Improvement of Self-optimizing in Selection and Composition of Services Using Reinforcement Learning Algorithm Based on Convex Hull

Hadis khorasaniNasab Abbasi
Faculty of Science& Computer Engineering
Shahid Beheshti University
Tehran, Iran
Email:Hadis.khorasani@gmail.com

Eslam Nazemi
Faculty of Science& Computer Engineering
Shahid Beheshti University
Tehran, Iran
Email:nazemi@sbu.ac.ir

***Abstract*—Web services are implemented by using many atomic or composite services. In a dynamic environment, some Web services require to select a service with defined Quality of Services(QoS) through runtime adaptation in changeable environments. In alignment with user satisfaction requirements, in selection of services a tradeoff between QoS should be considered, especially at runtime adaptation in dynamic environments. There are many methods for service selection and composite services with priority of QoS, but they do not predict optimizing service composition in the large scale environment. A self-optimizing method just continually adjusts the control service's parameters that pass to other services. In this paper, in a self-optimizing method, the goal and the procedure for selection and composition of optimal services are proposed. It includes three parts, services are limited in a defined scope by convex hull algorithm and then the optimal services are chosen by the divide-and-conquer algorithm. The optimal service selection is as input parameter goes to service composition algorithm. The QoS metrics taken into account and measured for the optimal service include response time, availability, throughput and reliability. The simulation results show that the system user satisfaction gradually increases by about 10% compared with the results of previous methods and show that the execution time is comparatively decreased by half.**

Keywords-text*; *self-adaption; self-optimizing; service composition; Reinforcement Learning; convex hull.

I. INTRODUCTION

Service-oriented environments have become more and more important in recent years, where various kinds of Web services and service-based processes are gathered within a certain domain or across domains [1][2]. They give people the ability to make, manage and share their own services, and make it possible to compose them based on a user's needs, providing them with extra value [1]. As reported in some previous studies on service selection, QoS attributes of atomic services are gathered for calculating the QoS of composite services in service composition environments [2][3].

Self-optimizing is considered a QoS optimization problem, choosing atomic services generating the highest QoS overall value as optimized solution [4][5]. It is presumed by most existing methods that QoS attributes pre-exist and QoS information of atomic services does not change. So, the ranking of declared QoS values is what determines the selection of a self-optimizing service. These approaches, however, have various constraints when the following problems are considered in the real environment. Firstly, service-oriented systems have various possible services because of the way they operate in distributed heterogeneous environments. Furthermore, existing services are ever-changing, so the selector should have the ability to adapt automatically to the dynamic environment. Finally, system should select optimal services based on QoS in reasonable time to meet user requirements.

In this paper, we propose a method for selecting and composition of services based on the self-optimizing. This is very important in service selection, because this kind of method can autonomously react to dynamic environments during its life cycle and adapt to them. This feature is very useful, and the reason is that nowadays, all services are distributed in large scale environment and they are always changing, so the system needs the method which can adapt to them. The self-optimizing method is also able to automatically improve behaviors by itself continually. It is considered as one of its features because one of the concerns for service composition in previous methods is selecting optimal services based on QoS.

The rest of the paper is organized as follows: In Section II, related works are discussed. Section III introduces a self-optimizing Method and process variability. Section IV demonstrates the validity of the proposed method by a series of simulation experiments. Finally, Section V draws some conclusions.

II. RELATED WORK

In this section, some related work from the perspectives of the self-optimizing service selection and composition based on QoS is introduced.

With the growth of Cloud Computing, Service Oriented Architecture (SOA) and Software as a service, possible services with similar functions but different QoS increase in numbers, which has made it far more difficult to select and compose services [5]. This has led to growing research in composition of QoS-aware Web service in SOA and Service-Oriented Computing (SOC) fields [6][7][8]. Yet, service composition is currently done mostly via approaches that utilize a semi-optimal approach relying on a single goal, instead of using Pareto optimal solutions that take into account the balance between various QoS objectives [9].
One sophistication that may arise is when a user quickly requires a service with a specific cost and certain performance, yet with increased availability. In real world usage, however, distinct dimensional attributes may not be compatible with one another.

Availability pair and the time it takes to respond are two of these contrasting characteristics. This means that QoS of optimal service composition has low response time and high Availability. Thus, reaching the optimal solutions given the different rules on the measured QoS and the competing goals is an NP-hard challenge. QoS weighting is used by some algorithms to dynamically adjust to these tradeoffs via a feedback controller [5]. However, the QoS weight sum method, has some constraints as follow: 1) weight vector directly influences the solutions, for which awareness of the problem in advance is needed; 2) there is a limited selection of solutions, which are not well-distributed; 3) as the scale of problem rises, so does the complexity and the time executed is dramatically increase; 4) if solutions are in out of reach areas of the Pareto front, Pareto optimal solutions may not be found; and 5) Clients may actually want to see a list of possible services, while only one, i.e., the Pareto optimal, is offered. Another field of work concerns utilizing the skyline operator to measure the real Pareto front [10][11]. The first one, the Bottom Up Algorithm, measures the biggest sections' workflow in order to boost the effectiveness of the process. The second Algorithm, consecutively provides the Pareto optimal services. But having temporal complexity in these algorithms is not possible, as the Pareto front might exponentially become larger with more tasks in the workflow. Also, the way the search area is pruned by the skyline operator means some possible services could be put aside even before selection occurs to meet the tradeoffs between multi QoS objectives, Pareto optimal workflows were set by Mostafa and Zhang [9]. It is provided tradeoff in linear domains with convex hall as well as the optimal Pareto front solution. Also Quick hull operator used to prune the search space may have polynomial time complexity because in the large number of workflow tasks, it has execution time at O(n2). Reinforcement Learning algorithm [12] has been introduced for solving sequential decision-making issue and makes learner optimal policy of Markov Decision Process (MDP) for services composition at runtime. This system can adapt to the dynamic environment by calculated reward function. It is supposed to receive reward value, which is equivalent to the cumulative reward of all the executed services [12]. However, there can be challenges as to how existing multi-goal service composition methods can work in dynamic environments. For example, to determine QoS value mathematical methods are used that presume a static environment. Once there are changes in the environment, there are no strategies for the system to deal with the emerging QoS. Also, some of these methods make use of explicit models so as to determine which services are chosen. No Rue is present in this model for addressing a new QoS parameter. Furthermore, in multi-object method, services are chosen by the weight which is defined in a static environment. Hence, the weight of a new service or an obsolete service in static environment cannot be dealt with changing environment. Finally, a near-optimal runtime policy is used by adaptive service Composition, meaning that in each of the system's lifecycles service composition is not optimal and the system cannot self-optimize.

In this paper, a new self-optimizing method is proposed. This method is based on Reinforcement Learning for calculated user satisfaction by reward function. A self-optimizing system is one that dynamically optimizes the operation of its service composition while it is running. The optimizer just continuously adjusts the control service which is selected based on QoS to compose with other services. In this system two main goals are followed, service composition can adapt with changing environment and system can optimize service composition based on QoS automatically and also multi-objective service composition approach is considered. In order to achieve those targets, this paper follows these steps: First, new search algorithm in convex hull is introduced for selecting multi-objective optimal services. Then, use Reinforcement Learning algorithm for compute services user satisfaction. This algorithm will obtain initial knowledge of the service selection from the divide-and-conquer algorithm and it will be optimized when service composition is based on optimal service.

III. PROPOSED METHOD FOR COMPOSITION OF SERVECES

In this section, the self-optimizing method is introduced for selection and composition of services in order to improve Reinforcement Learning method for service composition. In previous method, proposed service composition is not optimized in each life-cycle system. In this paper is proposed service composition that is optimized continuously. A self-optimizing composite service is one that dynamically own optimizes the service composition while it is running, so it needs to have some kind of rules that can follow in the system. The goal of the self-optimizing method is to maximize service composition based on QoS at all times. This method has ability to implement in the Large scale services and follows the goals like user satisfaction, being self-adaptive to the changeable environment, and presenting an automatic optimum service composition based on QoS. Before main algorithm is proposed, the schema of the self-optimizing method is mapped in MAPE_K loops, and the self-optimizing cycle in order to define the issue's scope should be explained.

A. *Adaption Loop*

Self-adaptive software is based on a closed-loop mechanism which is called the MAPE-K loop for autonomic computing, and includes the Monitoring, Analyzing, Planning and Executing functions. Self-optimizing is one of the most remarkable properties of self-adaptive systems. Therefore, my recommended plan for the self-optimizing is mapped in the MAPE-K loop in order to shows the workflow of the method.

Accordingly Figure 1, the QoS values are defined. In this paper response time, availability, throughput and reliability are as QoS parameters which are collected in “monitoring” step. Then, the collected data are analyzed in the “analysis” step. In this step, the value of those parameters is normalized. Then especial selected algorithm is executed in “Planning” stage. Optimal selected service as input parameter goes to the “execution” stage. In this stage, by Reinforcement Learning algorithm the best services are predicted for the user. MAPE-K loop is based on learning so there is one stage to share Knowledge with each part for predicting the system's behavior. In this case, the system needs to predict service composition to achieve high user satisfaction. Using reward function in Reinforcement Learning algorithm, the learning process is defined.

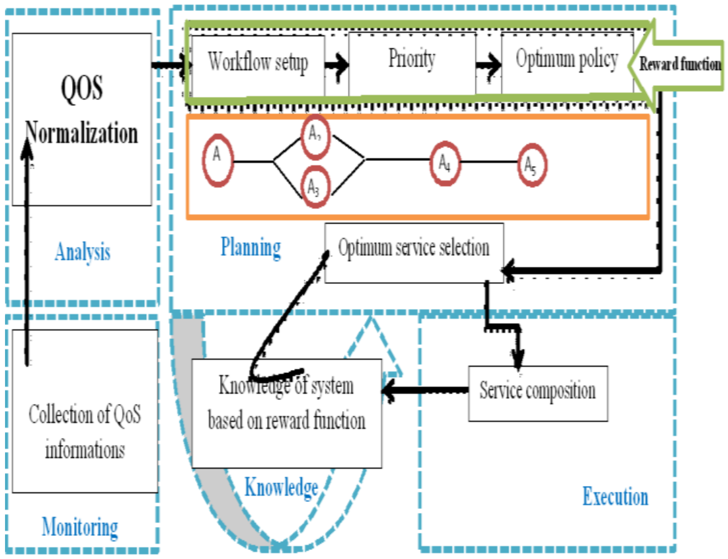

Figure 1. Abstract schema of the proposed method based on MAPE-K architecture.

B. *Self-optimizing cycle mapping*

The scope of service selection and composition is mapped on the cycle of the self-optimizing, which is depicted in Figure2.

The first step is “analyzing the current state”. It is defining some QoS parameters which are normalized in a certain data range. The second step is “determining the goal of system”. In this step the main goal of the system is defined. In this paper, the main target is selecting optimal services among distributed services, according to (1).

$$\pi \coloneqq S \rightarrow A. \qquad (1)$$

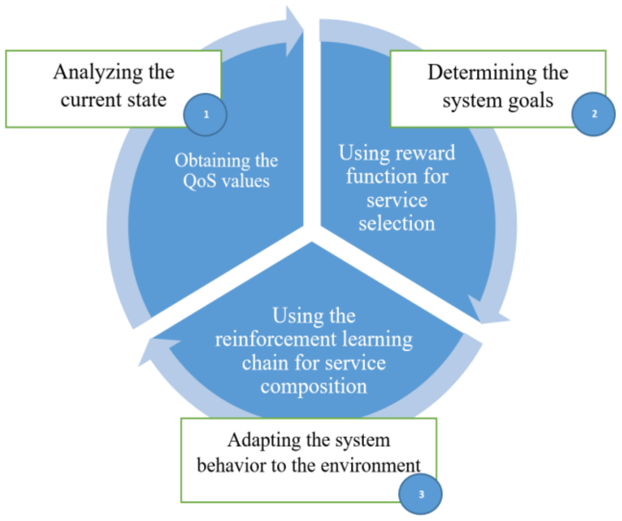
Analyzing the current state
1
Determining the system goals
2
Obtaining the QoS values
Using reward function for service selection
Using the reinforcement learning chain for service composition
Adapting the system behavior to the environment
3

Figure 2. Self-optimizing cycle mapping for the selection and composition of service.

Geometric convex hull operator is used in order to reduce number of services. The convex hull is the smallest convex polygon that encloses all points in specific space. Here, points are services which should locate in defined geometric place. The service is limited as (2).

$$CH(S) = \{s_{i_1}.s_{i_2}.\dots.s_{i_m}\}.m \le n.s_{i_j} \in S.j = 1.2.\dots.m. \qquad (2)$$

Equation (2) shows that the convex hull of services includes "m" is members and "n" is the number of available services. So the number of services, which are known as members, are smaller than the number of available services. New services are adding in specific space by incremental convex hull algorithm. This process is implemented in three steps: first, place the visible facets for the services; the boundary of the visible faces is the set of horizon ridges for the services. Second, construct a cone of new facets from the service to its horizon ridges. Third, eliminate the visible facets. Therefore, the convex hull of the new service and the previous services is formed. Moreover, Convex hull is clustering services and categorizes them in a finite-dimensional space to set services in. In (3), "d" is number of dimensions in convex hull. In this paper, two dimensions are used. So, this algorithm translates the interior service to half spaces by dividing offsets into coefficients. Dimensions are allocated two QoS parameters, which are analyzed in first step, it shows (3).

$$Q = \{Q_i \in R^d | \forall Q_i \in Q.\exists s_i \in S\}. \qquad (3)$$

According to Figure 3, the response time and throughput are determined with two-dimension space. Services are divided into four zones by the clustering of convex hull. It determines services' suitable zones according to the value of their QoS. For example, the optimal service placed in (b, c) zone which has highest throughput and lowest response time.

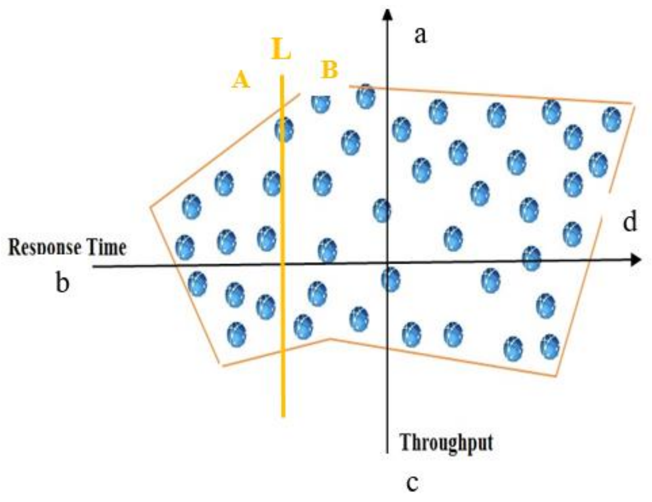

Figure 3. Selecting optimal services by clustering and divide-and-conquer.

After optimal services are determined in a defined zone, optimal service is selected by divide-and-conquer algorithm. This algorithm has O[$n \log n$] execution time in all cases. This algorithm operates as follows. In the first step, vertical line L divides services into two subsets A and B, each containing N/2 services (seen Figure 3). Since every service in A and B cluster has an x-value and y-value, in next step, x-value and y-value of each service in A and B cluster are compared together. In the last step, rank of each axes for any services is defined. As an example, when x-value of B dominates x-value of A but the y-value of B is not greater than the y-value of A. Comparing services based two QoS parameters are continued until the optimal service is selected; this is a recursive algorithm (4) and (see Figure4).

$$\pi^* \coloneqq Dhull_{\pi}(s_t).(\forall s_t \in S) \qquad (4)$$

```
Algorithm 1: selected optimal services based on convex hull and divide-and-conquer algorithm
1: initialize S, Q
2: /* clustering service in convex hull
3: For all atomic service $s_i \in S$ do
4:    For all QOS attribute $q_j(s_i) \in Q_{si}$ do
5:        Get convex hull  $H(s_i) = (q_j(s_i)\ldots, q_j(s_i))$
6:    End for
7:  End for
8:      Get divide-and-conquer on$\{H(s_i)\}$ for i step
9:          If get an optimal service
10:             Return s
11:         End if
```

Figure 4. Selected optimal services based on convex hull and divide-and-conquer algorithm.

The third step of self-optimizing cycling is composing optimal services. In this section, reinforcement learning algorithm schema to orchestrate service composition is introduced. In this algorithm, the task of the learner or decision-maker is to learn a policy based on reward function. The complete learning process is depicted in the algorithm in Figure 5[12]. In this algorithm, the task of the learner or decision-maker is to learn a policy based on reward function [12].

```
Algorithm 2: Baseline Reinforcement Learning Algorithm for Service Composition
1: initialize Q (s, a)
2: for each episode do
3:      s← $s_0$
4:      for each step of episode do
5:          Choose a $\epsilon$A(s) based on €-greedy policy
6:          Execute a, observe reward r and new state s'
7:  Q (s, a) ← Q (s, a) + α [ r+ γ max$_{a'}$ Q (s', a') - Q (s, a)]
8:          s ←s'
9:      End for
10: End for
```

Figure 5. The Baseline Reinforcement Learning for Service Composition [12].

In this algorithm, initial state s_0, terminal state s and Q(s ,a) are defined. Q (s, a) is simulation of observed reward. In each episode (round), the learner starts from the initial state s_0, and takes a sequence of actions by following the ϵ-greedy policy (which is introduced subsequently). As line 7 shows, optimal service is chosen based on ϵ-greedy policy and old Q (s, a) value is completely replaced with the new value of reward function. Rate of learning is α, which is quantity between 0 and 1. The discount factor is γ that reflects the learning policy. Both value of α and γ are different in differ issue. The value of €-greedy is (€ < 1). The most significant part of this algorithm is computing the reward function which calculates user satisfaction. In this paper reward function is used as well as this algorithm [12] to predict service composition and observe user satisfaction. The policy of reward function is determined according to (5).

$$R(s) = \sum w_i \times \frac{Att_i^s - Att_i^{min}}{Att_i^{max} - Att_i^{min}} \qquad (5)$$

where Att_i^s shows current value of the ith attribute of service s, and Att_i^{max} and Att_i^{min} show maximum and minimum value of Att_i for all services. W_i is the weighting factor of Att_i. This value is positive if users prefer Att_i to be high value (e.g. throughput). W_i is negative if users prefer Att_i to be low value (e.g. response time).

C. *Self-optimizing method for service composition*

In this selection, a self-optimizing method is proposed. According the self-optimizing cycle, the main goal is selecting optimal services based on QoS through distributed services. Response time, reliability, availability and throughput are the QoS parameters which analyze and compute the value of them for the self-optimizing method. This method has been shown in Figure6. According to the self-optimizing algorithm, Services and QoS parameters are initialized. Also, Q (s, a) as seen in Reinforcement Learning algorithm at the start of this algorithm is initialized. Line 3 to line 5, clustering convex hull based on QoS is calculated. All services in the convex hull are shown with H(si). The main purpose is selecting optimal services which is done with divide-and-conquer algorithm of convex hull. The optimal service selected is imported as an initial service parameter to composition algorithm. Then new action and next state (s') are defined in line 16. Reward function is calculated to compute user satisfaction in line 17. In order to predict services composition in next step, this algorithm needs to update the value of reward function for service selected according to calculate quantity of γ , α and new reward value of next optimal service is selected according line 18. In this line, new optimal service is selected

```
Algorithm 3: Self-Optimizing Algorithm for Select and Composition Service
1: initialize S,Q and Q (s, a) arbitrarily $\forall s$,a
2: /* clustering service in convex hull
3: For all atomic service $s_i \in S$ do
4:    For all QOS attribute $q_j(s_i) \in Q_{si}$ do
5:        Get convex hull  $H(s_i) = (q_j(s_i)\ldots, q_j(s_i))$
6:    End for
7:  End for
8: /* compute composition service
9: For each episode do
10:  For each step of episode do
11:      Choose action a ∈ A(H (s$_i$))
12:      Get divide-and-conquer on$\{H(s_i)\}$ for each step
13:          If get an optimal service, then
14:             Return s and DAC (s, a)
15:          End if
16:      Take action a and observe next state $s' \in$ DAC($s$)
17:     Observe reward vector $\vec{r} \in \vec{R}$
18:   Q (s, a) ← Q (s, a) + α [ r + $\gamma$ DAC Q (s', a') - Q (s, a)]
19:     s← s'
20:  End for
21: End for
```

Figure 6. The self-optimizing for service composition.

based on divide-and-conquer algorithm of convex hull, it shows with DAC (s', a').

In this way, the self-optimizing strategy provides maximum user satisfaction. In each cycle of system, the services that have maximum user satisfaction are suggested to the user. Also, this algorithm can adapt itself to the dynamic environment. Service selection accuracy provides full potential of each service. In order to compare the result of this method with previous methods, consider to calculated average value of reward function. If the system gets higher score in reward function than other methods, it is optimal behavior in service selection and composition.

IV. EVALUATION OF THE PROPOSED METHOD

One of the vita factors in tourism website produces Web service with high QoS which are available and can respond to user requirements in reasonable time. Customers on the Web want to do anything conveniently and simply, such as booking hotels and flights with one service, which is called a tourist package, or take the best service offer from the system. The significant concern in a self-optimizing tourist website is how to increase user satisfaction gradually. So, the propose method is implemented on a tourist website, which is composed of services and adjusts to dynamic environment in order to meet user requirements.

In this paper, the self-optimizing occurs at the source code level as done by the program. The tourist website was implemented by C#.net and Asp.net. The website is based on MVC Architecture and SQL Server 2016 database. The database was designed based on the normal equation in such a way that it does not have redundancy at updating time. Web services are provided from valid dataset [13] which has 356 real Web services. Those Web services have nine Quality of Web Service (QWS) attributes, which are measured with a commercial benchmark tool. The advantage of this dataset is that Web services are collected from public source discovery, integration, registration, search engines and service portals. It is remarkable that each service was tested over a ten-minute period for three consecutive days. Therefore, calculation of QoS was ignored. But before using the value for each QoSs, they should be normalized because they are distributed over a wide range. Equation (6) was used for normalization, as introduced in [14].

$$Q_i' = \frac{Q_i - \min_i Q_i}{\max_i Q_i - \min_i Q_i}. \qquad (6)$$

The main goal of the tourist website is that users choose a travelling destination. The website is based on two scenarios, both of which incorporate selection and composition of services to answer user requirements.

The first scenario is that the user fills a form in order to access a weather service. Then the user submits the information to the website in order to get a list of weather services for their destination. The website finds a list of weather services and presents the best ones as the result of the queries to the user. The second scenario consists of two possible ways. In the first way, flight and hotel services are chosen same as weather services. But the second way is definitely deferent, because service of hotels and flights is represented in one service, which is called tour package. User requests especial tour package which includes defined flight and hotel. The tourist agent requests to choose defined flight services. Then, it is receiving ID of flight services to send hotel service selector. At the end, it is receiving ID composition service which consists of flight and hotel services. The system can predict the best tour package service for users who enter the same information. The last scenario is payment services just like the first scenario. In Figure 6, each state is shown.

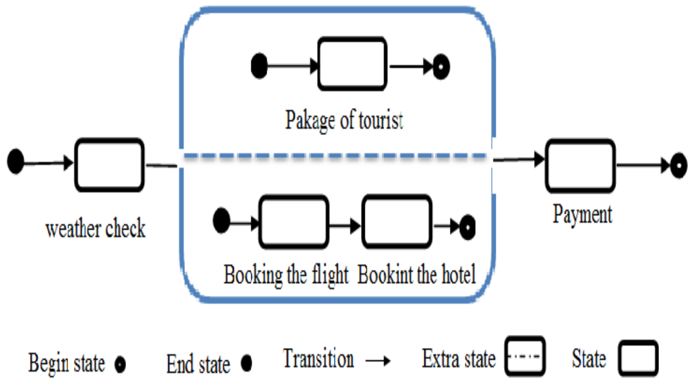

Figure 6. The work flow of tourist scenario.

The second scenario is the main scenario in order to obtain comparison between this method and the previous one. The previous method is Reinforcement Learning algorithm which is developed in the tourist website as well as the self-optimizing method. The reward function is calculated with deferent QoS parameters. In the experiment results, the discount factor ϵ is set at 0.9 and as the amount of α is set to 0.2 (Figure 7). Also according to Figure 8 , the value of γ is set to 0.5. All of the experiments were conducted on a Sony laptop with Core i5 3.1GHz processors and 12GB RAM, running Windows 7. All proposed services are observed on the tourist website after 2 minutes.

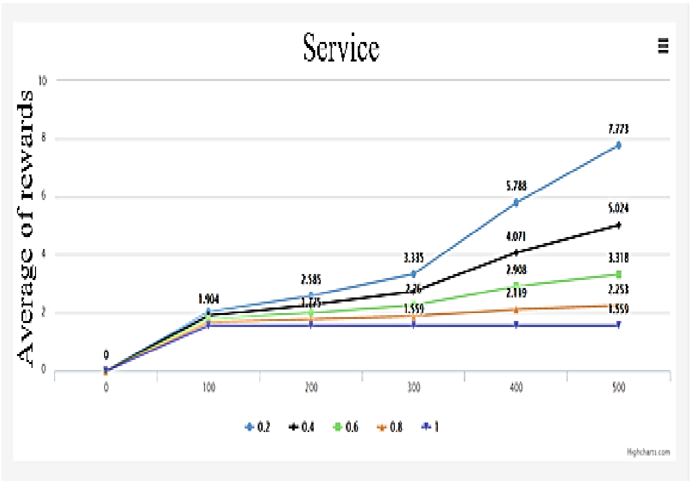

Figure 7. Choosing bestα.

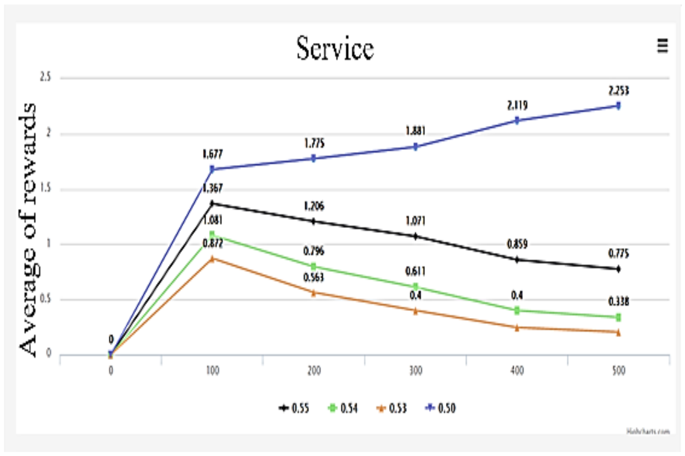

Figure 8. Choosing the probability γ.

In the first stage of evaluation, it is shown how Reinforcement Learning policy can be improved with the self-optimizing method. Reinforcement Learning algorithm is executed with zero knowledge about the QoS of the component of service composition and in some episodes the value of reward function is not better than the last stage and shows uniform behavior (see Figure 9). Proposed method is improved by adding a new policy about service selection for service composition, it has the ability to self-behavior and increase user satisfaction by achieving higher values of reward function. This simulation fixed six episodes; the number of services in each episode is increased in order to show the proposed method in large scale of services has the same self-optimizing attitude.

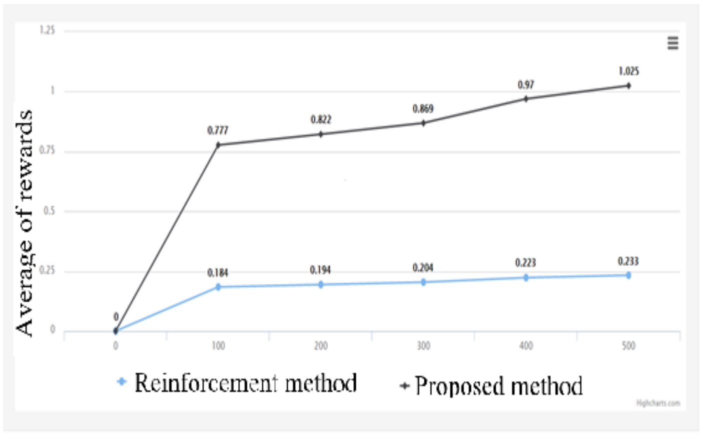

Figure 9. Comparing the proposed method with the Reinforcement learning method for increased user satisfaction.

In the second stage of our evaluation, it can be seen how a self-optimizing service composition adapts to the changes of the environment and value of the reward function, which represents user satisfaction, is steadily increased. Changing environments are simulated by periodically changing the QoS attributes of the services. At first environment is changed by 5%. It means, 5% of basic services are added to the environment with the uniform probability distribution formula. Then Reinforcement learning algorithm and the purpose algorithm are executed. Figure 10 shows the growth of the cumulative reward during the self-optimizing process. In comparison, increasing the change rate in Reinforcement learning algorithm has delay because it has to identify QoS and needs to learn optimal execution policy. Conversely, the reward value of the self-optimizing method is comparatively higher and changes do not stop the optimizing process. In second simulation, the environment is changing by 10% and the third simulation based on 15% .When the environment changes more and more, growth rate satisfaction of the self-optimizing method is more visible. In the third stage of our evaluation, Figure 11 shows how the self-optimizing service composition outperforms the reinforced algorithm in a large scale environment. In this evaluation, environment scale is represented by the number of services used in every tourist workflow. At first, hotel and flight services are increased up to 300, then reward function of the proposed approach is measured. The reward value depicts the user satisfaction. In the second and third picture reward functions are measured based on 400 and 500 services respectively. Comparing the proposed method with the Reinforcement learning method in the Large scale environment, self-optimizng method shows more satisfaction than the Reinforcement learning method. The fourth experimental results include test 1, test 2 and test 3. In this experiment, optimal services with low response time, high availability, high throughput and high reliability are selected. The results of test 1, as depicted in Figure 12, clearly show that the optimal tourist workflows have achieved high throughput, and high reliability among 50 services or lower response time and high availability. The outcomes of test 2 are represented in Figure 13, they support test1 statement, regardless of the bigger number of concrete Web services assigned to each task (100 services), as the optimal workflows obviously continue representing the same trend with lower response time and high availability, high reliability and high throughput. Finally test 3, as represented in Figure 14, has the same trends as test 1 and test 2 with large number of services (150 services). As a result, the size of environment does not affect selecting optimal services based on QoSs for each task in tourist website.

In Figure 15, the proposed method executed composed services in half the time of Multi-Object Service Composition algorithm which was introduced by Mostafa and Zhang [9], the reason is clearly observed; the previous algorithm used fast convex hull, whose execution time in the worst-case is O[n^2] in contrast execution time of the self-optimizing method in all cases is O[$n \log n$]. Therefore, by this chart, different levels of execution time of the two algorithms in large scale are more distinguished

V. CONCLUSION

The main purpose of this paper is to introduce a self-optimizing method in order to select and compose services. Optimal services are selected based on QoS and composed with other services to answer user requirements. The significant algorithm in the self-optimizing method is the Divide-and-conquer algorithm used for selection. The reason is that the execution time of this algorithm is O[$n \log n$] in all cases. So the self-optimizing method behaves similarly on the large scale, with shorter execution times. This time is half that of pervious method [9]. Therefore, when the tourist wants to take travel services, it can get optimal services in a reasonable time. Moreover, the Reinforcement Learning policy [12] has zero knowledge about the QoS of the component services, it takes some time to know services in large environment and predict user behavior.

Therefore, the rate of rewards value remains in consistent level when making comparison between

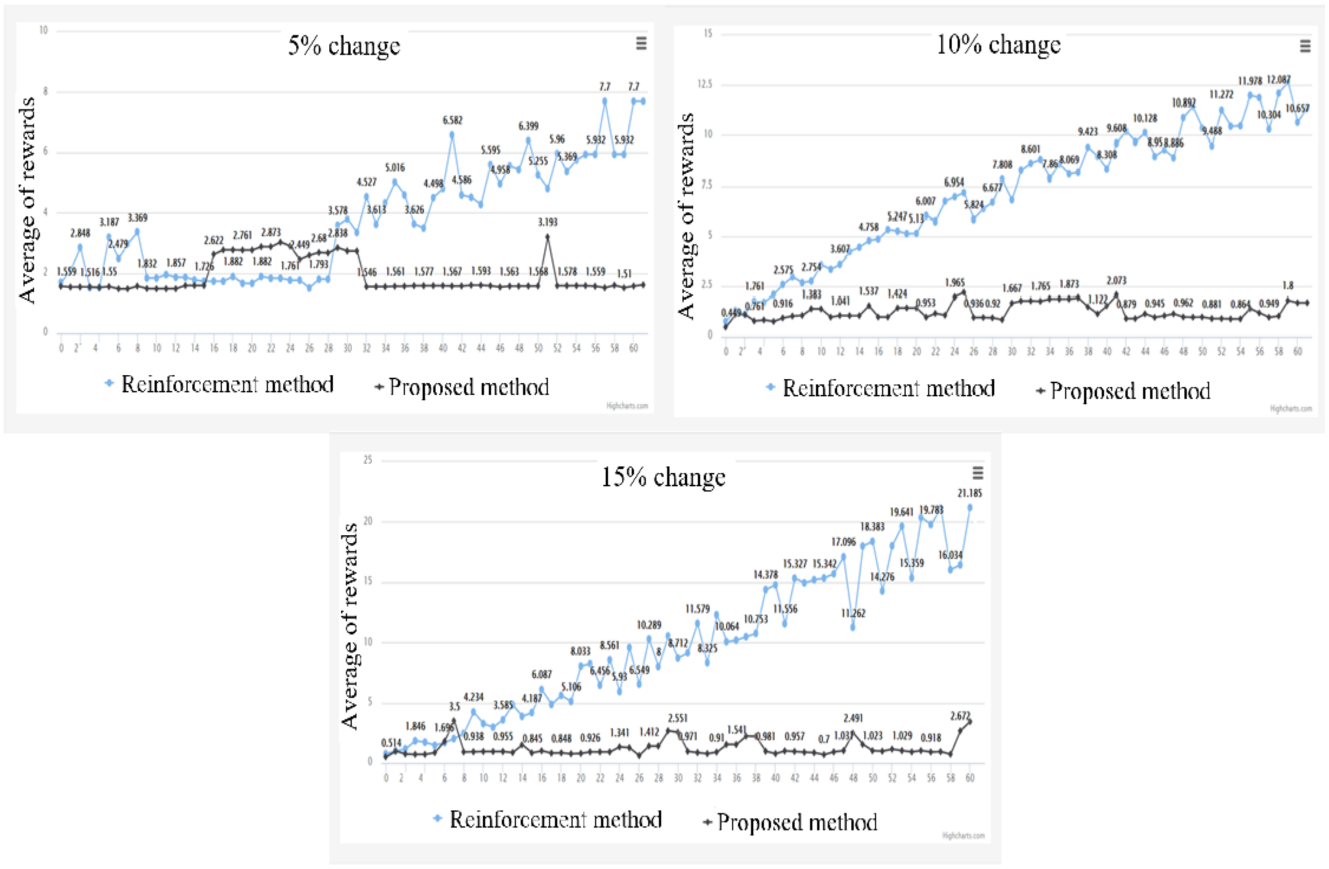

Figure 10. Comparing the self-optimizing method with the Reinforcement Learning method in dynamic environments.

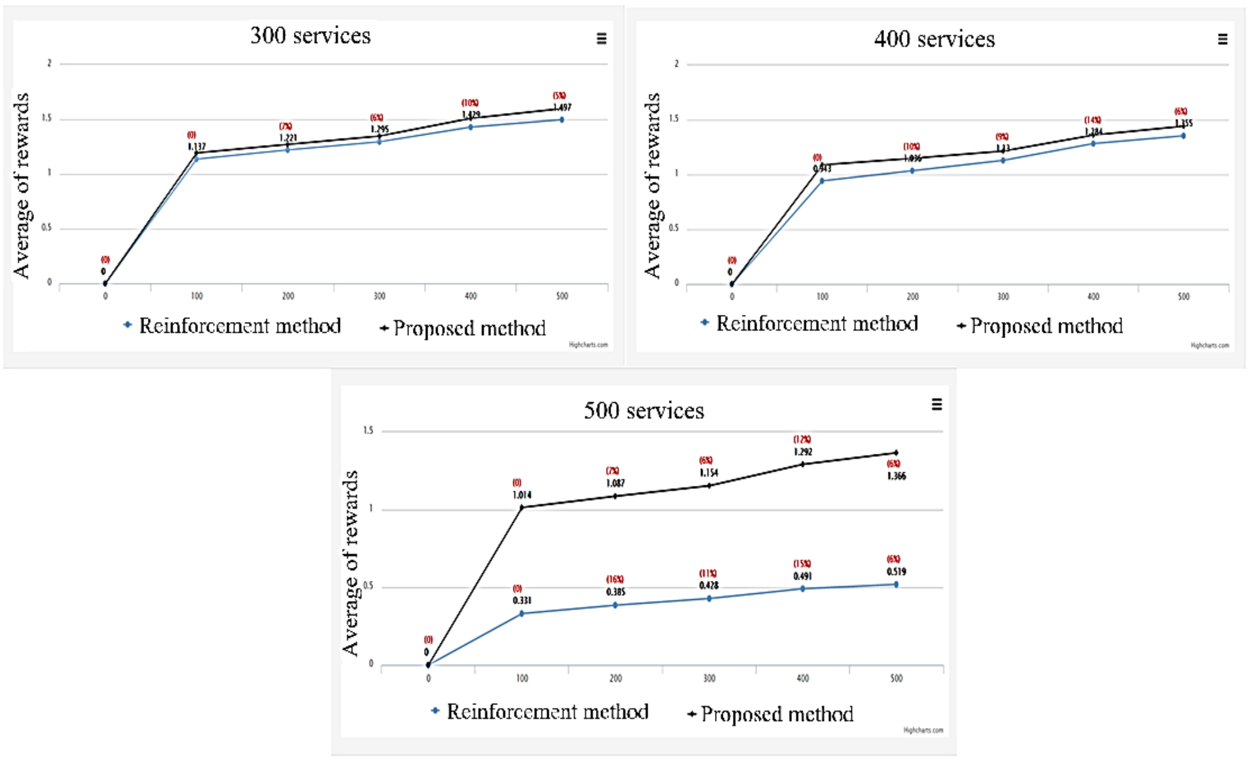

Figure 11. Comparing the performance of the proposed method with the base method.

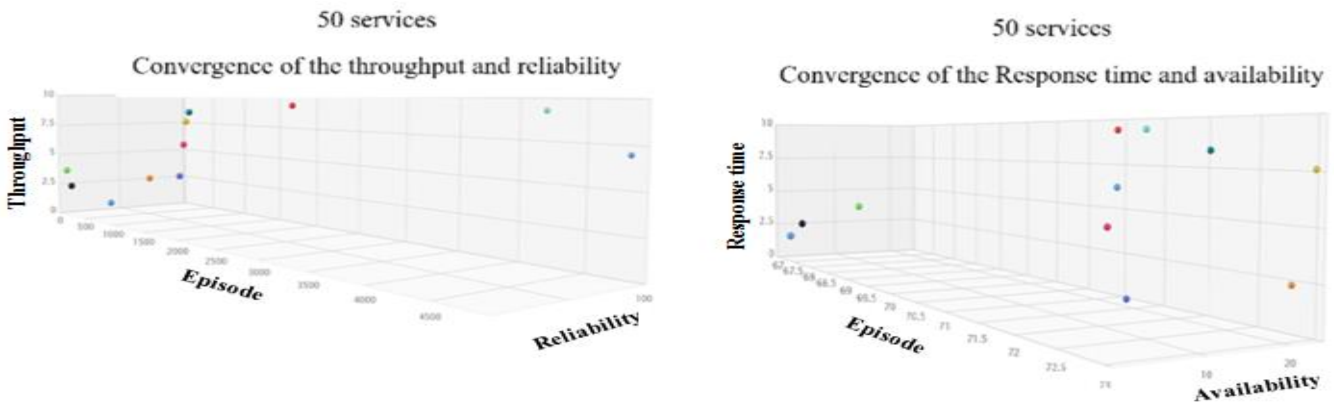

Figure 12. Optimal service based on QoS is Selected among 50services.

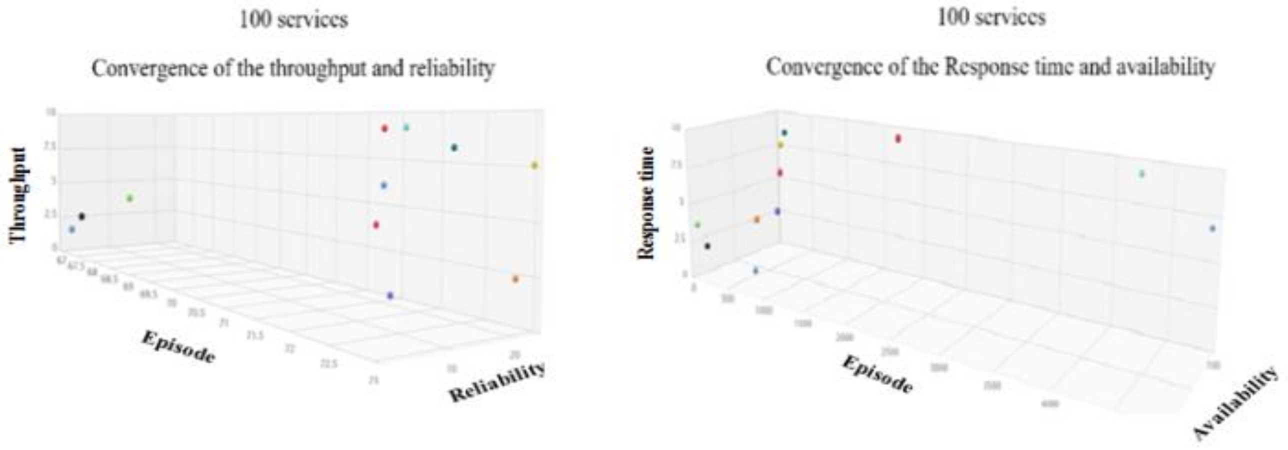

Figure 13. Optimal service based on QoS is Selected among 100services.

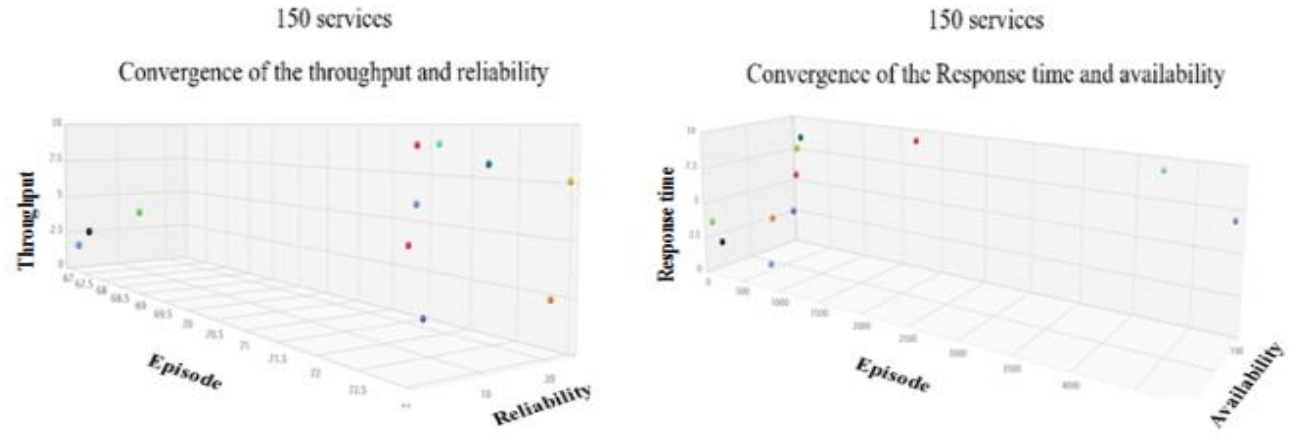

Figure 14. Optimal service based on QoS is Selected among 150services.

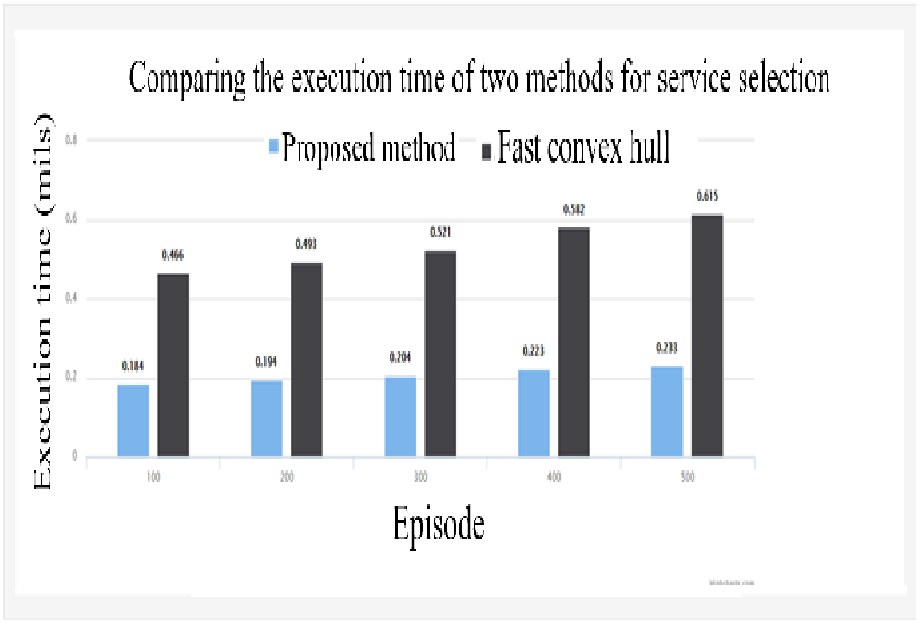

Figure 15. Comparing the proposed method with the method of selection service by fast convex hull.

two steps of this algorithm. It has near optimal execution policies efficiently. In this paper, Reinforce Learning algorithm is modified with new policy to selected optimal services. This policy optimizes service composition in each cycle-life. On the other hand, some QoSs is differing in best value. For example, response time is lower and availability is high in optimal services, so convex hull algorithm provides clustering in two diamonds which can choose optimal services with different values. The experiments show the efficiency of this algorithm is optimized in each steps.

Dynamic environment is one of the issues this day for websites like tourist website because services are always changing with improved performance or replaced with new services. The Self-optimizing method is a good solution to use here; it can optimize itself in large scale and adapt to a dynamic environment.

The proposed method takes about 2 ms to compute QoS. Therefore, it is useful for non-real-time systems like tourist systems. But it is not useful for real-time systems where time is very critical.

The future work is set to study the self-optimizing and self-reconfiguring method together. Therefore, when the service selection and composition fail, there are methods to help the system to reconfigure itself automatically. The self-optimizing and self-reconfiguring are implemented in many systems simultaneously but up to now, they are not implemented in service-oriented systems. Using this feature can also improve the proposed method.

References

[1] Donghui Lin, Chunqi Shi, and Toru Ishida," Dynamic Service Selection Based on Context-Aware QoS ", In 2012 IEEE Ninth International Conference on services Computing.

[2] A. Moustafa and M. Zhang, “Learning efficient compositions for QoS-aware service provisioning,” in 2014 IEEE International Conference on Web services, ICWS, 2014, Anchorage, AK, USA, June 27 - July 2, 2014. IEEE Computer Society, 2014, pp. 185–192.

[3] A. F. M. Huang, c.-w. Lan, and S. J. H. Yang, "An optimal QoS-based Web service selection scheme," Inf. Sci., vol.179, pp. 3309- 3322, September 2009.

[4] G. F. Franklin, J. D. Powell, and A. E. Naeini, Feedback control of dynamic systems. Prentice Hall, pp. 928, 2008.

[5] B. Chen, X. Peng, Y. Yu, and W. Zhao, “Requirements-driven self-optimization of composite services using feedback control,” IEEE Transactions on Services Computing, vol. 8, pp.107-12, January, 2014.

[6] L. Z. Zeng, B. Boualem, D. Marlon, J. Kalagnanam, and Q. Z. Sheng, “Quality driven Web services composition,” in Proc. of the 12th International Conference on World Wide Web, pp. 411-421, 2003.

[7] C. W. Zhang, S. Su, and J. L. Chen, “Genetic algorithm on Web services selection supporting QoS,” Chinese Journal of Computer, vol. 29, pp. 1029-1037, 2006.

[8] R. L. Graham, “An efficient algorithm for determining the convex hull of a finite planar set,” Elsevier, vol.1, pp. 132-133, 1972.

[9] A. Mostafa and M. Zhang, "Multi-objective service composition in uncertain environments," IEEE Transactions on Services Computing, vol. 99, pp.1-1, June2015.

[10] Q. Yu and A. Bouguettaya.” efficient service skyline computation for composite service selection”. *IEEE Trans. Knowledge and Data Engineering*, volume 25(4), pages 776–789, 2013.

[11] Q. Yu and A. Bouguettaya.” computing service skyline from uncertain qows”. *IEEE Trans. Services Computing*, volume3(1), pages 16–29, 2010

[12] H. Wang, X. Zhou, W. Liu, W. Li, and A. Bouguettaya, “Adaptive service composition based on Reinforcement Learning,” 8th International Conference ICSOC, pp. .92-107, 2010.

[13] Available from: https://github.com/wsdream/wsdream-dataset.

[14] Y. Chen, L. Jiang, J. Zhang, and X. Dong,” A Robust Service Selection Method Based on Uncertain os”, Mathematical Problems in Engineering, vol2016, p. p10, January 2016.