $s_2 \rightarrow s_4$ and $s_9 \rightarrow s_{10}$ can not form a full path from the initial state s_0 to the terminal state s_{10}.

Defnition 6 (Passed State Set). *The set* S_p *is a passed state set iff* S_p *contains all the states that agents in the team have passed by.*

We can display a back trace from the terminal state and check whether it can stretch back to the initial state using the Passed State Set. Next, we will propose our multi-agent model called TMG-WSC for service composition, which is based on Team Markov Games (TMG) and new concepts mentioned before.

Defnition 7 (TMG-based Web Service Composition(TMG-WSC)). *A TMG-WSC is a 7-tuple=*$< a, S, S_0, S_x, A, T, R >$, *where a is the set of agents; S is the discrete set of environment states;* S_0 *is the set containing all the candidate initial state* $\overline{S_0}$, *and an execution of the composition starts from one state* $\overline{S_0}^i \in S_0$; S_x *is the set containing all the possible terminal state* $\overline{S_x}$, *and an execution of the composition has a possibility to terminate upon reaching any state in* S_x; $A(\overline{s}) = A_1(s_1) \times A_2(s_2) \times ... \times A_n(s_n)$ *is the finite set of joint actions that can be executed in joint state* $\overline{s} \in S$, *where* $A_i(s_i)(i = 1, ..., n)$ *is the discrete set of actions available to the ith agent at state* S_i; $T{:}S \times A \times S \rightarrow [0, 1]$ *is the transition probability function labeled as* $P(\overline{s'} \mid \overline{s}, A(\overline{s}))$, *giving for each joint state and each joint action;* $R{:}S \times A \rightarrow \Re$ *is the common reward function for all the agents in the team. When the set of services corresponding to the joint action are invoked and the environment has changed into the resulting state* $\overline{s'}$, *the team will receive an immediate reward* $R(\overline{s'} \mid \overline{s}, A(\overline{s}))$ *according to the feedback of this execution.*

A TMG-WSC can be visualized as a multi-dimensional transition network based on the MDP-WSC transition graph. Fig.2 shows a part of the TMG-WSC transition graph for vacation plan in a 3-agent scenario, which is constructed based on the MDP-WSC graph in Fig.1.

The solution to a TMG-WSC is a deterministic decision policy, which is defined as a procedure for service selection $ws \in A$ by all agents in every state s. These policies, represented by π, are actually mappings from states to actions, defined as $\pi : S \rightarrow A$.

Each deterministic policy can uniquely determine a workflow, and therefore the task of our service composition model is to identify the optimal policy or workflow that offers the best cumulative reward depending on QoS attributes.

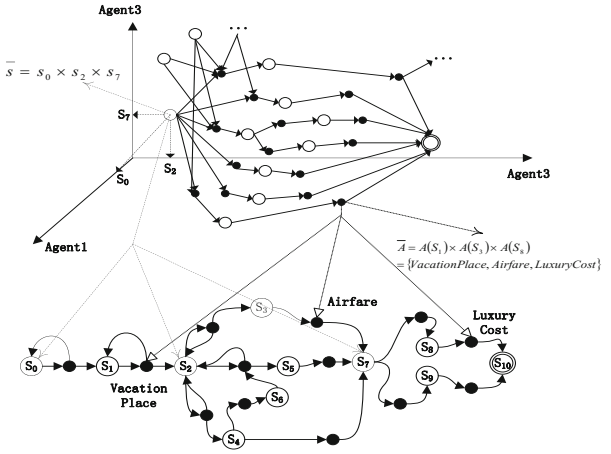


Fig. 2. A Part of the TMG-WSC of a Composite Service

4 Multi-agent On-policy Learning for Composition

The introduced TMG-WSC model allows engineers to integrate multiple alternative services into a single service composition. If the complete information of TMG-WSC is known, the theoretical optimal policy can always be calculated. However, this hypothesis is not true in practice. We may not have complete knowledge about the state transition functions and reward functions.

Moreover, both the state transition functions and the reward functions will change along the time, and the computational cost in a large-scale scenario will inevitably increase to an intolerable point. To solve the above issues, we propose an approach based on Multi-agent techniques and Sarsa algorithm in RL to learn the optimal policy of a TMG-WSC at runtime.

4.1 SARSA

Compared with off-policy learning methods like Q-learning, on-policy learning methods has an advantage in on-line performance, since the estimation policy, that is iteratively improved, is also the policy used to control its behavior [17].

Sarsa is a classic on-policy reinforcement learning method. The task of the learner in Sarsa is to learn a policy that maximizes the expected sum of reward. The cumulative reward starting from an arbitrary state s_t and following a policy π is defined as Eq.1, where r_{t+i} is the expected reward in each step, and γ is a discount factor.

$$V^{\pi}(s_t) = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + ... = \sum_{i=0}^{\infty} \gamma^i * r_{t+i} \tag{1}$$

Based on Eq.1, we can deduce the reward of action pair $< s_t, a_t >$, that is, the feedback of executing action a_t at state s_t, which is defined as Eq.2, where s_{t+1}

is the resulting state by executing a_t and $P(s_{t+1}|s_t, a_t)$ is the probability distribution, $r(s_t, a_t)$ represents the immediate reward of taking action a_t at state s_t, which is defined as Eq.3.

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma * \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) * V^{\pi}(s_{t+1}) \tag{2}$$

$$r(s_t, a_t) = \sum_{i=1}^{m} w_i * \frac{Att_i^{a_t} - Att_i^{min}}{Att_i^{max} - Att_i^{min}} \tag{3}$$

In Eq.3, $Att_i^{a_t}$ represents the observed value of the ith attribute of the service corresponding to the executed action a_t, and Att_i^{max}, Att_i^{min} represent the maximum and minimum values of Att_i for all services. w_i is the weighting factor. w_i is positive if users prefer Att_i to be high (e.g. reliability). w_i is negative if preferring Att_i to be low (e.g. service fee). m is the number of QoS attributes.

The Q function represents the best possible cumulative reward of executing a_t at s_t. We can run dynamic programming (value iteration) by performing the Bellman back-ups in terms of the Q function as follows:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma * \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) * Q(s_{t+1}, a_{t+1}) \tag{4}$$

Further on, we rewrite this recursive formula in a stochastic version:

$$Q(s_t, a_t) \leftarrow (1 - \alpha) * Q(s_t, a_t) + \alpha * [r(s_t, a_t) + \gamma * Q(s_{t+1}, a_{t+1})] \tag{5}$$

$\alpha(0<\alpha<1)$ is the learning ratio, which is an important tuning factor in Sarsa. The stochastic version does not require a priori knowledge of the transition probability distribution P or the reward function R. Eq.5 forms the basis of the Sarsa algorithm, which starts with initial values of Q(s_t, a_t), and updates Q(s_t, a_t) recursively using the actual reward received.

More specifically, Q(s_t, a_t) is initialized to 0 for all s_t and a_t at the beginning. Then, the learning process is performed recursively. The learner starts from the initial state s_0, and takes a sequence of actions following a Boltzmann policy (which is introduced subsequently) in each learning episode. $Q(s_t, a_t)$ is updated by the real feedback of next state-action pair$< s_{t+1}, a_{t+1} >$ rather than the maximum estimation value in Q-learning, which means that it is depending on the engine's on-line execution and performance. However, Eq.5 is just the single-agent version. To incorporating Sarsa with multi-agent techniques, we need to extend Eq.5 for multi-agent scenario.

We first define the reward function in multi-agent framework as follows, which aggregates the reward values of the services invoked by every agent. n is the number of agents, $\overline{s}_t$ is the current joint state, $\overline{a}_t$ is the joint action executed.

$$R(\overline{s}_t, \overline{a}_t) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} * \frac{Att_{ij}^{\overline{a}_t} - Att_{ij}^{min}}{Att_{ij}^{max} - Att_{ij}^{min}} \tag{6}$$

Based on Eq.6, we can plug this and rewrite Eq.5 in a multi-agent form:

$$Q_{i_1,i_2\ldots i_n}(\overline{s}_t,\overline{a}_t) \leftarrow (1-\alpha) * Q_{i_1,i_2\ldots i_n}(\overline{s}_t,\overline{a}_t) + \alpha * [R(\overline{s}_t,\overline{a}_t) + \gamma * Q_{i_1,i_2\ldots i_n}(\overline{s}_{t+1},\overline{a}_{t+1})] \tag{7}$$

4.2 Equilibrium Coordination

For many Markov games, there is no policy that is un-dominated because the performance depends critically on the behavior of the other agents. Then, how can we define a deterministic optimal policy in this case? An natural idea from the game-theory literature is to define an agent's optimal behavior as being its behavior at a Nash equilibrium. Some researchers, like Littman, have already done such work in the field of MARL [6,9,10]. Here we adopt Littman's idea and give the definition of multi-agent optimal policy as follows:

$$\begin{aligned}\pi^*(s,a_1,...,a_n) &= \sum_{a_1,\ldots,a_n} \pi_1(s,a_1) * \ldots * \pi_n(s,a_n) * Q(s,a_1,...,a_n) \\ &= \max_{a_1,\ldots,a_n} Q(s,a_1,...,a_n)\end{aligned} \tag{8}$$

This definition is based on coordination equilibrium, which means all agents have precisely the same goal and achieve their maximum possible payoff in coordination team. It is obvious that the equilibrium of a team Markov game is a Coordination equilibrium as all the players involved strive for a common task.

In view of this, seeking the optimal policy in Team Markov games can be turned into an old question of optimizing the Q-value. To sum up, we can compute an optimal policy by just applying Sarsa in the multi-agent scenario.

However, another important open problem for Markov games is finding a general way of selecting an equilibrium when there exists multiple coordination equilibriums, which is very common in multi-agent scenario [21]. Here we use the indirect coordination methods to solve the problem, which places bias on action selection toward actions that are likely to result in good rewards.

An easy and well-understand indirect coordination method for equilibrium selection in game theory is fictitious play [12]. The key idea of fictitious play is estimating the empirical distribution of others' past actions and thus to help the agents figure out a best response until now. More specially, suppose that $C^j_{a_j}$ is the frequency of agent j invoking action a_j in the past, where a is the set of agents, each $j \in a$ and $a_j \in A_j$ (A_j represents the set of actions available to the jth agent). Then, agent i assumes agent j to play action a_j with the probability as Eq. 9. After each round of playing, agent i will update its $C^j_{a_j}$ according to the actions taken by the others in the last round. In a sense, $C^j_{a_j}$ reflects the beliefs an agent has given the historical choices of others.

$$\Pr^i_{a_j} = \frac{C^j_{a_j}}{\sum\limits_{b_j \in A_j} C^j_{b_j}} \tag{9}$$

Dov Monderer gives the definition of Fictitious Play Property and also proves the following theorem in his work [12].

Defnition 8 (Fictitious Play Property). *A game has the fictitious play property (FPP) if every fictitious play process converges in beliefs to equilibrium.*

Theorem 1: Every game with identical payoff functions has the fictitious play property. □

In view of Theorem 1, we can deduce that the team game where the agents have common interests has the fictitious play property. Hence, fictitious play process can be applied in the Team Markov Games and help to converge to a unique equilibrium surely despite the existence of multiple equilibriums.

To improve the efficiency of the fictitious play process, Young [23] proposed an optimized version and proved its validity. Based on it, we construct a new function that combines the Q-value and fictitious play process together for estimating cumulative reward of joint action in TMG-WSC. It is defined as follows, $\frac{K_t^m(A_j)}{k}$ is a probability model for agent i at the joint state $\overline{s}$, based on the fictitious play process. t is the number of times for attending state $\overline{s}$. a_i is the action chosen by the ith agent. m is the length of the queue which stores the reduced joint action $\overline{a_{-i}}$ of agent $i's$ opponents in chronological order. $\Psi(\overline{s}, \overline{a_{-i}})$ is the best response for agent $i's$ opponents' joint action at state $\overline{s}$.

$$WEQ(\overline{s}, a_i) = \sum_{\substack{A_j \in \Psi(\overline{s}, \overline{a_{-i}}) \\ 1 \le j \le n, j \ne i}} \frac{K_t^m(A_j)}{k} Q_{i,j}(\overline{s}) \tag{10}$$

Finally, we need a learning policy for the learner to execute the TMG-WSC during the learning. Here we choose the Boltzmann learning policy as it can better characterize our coordination mechanism and equilibrium selection technique. The Boltzmann exploration used here can be depicted as follows, T is temperature parameter, $T = T_0 * (0.999)^c$, T_0 is an initial value, c is the frequency that the learner is in state $\overline{s}_t$.

$$\Pr(\overline{a}_t | \overline{s}_t) = \frac{e^{WEQ(\overline{s}_t, \overline{a}_t)/T}}{\sum_{b \in A} e^{WEQ(\overline{s}_t, \overline{a}_t)/T}} \tag{11}$$

The complete learning process is depicted in Algorithm 1.

5 Simulation Results and Analysis

In this section, we conduct a simulation study to evaluate the properties of our service composition mechanism. We demonstrate the convergence and efficiency of multi-agent Sarsa algorithm. We also compare it with single-agent Sarsa and Q-learning, and analyze the corresponding inherent cause and effect.

Initialization:
$Q_{i1,i2,...,in}(\overline{s}_t, \overline{a}_t)$
repeat
// for each episode
each agent choose an action $a_i (i = 1, 2, ..., n)$ based on Eq.11,
and form the joint action $\overline{a}_t = a_1 \times a_2 \times \cdots \times a_n$;
repeat
// for each step of a episode
1. On-Policy Learning
take joint action $\overline{a}_t$, observe R, $\overline{s}_{t+1}$, each agent choose action a based on Eq.11, and form the joint action $\overline{a}_{t+1} = a_1 \times a_2 \times \cdots \times a_n$
$Q(\overline{s}_t, \overline{a}_t) \longleftarrow (1 - \alpha) * Q(\overline{s}_t, \overline{a}_t) + \alpha * [R + \gamma * Q(\overline{s}_{t+1}, \overline{a}_{t+1})]$
$\overline{s}_t \longleftarrow \overline{s}_{t+1}$, $\overline{a}_t \longleftarrow \overline{a}_{t+1}$;
2.Terminal condition check
if $\overline{s}_t$ is a possible terminal state, $\overline{s}_t = s_1 \times s_2 \times ... \times s_n$ **then**
Create a set named $Temp$, $Temp = \{\overline{s}_t\}$,
Create a set named $Prev$, $Prev$ contains all the previously passed states of any element in $Temp$
end if
while $S_p \cap Temp \neq \Phi$ and $s_0 \notin Prev$ **do**
$Temp \leftarrow S_p \cap Temp$
$Prev \leftarrow$ all the previous states of any element in $Temp$
end while
if $Prev$ contains S_0 **then**
This episode is ended
end if
until The current episode is ended
until the cumulative reward matrix converges

Algorithm 1. Multi-agent Sarsa based on TMG-WSC

5.1 Experiment Setting

We randomly generate MDP-WSC transition graphs and use them as the input for the TMG-WSC model, and four QoS attributes are mainly considered as an example for reward assessment, which are *ResponseTime*, *Throughput*, *Availability* and *Reliability* based on the extended QWS Dataset [1]. A number of key parameters are set up for both experiments as follows. The learning rate α of single-agent algorithm is set to 0.6, the discount factor γ is set to 0.9 and the $\epsilon - greedy$ exploration strategy value is set to 0.6. The experiments are conducted on an Intel i3-2120 3.30GHz PC with 4GB RAM.

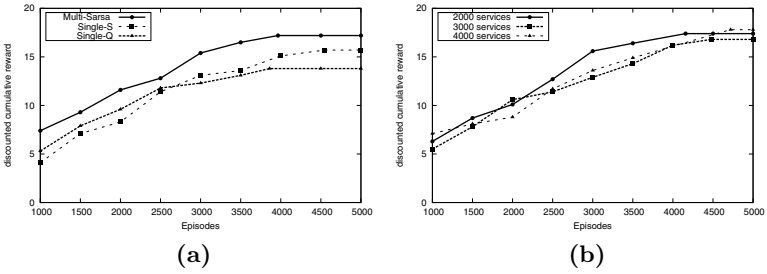
5.2 Result Analysis

1. Effectiveness and Efficiency
The purpose of the first experiment is to examine the ability of the multi-agent Sarsa algorithm with Boltzmann exploration strategy (abbr. Multi-Sarsa).

[1] `http://www.uoguelph.ca/~qmahmoud/qws/`

We compare the Multi-Sarsa with single-agent Sarsa (abbr.Single-S) and single-agent Q-learning (abbr. Single-Q) in 4-agent scenario with 100 state nodes and 1000 services for each node. As shown in Fig.3 (a), the proposed Multi-Sarsa algorithm yields higher discounted cumulative rewards and efficiency than Single-S algorithm, and is closed to the convergence rate of Single-Q. For instance, Multi-Sarsa converges to the rewards at 17.2, that is higher than Single-S at 15.7. Furthermore, Multi-Sarsa converges at about the 4000th episode, which is closed to Single-Q at about the 3900th episode. However Single-S is slower for converging at about the 4500th episodes. Single-S achieves higher discounted cumulative rewards than Single-Q but performs worse in convergence rate.

Not surprisingly, in Eq.5, the use of a_{t+1} introduces additional variance into the update rule, which may slow convergence rate when compared to Single-Q. However, differing from off-policy Q-learning method, on-policy Sarsa approach has stronger convergence guarantees when combined with function approximation and it has a potential advantage over off-policy methods in its on-line performance. In the light of those characteristics, we propose the Multi-Sarsa algorithm to offset the convergence rate of Single-S by mutual collaboration between each agents and become closer to the optimal convergence simultaneously in which multiple agents explore the learning space adequately.



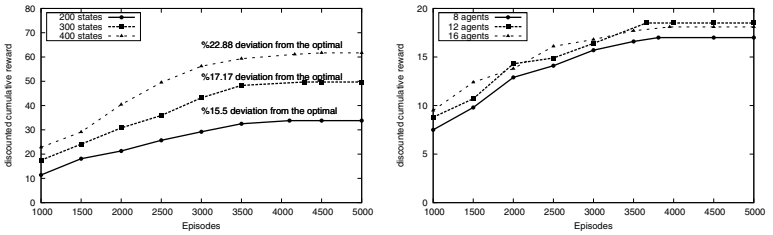
(a) **(b)**

Fig. 3. (a) Effectiveness and Efficiency Comparison (b) Different number of services

2. Scalability

The purpose of the second experiment set is to assess the scalability of the proposed Multi-Sarsa algorithm. We probe the influence of the service, state and agent number respectively.

Firstly, we vary the number of services for each state node from 2000 to 4000 while fixating the agents number for 4 and state nodes for 100. From Fig.3 (b), we know that the increasing number of candidate services for each state node may postpones the convergence. In 2000-service scenario, the Multi-Sarsa converges at about the 4200th episode, while converging at about the 4500th episode in 3000-service and about the 4700th episode in 4000-service. However, increasing the number of services does not necessarily mean the corresponding improvement of service quality, so the rewards may be higher or lower. In a word, Multi-Sarsa always converge at an acceptable time despite of vast candidate services.



(a) **(b)**

Fig. 4. (a) Different number of state nodes (b) Different number of agents

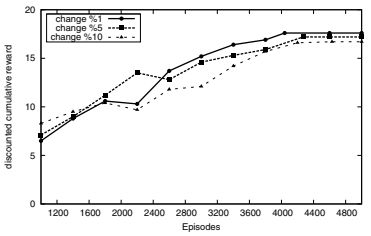


Fig. 5. Adaption Testing

Secondly, we fix the agent number and service number of each state node as 4 and 1000 respectively, and vary the state nodes from 200 to 400. As shown in Fig.4 (a), the bigger number of state nodes corresponds to higher values of the optimal convergence and a slower convergence rate. In 200-state-node scenario, the Multi-Sarsa converges at about the 4100th episode with rewards 33.8, and in 400-state-node, it converges at about the 4500th episode with rewards 61.7. What's more, we calculate the deviation of the current convergence rewards from the optimal convergence rewards in different scenarios by $D = \frac{OPR-CCR}{OPR}$, where D represents deviation degree, OPR indicates the optimal convergence rewards, and CCR is the current convergence rewards. It can be seen from Fig.4 (a), the D is %17.17 in 300 states nodes, and %22.88 in 400 scenario. That is to say, the increasing number of state nodes may aggravate the deviation from the optimality and fall into local optima. Hence, we can conclude that the Multi-Sarsa has the scalability when face with the increment of states nodes.

Finally, we come to the affect of agents number. We set the state nodes for 100, the services number for 1000 to each state node. From Fig.4 (b), we know that the more agents involved, the more adequate space exploration will be done, consequently the discount cumulative rewards is apparently bigger in scenario of 12 and 16 agents. However, the increasing number of agents brings another severe problem, that is, the communication consumption in the process of fictitious play. So, 16-agent does not perform better than 12-agent. In brief, 12-agent may be a compromise for Multi-Sarsa, the increasing number of agents does not necessarily leads to an improvement in efficiency, and the communication consumption between agents must be considered as an important factor.

To sum up, the Multi-Sarsa algorithm can be applied to large-scale service composition scenarios with good scalability.

3. Adaptivity

To simulate the dynamic environment, we randomly change the QoS values of candidate services during the learning process. In order to facilitate comparison, we fixate the number of agent for 4, state nodes for 100, and 1000 services for each node. We also cause QoS fluctuations between the 2000th episode and the 2500th episode. What's more, we allocated the sum of changed QoS values to each candidate services group of each state node averagely. Fig.5 gives clear illustration, no matter how big the volatility of QoS is, the Multi-Sarsa algorithm can converge by learning deterministically, and merely differentiates in convergence time. In short, the changes do not stop the optimization process, and the execution polices are still being optimized when the learning process goes on.

In conclusion, Multi-Sarsa does a good performance in large-scale and highly dynamic environment.

6 Conclusions and Future Directions

In this paper[2], we integrate on-policy reinforcement learning with multi-agent techniques for large-scale and adaptive service composition. First, we propose the new composition model called TMG-WSC, then utilize Multi-Sarsa algorithm in multi-agent scenario to find the optimal solution which is extended from single-agent Sarsa. Additionally, to ensure the convergence of the Multi-Sarsa algorithm, we introduce the fictitious play process which assures the unique equilibrium for equilibrium selection and incorporate it with the Boltzmann learning policy. Our experiments demonstrate that the proposed Multi-Sarsa performs well for large-scale and dynamic service composition.

However, we still have some room for optimizing the proposed framework. Firstly, we do not address the problem of failure services that may lead to entire paralysis of the composition solution. Therefore, reliability prediction or fault-tolerant technologies should be taken into the consideration. Secondly, we just consider the local QoS constraints, while users may only give a global QoS constraints, so how to decompose the global QoS constraints to the local is also a tough challenge. All in all, we will pay more efforts to optimize this framework.

References

1. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering 33(6), 369–384 (2007)
2. Beauche, S., Poizat, P.: Automated service composition with adaptive planning. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 530–537. Springer, Heidelberg (2008)

[2] This work is partially supported by NSFC Key Project (No.61232007) and Doctoral Fund of Ministry of Education of China (No.20120092110028).

3. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(2), 156–172 (2008)
4. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI/IAAI, pp. 746–752 (1998)
5. Gutierrez-Garcia, J.O., Sim, K.-M.: Agent-based service composition in cloud computing. In: Kim, T.-h., Yau, S.S., Gervasi, O., Kang, B.-H., Stoica, A., Ślęzak, D. (eds.) GDC and CA 2010. CCIS, vol. 121, pp. 1–10. Springer, Heidelberg (2010)
6. Hu, J., Wellman, M.P.: Multiagent reinforcement learning: theoretical framework and an algorithm. In: ICML, vol. 98, pp. 242–250. Citeseer (1998)
7. Jureta, I.J., Faulkner, S., Achbany, Y., Saerens, M.: Dynamic web service composition within a service-oriented architecture. In: IEEE International Conference on Web Services, ICWS 2007, pp. 304–311. IEEE (2007)
8. Könönen, V.: Asymmetric multiagent reinforcement learning. Web Intelligence and Agent Systems 2(2), 105–121 (2004)
9. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: ICML, vol. 94, pp. 157–163 (1994)
10. Littman, M.L.: Value-function reinforcement learning in markov games. Cognitive Systems Research 2(1), 55–66 (2001)
11. Maamar, Z., Mostefaoui, S.K., Yahyaoui, H.: Toward an agent-based and context-oriented approach for web services composition. IEEE Transactions on Knowledge and Data Engineering 17(5), 686–697 (2005)
12. Monderer, D., Shapley, L.S.: Fictitious play property for games with identical interests. Journal of Economic Theory 68(1), 258 (1996)
13. Moustafa, A., Zhang, M.: Multi-objective service composition using reinforcement learning. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 298–312. Springer, Heidelberg (2013)
14. Oh, S.C., Lee, D., Kumara, S.R.: Effective web service composition in diverse and large-scale service networks. IEEE Transactions on Services Computing 1(1), 15–32 (2008)
15. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. In: Proceedings of 2005 Autonomous Agents and Multi-Agent Systems(AAMAS), vol. 11(3), pp. 387–434 (November 2005)
16. Papadopoulos, P., Tianfield, H., Moffat, D., Barrie, P.: Decentralized multi-agent service composition. Multiagent and Grid Systems 9(1), 45–100 (2013)
17. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. University of Cambridge, Department of Engineering (1994)
18. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, vol. 1. Cambridge Univ. Press (1998)
19. Wang, H., Wang, X.: A novel approach to large-scale services composition. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 220–227. Springer, Heidelberg (2013)
20. Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 92–107. Springer, Heidelberg (2010)
21. Wang, X., Sandholm, T.: Reinforcement learning to play an optimal nash equilibrium in team markov games. In: NIPS, vol. 15, pp. 1571–1578 (2002)
22. Xu, W., Cao, J., Zhao, H., Wang, L.: A multi-agent learning model for service composition. In: 2012 IEEE Asia-Pacific Services Computing Conference (APSCC), pp. 70–75. IEEE (2012)
23. Young, H.P.: The evolution of conventions. Econometrica 61(1), 57–84 (1993)