Received: March 30, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

PGAQK: An Adaptive QoS-aware Web Service Composition Approach

Doaa Elsayed[1]* Eman Nasr[2] Alaa El Din Ghazali[3] Mervat Gheith[4]

[1] *Department of Information Systems and Technology,*
Institute of Statistical Studies and Research, Cairo University, Cairo, Egypt
[2] *Independent Researcher, Cairo, Egypt*
[3] *Department of Computer and Information Systems, Sadat Academy for Management Sciences, Cairo, Egypt*
[4] *Department of Computer Science,*
Institute of Statistical Studies and Research, Cairo University, Cairo, Egypt
* Corresponding author's Email: doaa.hani@hotmail.com

Abstract: Web Service Composition (WSC) has attracted considerable attention and research to support Service Oriented Architecture (SOA). WSC constructs complex applications by combing atomic Web services together to achieve users' requirements. The selection of the best Web service that fulfils the Functional Requirements (FRs) and optimizes the Quality of Service (QoS) requirements, such as response time, cost, reliability, etc., is a critical part of WSC, especially in a dynamic environment. Since requirements frequently change, this demands that WSC must be adapted to provide the most suitable composite services and fulfil the users' requirements that emerge. This paper presents a new hybrid approach for dynamic optimization of WSC using Parallel Genetic Algorithm (PGA) based on Q-learning, which we integrate with K-means clustering. We call it PGAQK. Q-learning is utilized to generate an initial population to enhance the effectiveness of PGA. K-means clustering is used to cluster Web services based on a fitness function. It is applied in the *mutation* operator, and used for Web service substitution to prune the Web services in the search space to find the best Web services for the environment changes that might occur at runtime. To the best of our knowledge such a hybrid approach has not been used for WSC before. We implemented PGAQK over the .NET Framework using C# programming language. A series of comparable experiments were carried out showed that PGAQK outperforms traditional PGA and Q-learning approaches in terms of fitness values. PGAQK allows WSC to modify itself dynamically and achieve a better fitness value to fit the changing environment, where the characteristics of composite web services continue to change.

Keywords: Web service composition, K-mean clustering, Parallel genetic algorithm, Quality of service, Q-learning, User requirement.

1. Introduction

Service Oriented Architecture (SOA) is an architectural paradigm to design and develop distributed systems in heterogeneous environments that are independent of any language and platform [1, 2]. The key technology for building software applications in SOA is the Web service. Web services are software designed to support interoperability, which is the ability of a system to work with another system to achieve a common goal [1, 2]. A Web service is described in Web Services Description Language (WSDL), which is an eXtensible Markup Language (XML) document. In complex business transactions, an atomic Web service is not sufficient to respond to the users' requirements, and often Web services should be combined together to achieve a specific goal [2]. The process of combining atomic Web services together to construct a complex task to meet users' requirements is called Web Service Composition (WSC). The descriptions of a Web service could be divided into Functional Requirements (FRs) and Non-Functional Requirements (NFRs) [3]. FRs represent functionality of a Web service (i.e., execution of a Web service), and NFRs represent quality of the Web

service, such as Quality of Services (QoS), scalability, usability, maintainability, etc.

In the context of WSC, an Abstract Web Service (AWS) represents the FRs of the service, while a concrete Web service represents a real service that implements the FRs specified by an AWS [4]. The number of candidate Web services on the internet has increased dramatically which raises the complexity of the selection of a concrete Web service. The selection of the best Web service that fulfils the FRs and optimizes QoS requirements is called QoS-aware WSC, which is an NP-hard problem [5].

A major challenge for QoS-aware WSC is how it deals with dynamic environments. In dynamic environments the properties of a Web service often change during design and runtime, such as availability of Web service, changes in QoS parameters and composition requirement. Therefore, the dynamic WSC approaches should be able to adapt to meet changing requirements that occur at design time and runtime.

The QoS aware WSC problem is solved by using exhaustive algorithms and mathematical programming-based algorithms, such as Linear Programming (LP), Linear Integer Programming (LIP) and Mixed Integer Programming (MIP) [5]. These algorithms are inefficient in the large set of candidate Web services. The search process of these algorithms are time-consuming and will violate the real-time execution constraints in the Service Level Agreement (SLA) contract. Genetic Algorithms (GAs) are the most widely used Bio-inspired algorithms [5]. GAs are unconstrained procedures to solve multi-objective optimization problem. Also, they are effective and efficient to find the optimal solution in the large set of candidate Web services. The two major limitations of GAs are randomly generating the initial population and their time consuming nature. We attempted to address finding the optimal solution in QoS-aware WSC in a recent paper [6] by using Genetic Algorithm (GA) with Q-learning. Q-learning was used to generate the initial population to improve the effectiveness of GA. In another recent paper [7], we attempted to use the synchronous master-slave Parallel Genetic Algorithm (PGA) to make the algorithm as time efficient as possible.

The WSC environment becomes more dynamic due to the increase in the number of Web services whose properties change frequently. In the literature, Several GAs approaches [8 - 11] have been proposed to solve QoS-aware WSC are suffer from dynamicity. Therefore, in this paper we extend our previously published hybrid approach by integrating K-means clustering with our hybrid approach. We call it PGAQK. K-means clustering is used to cluster the candidate Web services based on fitness function which describes in section IV. K-means clustering is applied in mutation operator and used to sustain the concrete Web service with another one in the case of failure Web services to ensure fault tolerance. As far as we know, K-means clustering with Q-learning based PGA has never been applied in adaptation of QoS-aware WSC. K-means clustering has been applied in QoS-aware WSC before, but not with PGA to the best of our knowledge. PGAQK contributions include a considerable increase fitness value and applying the K-means clustering to prune the number of candidate Web services in the search space to find the best Web services to adapt WSC in the dynamic changes. We implemented PGAQK over the .NET Framework using C# programming language. We carried experiments for evaluation which results indicate the effectiveness of PGAQK compared to PGA and Q-learning approaches.

The rest of this paper is organized as follows. Section 2 presents essential background about K-means Clustering. Section 3 gives a brief review of the related literature. Section 4 presents PGAQK. Section 5 gives our evaluation of PGAQK. Finally, Section 6 gives the conclusion and future work.

2. K-means clustering

Cluster analysis is a data mining technique that aims to group a multivariate dataset into clusters (groups) [12]. The aim of this clustering is to have the objects in the same cluster similar to each other, and different from the objects in the other clusters [12]. Cluster analysis techniques have been classified into hierarchical and partitioning clustering [12]. Hierarchical clustering is a set of nested clusters that are organized as a tree [12]. Partitioning clustering splits data objects into non-overlapping clusters such that each data object is in exactly one cluster [12]. The K-means clustering algorithm is a well-known partitioning clustering algorithm because of its ease of implementation, and low memory consumption, as well as its being a statistical and quite scalable method [12].

K-means clustering algorithm is as follows. Suppose we are given an x dataset in the real d-dimensional space Rd, and integer M clusters. K-means clustering aims to partition the x dataset into M clusters. Each cluster is represented by the centroid (center point). In K-means clustering the number of clusters must be specified first. The K-means clustering steps are [12]:

1. Select random K points as the initial centroid.
2. Calculate the weighted Euclid distance

between each data set and the centroid. The dataset is assigned to the cluster according to the minimum distance *d* formulated as in Eq. (1) [12]:

$$d = \sqrt{(a - \bar{a})^2} \quad (1)$$

where a denotes the value of the data set and $\bar{a}$ denotes the value of the centroid.

3. When all the datasets are assigned to their respective clusters, recomputed the average of the centroid and get a new centroid of the *K* clusters.
4. Repeat steps 2 and 3 until the centroids don't change.

3. Related work

As the number of Web services on the Internet has increased exponentially, the QoS attributes have to be taken into consideration to satisfy the users' requirements. In the literature, a number of QoS-aware WSC approaches based on GA have been proposed, these approaches aim to select the best Web service that optimize the global QoS requirement. In this paper, the focus is on QoS-aware WSC approaches based on GA only. Hence other optimization approaches such as fruit fly optimization [13] are not taken into consideration in this section. Also approaches that cover WSC in geo-distributed cloud environment [14] are not taken into consideration in this section too.

Canfora et al. [15], Su et al. [16] and Xiangbing et al. [11] proposed approaches based on GA to solve QoS-aware WSC problem. Canfora et al. [15] show that LIP outperforms GA in small scale problem. Su et al. [16] present an initial- population policy as a relation matrix coding scheme of chromosomes. Xiangbing et al. [11] propose an optimal approach to the QoS based Web Service Modeling Ontology (WSMO) WSC model by using GA. All of these approaches are randomly generated initiate population therefore these approaches are time-consuming to find near optimal solution. Therefore, Ma et al. [17] and Liu et al. [18] proposed a hybrid approaches to solve this limitation. Ma et al. [17] propose Genetic Programming (GP) based random greedy algorithm. The random greedy algorithm is used to generate locally optimal chromosome in the initial population of GP instead of having it generated randomly. The objective of using random greedy algorithm is to improve the globally optimal by GP. The random greedy algorithm also is used to perform mutation operations during GP. The conflict between QoS criteria is not taken into consideration. Liu et al. [18] combine Ant Colony Optimization (ACO) and GA. ACO is used to generate an initial population to enhance the quality of a chromosome in the initial population and to improve the convergence process. To improve the efficiency of this algorithm, the search space is reduced by using Orthogonal list. The Orthogonal list contains the sorted QoS sparse matrix and the available services which are encoded into a binary string. This approach is not suitable for large scale service-oriented system.

Recently, adaptive QoS-aware WSC in a dynamic environment have received more attention. The approaches addressing how to adapt WSC for changes that occur in FRs are out of the scope of this paper. Our recent publication [19] gives a review about self-adaptive WSC approaches for changes that occur in FRs. Imed et al. [20] address the change of QoS by proposing three variability operators; *replicate*, *delete*, and *replace*. *Replicate* and *delete* operators are applied to add and delete service instances in WSC, while *replace* is applied to change some improper Web services. These operators are applied to reconfigure WSC when the Service Level Agreement (SLA) contract is violated. However, the three variability operators are not sufficient to adapt WSC to the change in QoS properties.

Wang et al. [21] present a self-adaptive WSC framework based on Reinforcement Learning (RL). Markov Decision Process (MDP) is used in this approach to model WSC. Business processing workflows and candidate services are integrated into a single WSC. At runtime, depending on the status of the services and the environment, the concrete workflows and concrete Web services are selected. An optimal policy is found by using Q-learning. Then Wang et al. [22] extend their Q-learning RL framework by describing a Multi-Agent Reinforcement Learning (MARL) mechanism to enable adaptive WSC. In addition, Team Markov Games (TMG) is used to model multi-agent WSC. Wang et al. [23] extend their MARL based approach by integrating the multi State-Action-Reward-State-Action (SARSA) learning algorithm to find the optimal solution. In a later publication, Wang et al. [24] also use MARL like in Wang et al. [22]. In this approach the agents are work together to find the optimal WSC. This approach presents a sharing strategy to share information with an agent that makes an agent use the policies found by the others. Although this approach is more efficiency than traditional Q-learning in the large scale WSC, it also does not take into consideration the problem of failure of Web services. However, Wang et al. [21, 22, 23, 24] do not take into consideration the case of

global QoS constraint, fault tolerant technologies and the problem of unavailability Web services.

Xia et al. [25] present an algorithm to adapt WSC by using a clustering technique to cluster a huge number of Web services into a number of groups according to QoS properties. Business Process Execution Language (BPEL) tree is used to model WSC. The Ordering Points To Identify the Clustering Structure (OPTICS) algorithm is used to cluster Web services. Their mechanism was the prototype and there was not any implementation.

Therefore, most methods suffer from reduced search space that contains only Web services that are equivalent to changing Web services. Therefore, we proposed PGAQK by applying the K-means clustering technique in the mutation operator and used it to sustain the concrete Web service with another one if the Web services fail to ensure fault tolerance. The objective is to adjust WSC to fit a changing environment where the properties of the composite Web services continue changing and achieve a better fitness value in the new environment.

4. The hybrid approach using PGA, Q-Learning and k-means clustering (PGAQK)

Currently, most of the existing WSC PGA-based approaches start with a low-quality population at the initial stage. To overcome this matter, we previously proposed a Q-learning based PGA approach [7]. It used Q-learning combined with PGA to generate the initial population of our PGA-based WSC algorithm. This resulted in a considerable improvement in the optimal solution as demonstrated in [7]. We focus on the sequential WSC model. In this section we first give steps of our previous research since PGAQK is an extension of it, before embarking on describing the integration of K-means clustering to handle adaptiveness.

4.1 Q-learning based PGA

In this section, we give an explanation that underpins our previous Q-learning based PGA approach for WSC as presented in [7]; Fig. 1 gives the flowchart of it. The main focus of this approach is using Q-learning to generate the initial population of PGA instead of having it generated randomly. This approach consists of two general steps. The first step is creating the Q-table using Q-learning. The second step is applying the synchronous master-slave PGA on the initial population created from the Q-table of the previous step. Each step is described in more details below in the following subsections.

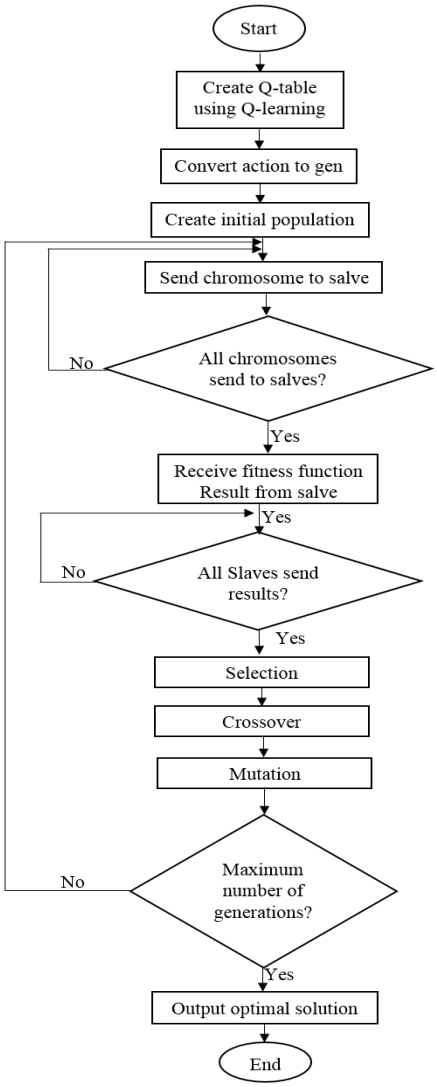

Figure. 1 The flowchart of Q-learning based PGA approach

4.1.1 Create Q-table Using Q-learning

In this step the Q-table is defined based on action-state space. Q-learning is applied to create the Q-table. Each state in the Q-table has represented an AWS and the actual Action selected; the one candidate Web service.

The ε −greedy strategy is used to select the action. With ε −greedy, the random action is selected with a fixed probability, $0 \leq \varepsilon \leq 1$. If the random number is less than ε − greedy, the agent selects a random candidate Web service. Otherwise, the agent selects the candidate Web service that has maximum reward as shown in Eq. (2) referred by [24]. Ɛ stands for random value.

$$\pi(s) = \begin{cases} \text{random action from A(s)} & \text{if } Ɛ < \epsilon \\ \text{argMax}_a\ Q(s,a) & \text{otherwise} \end{cases} \quad (2)$$

The reward of QoS attributes is given by Eq. (2). Eq. (3) referred by [11] assumes that each service has *n* service quality criteria.

$$\begin{cases} \sum_{x=1}^{n} \frac{(Q_x^{max} - Q_x^{neg})}{(Q_x^{max} - Q_x^{min})} * w_x \\ \sum_{x=1}^{n} \frac{(Q_x^{pos} - Q_x^{min})}{(Q_x^{max} - Q_x^{min})} * w_x \\ \sum_{x=1}^{n} 1 * w_x \end{cases} \begin{vmatrix} \text{if} Q_x^{max} - Q_x^{min} \neq 0 \\ \text{if} Q_x^{max} - Q_x^{min} \neq 0 \\ \text{if} Q_x^{max} - Q_x^{min} = 0 \end{vmatrix} \quad (3)$$

The matrix Q is built, in which each row Q_x^{neg} corresponds to a negative Web service and Q_x^{pos} corresponds to a positive Web service, while each column corresponds to a quality dimension. Q_x^{max} and $Q_x^{\min}$ are the maximal and minimal value of a quality criterion in the matrix Q respectively. $w_x \epsilon$ [0; 1] represents the weight criterion x of which the values are provided by the users based on their own preferences. After the reward function of current state is calculated, the Q-value of this state is updated based on Eq. (4) referred by [24]. In Eq. (4) s stands for the state of the agent,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \propto \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (4)$$

and a means the action is executed by agent. After executing the action a in the state s, the r represent the resulting reward in the new state s'. $Q(s, a)$ represents Q-value of state s and action a. $\propto$ is a learning rate and its range is from 0 to 1 ($0<\propto<1$). γ is the discount rate and it takes this range ($0\leq\propto\leq1$). This formula is repeated for each action in the policy.

4.1.2 Applying PGA

The first step of finding the optimal solution using PGA is to define a way to encode the Q-table into chromosome. Each policy (path) in Q-table are converted to a population of chromosomes. A chromosome consists of a number of genes. The chromosome represents a single policy in the Q-table and the gene represents concrete Web service in the state of Q-table policy. After creating the initial population, the synchronous master-slave PGA is used to find the optimal solution. The process of synchronous master-slave PGA is as follows:

1. In each generation, the initial population stored in the master and the fitness function is evaluated in the slave. The fitness function used in this paper is Eq. (3).
2. The master waits to receive the fitness values for all the population from the slave.
3. Sorting the population according the fitness function score.
4. Roulette wheel selection is used to select the parent chromosomes from the population.
5. Depending on the crossover rate, apply *two-point crossover* into the parent. Two crossover points in the parent chromosomes are selected randomly and the gens between them are exchanged.
6. Depending on the mutation rate, apply *mutation* into the new offspring from *crossover*.
7. Repeat steps 4, 5 and 6 until a new population has been generated.
8. After the generation is finished, update the Q-values with the best solution and determine the optimal solution. The optimal solution is the chromosome with the highest fitness function.

4.2 Integrating k-means clustering

K-means clustering is clustered candidate Web services into clusters based on fitness function. The input of K-means clustering algorithm is the number of clusters and the fitness function of all candidate Web services. When Web service properties, such as availability of Web services, changes in QoS (e.g. cost, availability, response time, etc.), are often changed, this demands that the cluster of candidate Web services and means will be changed and WSC must be adapted to provide the most suitable composite services and fulfill the users' requirements that emerge. Therefor K-means clustering is applied in *mutation* operator to replace the mutation gen with another one from the cluster better than the mutation gen cluster. In the case of unavailability of Web service, K-means clustering is used to sustain with another concrete Web service in the same cluster to keep the chromosome value. The explanation that integrating K-means clustering in *mutation* operator and use it in Web service substitution is described in more details below in the following subsections.

4.2.1 K-means clustering mutation operator

The *mutation* operator is used primarily as a mechanism for maintaining genetic diversity in the population. PGAs and GAs use simple mutation, which changes a random gen position of a randomly selected Web services. A random selection leads to reduction in fitness function while the objective is to maximum the fitness function. Therefore, K-means clustering algorithm applies in *mutation* operator. The main objective of applying it in *mutation* operator is to cluster candidate Web services and prune the clusters which contain fitness function less than mutation gen. The K-means clustering algorithm

combines with Q-learning based PGA in *mutation* operator to decrease the search space and find the best Web services to environment changes that may occur at runtime. The result of Q-learning based PGA is finding the chromosome which satisfies the optimal global QoS constraints. In Algorithm 1 the pseudocode for applying the K-means clustering algorithm in *mutation* is given, which are as follows:

1. The random gene is selected.
2. The candidate Web services which are related to the gene are retrieved.
3. These candidate Web services are clustered by using the K-means clustering.
4. The K-means centroids are arranged.
5. The K-means centroids which are higher than the centroid of the gene are retrieved.
6. The roulette wheel selection is applied to selected K-means centroids.
7. The candidate Web service from the selected centroid which has the maximum fitness function is selected.

4.2.2 Web service substitution stage

Web service substitution is the ability of replacing a Web service by another one with the same functionality and QoS properties to ensure fault tolerance. When Web service is unavailability, the failing gene which contains this Web service will be changed. Therefore, the concrete Web service for the failing gene needs to sustain with another concrete Web service to keep the chromosome value. Therefore, we use K-means clustering to cluster the fitness values of the candidate Web services for the failing gene then all candidate Web services in the same cluster of this failing gen are retrieved because this cluster contains the nearest Web service to the failing gen. The process of the K-means clustering algorithm in the case of unavailable Web service is illustrated:

1. The fitness function and cluster number of the failing gene are retrieved.
2. The candidate Web services in the same cluster of the failing gen are retrieved.
3. Calculate the substitution Coefficient to quantify the similarity degree between the failure Web service and candidate Web services in the same cluster by using Eq. (5):

$$Sc(Q_i, Q_{j)} = \frac{F_{Q_j}}{F_{Q_J} - F_{Q_i}} \quad (5)$$

Where $Sc(Q_i, Q_{j)}$ represents substitution Coefficient between unavailable Web service j and candidate Web service i. Q_i represents candidate Web service. Q_J represents the unavailable Web service. F_{Q_j} represents the fitness value of unavailable Web service and F_{Q_i} represent the fitness value of candidate Web service i.

4. The candidate Web service with the higher substitution Coefficient will be selected.

Algorithm 1: The pseudocode for the applying K-means Clustering algorithm into *mutation* operator

Input: the new offspring from *crossover*, cluster number
Output: optimal Web service
Repeat for each new offspring from *crossover*.
  Select random value.
  If random value less than mutate rate.
    Select random gene.
    Retrieve all candidate Web services which are related to this random gene.
    Apply the K-means Cluster algorithm to these candidate Web services.
    Arrange the K-means centroids.
    Retrieve the K-means centroids which are higher than the centroid of the gene.
    K-means centroid is selected by applying roulette wheel selection into selected K-means centroids.
    Select the candidate Web service from the selected centroid which has the maximum fitness function.
  End if
Until all new offspring from *crossover* have been covered.

5. Experiments and analysis

In the Web services environment, the QoS attributes of a service are not always static due to the changes that occur in the network and software systems [24]. Therefore, WSC should be able to automatically adapt to these changes. When QoS attributes of a service change, the original optimal solution might not be optimal anymore. In this experiment, we consider three QoS attributes for services; namely, cost, response time, and reliability. The weight of service cost, response time and service reliability are 0.5, 0.2 and 0.3. In order to evaluate PGAQK approach, the algorithm is implemented in visual Studio 2015 using C# language. The QoS of the candidate Web services are saved in SQL server 2014. The tests are conduct using a Dell Laptop with configuration an Intel Core i7 at 2.50 GHz, 8 GB RAM, running Microsoft Windows 10. The number of cluster is 5. The crossover probability parameter is set as 0.8; the mutation parameter is set as 0.2. The

learning rate parameter $\propto$ is set as 0.5; the discount rate parameter γ is set as 0.8. The of $\varepsilon-$greedy exploration strategy value is set as 0.85. Then, we execute the algorithm and record the optimal solution with different numbers of initial population size.

In our experiment, the QoS values of each attribute used are based on the QWS dataset [26, 27]. The data records in QWS Dataset were gathered from all kinds of public sources on the Web, such as UDDI registries, search engines, and service portals. Based on this, we have also expanded the dataset to satisfy random assignment for each AWS with one certain QoS data record. In addition, the QoS values of each AWS in our experiment keep changing periodically with a certain frequency to simulate the dynamic experimental environment. After executing the algorithm, the optimal solution with a different number of initial population size are recorded.

5.1 Fitness value results

The aim of the first experiment is to examine the efficiency of PGAQK to enable WSC to obtain the optimal solution. The optimal solution is the total reward (fitness function in PGA) for each AWS in the WSC path. The number of episode is the same as the number of initial population. We compare the PGAQK with PGA and Q-learning approaches utilized by Wang et al. in [21]. In this experiment, each AWS has 364 candidate Web services. Fig. 2 illustrates the experiment results for 500 chromosomes in the initial population. The number of AWS is fixed to 40 Web services. The number of generations are 10, 30, 50, 70, 90, 110, 130 and 150. In PGAQK and PGA, the optimal solution is the chromosome with the highest cumulative reward while in Q-learning the optimal solution is the policy with the maximum reward.

As shown in Fig 2, the optimal solution of PGAQK outperforms PGA technique and Q-learning in all generations number. The best optimal solution in the case of using PGAQK found in generation 150 was 31while the best optimal solution in the case of using PGA and Q-learning approaches found in generation 130 and 90 was 18.7 and 21.5 respectively. Fig. 3 illustrates the experiment results for 1000, 1500, 2000, 2500, 3000 and 3500 chromosomes in the initial population. The number of generation in this experiment is 10 and the number of AWS is 40 Web services. The optimal solution in the case of using PGAQK, PGA and Q-learning approaches in the initial population of 1000 chromosomes was 25.8, 18.1 and 22.3 respectively then the optimal solution changed by increasing or decreasing in another initial

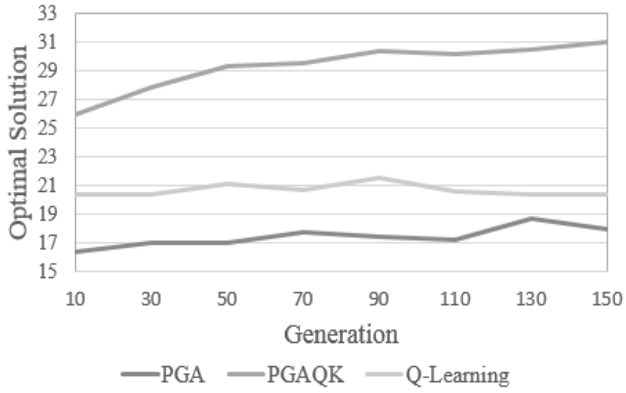

Figure. 2 Optimal solutions for 500 chromosomes in the initial population

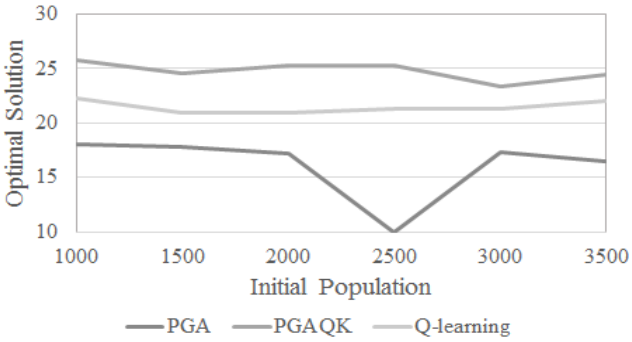

Figure. 3 Optimal solutions from 1000 to 3500 chromosomes in the initial population

population but it didn't reach the optimal solution that achieved in the initial population of 1000 chromosomes. As could be seen from Fig. 3, PGAQK can obtain the best cumulative reward value with a lower number of chromosomes in the initial population.

Fig. 4 illustrates the experiment's results from 10 to 100 AWS for 500 chromosomes in the initial population. The number of generation in this experiment is 10. In 20 and 30 AWS, the optimal solution in the case of using PGAQK and PGA and Q-learning approaches are the same otherwise the optimal solution in the case of using PGAQK is better than the optimal solution in the case of using PGA and Q-learning approaches. As could be seen from Fig. 2, Fig. 3 and Fig. 4, the optimal solution in the case of using PGAQK is generally better than the optimal solution in the case of using PGA and Q-learning approaches.

Fig. 5 illustrates the experiment results for 5000 chromosomes in the initial population. The number of AWS is varied from 100 to 500 Web services. The number of generation is 10. In this experiment, we mainly focus on verifying the impact of the number of AWS on the efficiency of the algorithm to show the scalability. As could be seen from Fig. 5, the

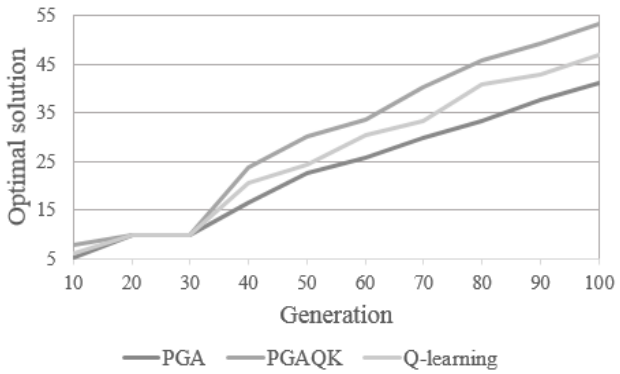

Figure. 4 Optimal solutions from 10 to 100 AWS

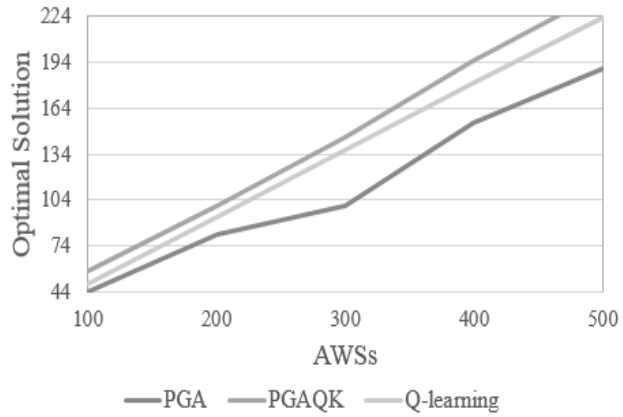

Figure. 5 Optimal solutions for different number of AWS

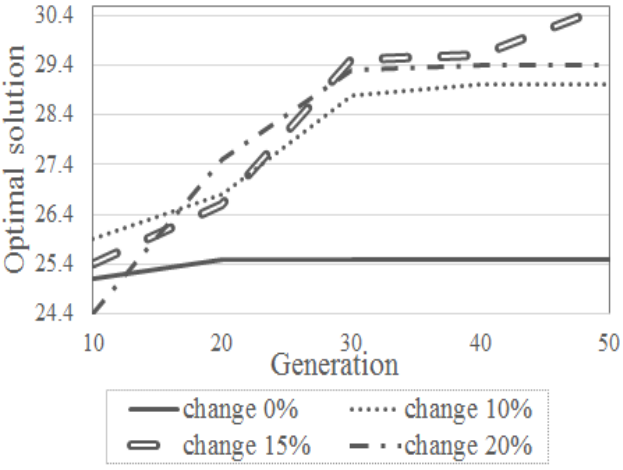

Figure. 6 Adaption testing

growth trend about cumulative reward value obtained in the different number of AWS through PGAQK. Overall, PGAQK can obtain higher optimal solution than the PGA and Q-learning approaches in the case of the same number of AWS, and this advantage becomes more apparent with the increase in numbers of AWS.

5.2 Adaptability to changes

In the second stage of our evaluation, we studied

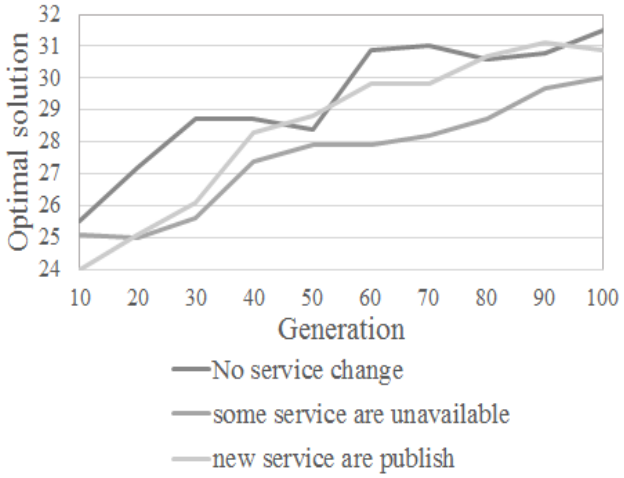

Figure. 7 Impact of changes in web services

how well PGAQK adapt to the changes of the environment. When the service QoS attributes changes, the original optimal solution may not be optimal anymore. To simulate the dynamic environment, we randomly change the QoS values of candidate Web services during the generations. In this experiment, the number of AWS is 40 Web services, the number of candidate Web service is fixed as 1000 Web services. The number of initial population is 1000 chromosomes and the generation number is fixed as 50 generations. In generation 19, we varied the 10%,15% and 20 % of Web service's QoS attributes respectively. Fig. 6 shows the growth of the optimal during the generations. We can see that the best optimal solution in the case of no variation in QoS was 25.5 in the generation 20. After the re-setting on QoS attribute values, the optimal solution which received in generation 20 is the optimal solution for the changed circumstance. Fortunately, the K-means clustering is applying in *mutation* operator to obtain the best Web services in the search space into the population, so it will still find a new optimal solution eventually. For example, in 15% variation of QoS, the optimal solution in generation 20 was 26.6. Then the optimal solution increased until it reached the best optimal solution for the changed circumstance in generation 50. As a result, this experiment indicates that PGAQK can address the changes emerging from the environment dynamically.

In the second set of experiments, we simulated the environment changes by adding and deleting 2% Web services per 20 generations. In this experiment, the number of AWS is 40 Web services, the number of candidate Web service is fixed as 1000 Web services. The number of initial population is 500 chromosomes and the generation number is fixed as 100 generations. As shown in Fig. 7, when some services become unavailable and some new services

are published, the optimal solution is different, because those unavailable Web services make the search area become smaller and new published Web service make the search area become bigger.

6. Conclusion and future work

In this paper, we presented a new approach called PGAQK for QoS-aware WSC in a dynamic environment. We integrated K-means clustering with PGA based Q-learning to decrease the search space to find the most suitable Web services and fulfill emerging users' requirements. PGA based Q-learning is used to find the optimal solution. Q-learning is used to enhance the effectiveness of PGA by generating the initial population. The *mutation* operator is used primarily as a mechanism for maintaining genetic diversity in the population. The K-means clustering is applied in the *mutation* operator to cluster candidate Web services and prune the clusters which contain fitness functions less than the mutation gene. When change occurs in a service's QoS attributes, the optimal solution obtained before a change is not the optimal solution in the new environment. Therefore, we used the K-means clustering in the *mutation* operator to obtain the best Web services in the search space in the population. In the case of unavailable or failing Web services, it used to sustain the Web service by decreasing the search space to the Web services in the same cluster of the failing or unavailable Web services.

Although PGAQK currently considers only three QoS attributes, it can be easily extended to include more QoS attributes by incorporating them into reward and fitness functions without making any changes to any other part of PGAQK. We reported in this paper the results of a series of comparable experiments that we performed to test the efficiency, scalability and adaptability of PGAQK. For efficiency and scalability experiments, PGAQK was compared with traditional PGA and Q-learning approaches. The experiments' results show that the effectiveness of PGAQK in terms of the fitness value. When the number of population is 500, the number of AWSs are 40 and the generations number is 150, PGAQK achieves a fitness value of 31, which is better than PGA and Q-learning approaches where their fitness values were 18.7 and 20.4 respectively. We contend that PGAQK can dynamically optimize WSC to fit a changing environment, where the properties of the composite Web services continue changing. In our future work we intend to integrate PGAQK with MAPE-K (Monitor-Analyze- Plan-Execute over a shared Knowledge) component and a formal model to compose a formal QoS-aware WSC model. The aim is to have this model take inter-service dependencies and conflicts into consideration.

References

[1] F.-S. Hsieh and J.-B. Lin, "A Self-adaptation Scheme for Workflow Management in Multi-agent Systems", *Journal of Intelligent Manufacturing*, Vol. 27, No. 1, p. 131–148, 2016.

[2] N. Ide and J. Pustejovsky, "What Does Interoperability Mean, Anyway? Toward an Operational Definition of Interoperability for Language Technology", In: *Proc. of the 2nd International Conf. on Global Interoperability for Language Resources*, 2010.

[3] N. H. Rostami, E. Kheirkhah, and M. Jalali, "An Optimized Semantic Web Service Composition Method Based on Clustering and Ant Colony Algorithm", *International Journal of Web & Semantic Technology*, Vol. 5, No. 1, pp. 1-8, January 2014.

[4] B. Rohallah, M. Ramdane, and S. Zaidi, "Agents and Owl-s based Semantic Web Service Discovery with User Preference Support", *International Journal of Web & Semantic Technology*, Vol. 4, No. 2, pp. 57-75, April 2013.

[5] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational Intelligence based QoS-aware Web Service Composition: A Systematic Literature Review", *IEEE Transactions on Services Computing*, Vol. 10, No. 3, pp. 475-492, 2015.

[6] D. H. Elsayed, E. S. Nasr, and A. E. M. El Ghazali, "A New Hybrid Approach Using Genetic Algorithm and Q-learning for QoS-aware Web Service", In: *Proc of the International Conf. on Advanced Intelligent Systems and Informatics*, pp. 537- 546, 2017.

[7] D. H. Elsayed, E. S. Nasr, and A. E. M. El Ghazaly, "Integration of Parallel Genetic Algorithm and Q-learning for QoS-aware Web Service Composition", In: *Proc. of the 12th International Conf. on Computer Engineering and Systems*, pp. 221-226, 2017.

[8] D. Wang, Y. Yang, and Z. Mi, "A Genetic-based Approach to Web Service Composition in Geo-Distributed Cloud Environment", *Computers and Electrical Engineering*, Vol. 43, pp. 129-141, 2014.

[9] R. Iordache and F. Moldoveanu, "A Genetic Algorithm for Automated Service Binding", *Procedia Engineering*, Vol. 69, pp. 1162-1171, 2014.

[10] A. E. Yilmaz and P. Karagoz, "Improved Genetic Algorithm Based Approach for QoS

Aware Web Service Composition", In: *Proc. of the 2014 IEEE International Conf. on Web Services*, 2014.

[11] Z. Xiangbing, M. Hongjiang, and M. Fang, "An Optimal Approach to the QoS-based WSMO Web Service Composition using Genetic Algorithm," In: *Proc. of the International Conf. on Service-Oriented Computing*, 2012.

[12] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson, p. 494, 2006.

[13] B. B. Savarala and P. R. Chella, "An Improved Fruit Fly Optimization Algorithm for QoS Aware Cloud Service Composition", *International Journal of Intelligent Engineering and Systems*, Vol. 10, No. 5, pp. 105-114, 2017.

[14] S. B. Bhushan and P. R. CH, "A QoS Aware Cloud Service Composition Algorithm for Geo-Distributed Multi Cloud Domain", *International Journal of Intelligent Engineering and Systems*, Vol. 9, No. 4, pp. 147-156, 2016.

[15] G. Canfora, P. D. Massimiliano, R. Esposito, and M. L. Villani, "An Approach for QoS aware Service Composition based on Genetic Algorithms", In: *Proc. of the 7th Annual Conf. on Genetic and Evolutionary Computation*, 2005.

[16] S. Su, C. Zhang, and J. Chen, "An Improved Genetic Algorithm for Web Services Selection", In: *Proc. of the International Conf. on Distributed Applications and Interoperable Systems*, 2007.

[17] H. Ma, A. Wang, and M. Zhang, "A Hybrid Approach Using Genetic Programming and Greedy Search for QoS-Aware Web Service Composition", In: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XVIII*, pp. 180-205, 2005.

[18] H. Liu, F. Zhong, B. Ouyang, and J. Wu, "An Approach for QoS-aware Web Service Composition based on Improved Genetic Algorithm", In: *Proc. of the 2010 International Conf. on Web Information Systems and Mining*, 2010.

[19] D. H. Elsayed, E. S. Nasr, A. E. M. El Ghazali, and M. H. Gheith, "Appraisal and Analysis of Various Self-Adaptive Web Service Composition Approaches", In: *Ramachandran M., Mahmood Z. (eds) Requirements Engineering for Service and Cloud Computing*, Springer International Publishing, pp. 229–246, 2017.

[20] A. Imed, M. Graiet, S. Boubaker, and N. B. Hadj-Alouane, "A Formal Approach for Verifying QoS Variability in Web services composition using EVENT-B", In: *Proc. of the 2015 IEEE International Conf. on Web Services*, 2015.

[21] H. Wang, X. Zhou, X. Zhou, W. Liu, and W. Li, "Adaptive and Dynamic Service Composition Using Q-Learning", In: *Proc. of the 22nd International Conf. on Tools with Artificial Intelligence*, 2010.

[22] H. Wang, X. Chen, Q. Wu, Q. Yu, Z. Zheng, and A. Bouguettaya, "Adaptive and Dynamic Service Composition via Multi-agent Reinforcement Learning", In: *Proc. of the 2014 IEEE International Conf. on Web Services*, 2014.

[23] H. Wang, X. Chen, Q. Wu, Q. Yu, and Z. Zheng Athman, "Integrating On-policy Reinforcement Learning with Multi-agent Techniques for Adaptive Service Composition", In: *Proc. of the 12th International Conf. Service Oriented Computing*, 2014.

[24] H. Wang, X. Wang, X. Hu, X. Zhang, and M. Gu, "A Multi-Agent Reinforcement Learning Approach to Dynamic Service Composition", *Journal of Information Sciences*, Vol. 363, p. 96–119, 2016.

[25] Y. Xia, P. Chen, L. Bao, M. Wang, and J. Yang, "A QoS-Aware Web Service Selection Algorithm Based On Clustering", In: *Proc. of the 2011 IEEE International Conf. on Web Services*, 2011.

[26] E. Al-Masri and Q. H. Mahmoud, "QoS-based Discovery and Ranking of Web Services", In: *Proc. of the 16th International Conf. on Computer Communications and Networks*, 2007.

[27] A.-M. Eyhab and M. H. Qusay, "Discovering the best web service", In: *Proc. of the 16th International Conf. on World Wide Web (WWW)*, 2007.