A Self-Healing Model for QoS-aware Web Service Composition

Doaa Elsayed[1], Eman Nasr[3], Alaa El Ghazali[4], and Mervat Gheith[2]
[1]Department of Information Systems and Technology, Cairo University, Egypt
[2]Department of Computer Science, Cairo University, Egypt
[3]Independent Researcher, Egypt
[4]Department of Computer and Information Systems, Sadat Academy for Management Sciences, Egypt

Abstract: *In the Web Service Composition (WSC) domain, Web Services (WSs) execute in a highly dynamic environment, as a result, the Quality of Service (QoS) of a WS is constantly evolving, and this requires tracking of the global optimization overtime to satisfy the users' requirements. In order to make a WSC adapt to such QoS changes of WSs, we propose a self-healing model for WSC. Self-healing is the automatic discovery, and healing of the failure of a composite WS by itself due to QoS changes without interruption in the WSC and any human intervention. To the best of our knowledge, almost all the existing self-healing models in this domain substitute the faulty WS with an equivalent one without paying attention to the WS selection processes to achieve global optimization. They focus only on the WS substitution strategy. In this paper, we propose a self-healing model where we use our hybrid approach to find the optimal WSC by using Parallel Genetic Algorithm based on Q-learning, which we integrate with K-means clustering (PGAQK). The components of this model are organized according to IBM's Monitor, Analyse, Plan, Execute, and Knowledge (MAPE-K) reference model. The PGAQK approach considers as a module in the Execute component. WS substitution strategy has also been applied in this model that substitutes the faulty WS with another equivalent one from a list of candidate WSs by using the K-means clustering technique. K-means clustering is used to prune the WSs in the search space to find the best WSs for the environment changes. We implemented this model over the NET Framework using C# programming language. A series of comparable experiments showed that the proposed model outperforms improved GA to achieve global optimization. Our proposed model also can dynamically substitute the faulty WSs with other equivalent ones in a time-efficient manner.*

Keywords: *Web service composition, self-healing, quality of service, user requirements, K-means clustering.*

Received June 29, 2018; accepted January 28, 2020
https://doi.org/10.34028/iajit/17/6/2

1. Introduction

Service Oriented Architecture (SOA) is an architectural style that uses loosely coupled, distributed, and adaptive software applications [19]. Web Service (WS) is the underpinning of SOA. With the popularity of WSs, Web Service Composition (WSC) that combines atomic WSs from different service providers together to satisfy users' requirements has become one of the challenges in the SOA [7]. The selection of the best WS from a lot of candidate WSs according to Quality of Service (QoS) is called QoS-aware WSC, which has become one of the current hot topics [7]. As far as we know, there are currently two main approaches proposed in the literature for QoS-aware WSC, namely local and global optimization [12]. The local optimization approach involves selecting the best WS from its set of candidate WSs for each Abstract Web Services (AWS) individually. Although this approach is optimized locally and efficient with low time complexity of O(m), where m is the number of concrete WSs for each AWS [26], the global QoS constraints of WSC may not be satisfactory. Therefore, there are many algorithms to solve the global optimization problem. The global optimization approach considers QoS constraints and preferences as a whole, for example, when the whole response time is constrained. Genetic Algorithms (GAs) are approaches commonly used for solving a global optimization problem in QoS-aware WSC. The performance of GAs depends on the initial population. Therefore, we attempted to address this restriction in a recent paper [8] by utilizing Q-learning to generate the initial population to improve the performance of GAs. The nature of GAs is time-consuming, so in another recent paper [9], we aimed to make the algorithm as time-efficient as possible by using the synchronous master-slave Parallel Genetic Algorithm (PGA).

WSs execute autonomously in a highly dynamic environment. Accordingly, WSC is characterized by continuous evolution, as service providers may change the QoS properties of WSs, existing WSs may be unavailable, new WSs may become available, and variations in the infrastructure may affect the performance of the existing WSs [6]. Therefore, WSC should be equipped with self-healing ability, which means that a composite WS is automatically discovered and healed itself according to the QoS

changes without stopping the WSC process, and without any human intervention [18, 19]. The lack of an effective self-healing ability during WSC at runtime may violate the users' requirements. In the literature, several approaches have been proposed to solve the QoS-aware WSC problem in a dynamic environment, e.g., [1, 21, 22, 23, 24, 25]. These approaches do not take into consideration the case of some WSs' failures that can lead to a whole disability of this WSC workflow. Furthermore, these approaches cannot support self-healing execution. In a recent paper [8], we proposed a new hybrid approach for dynamic optimization of WSC by using PGA based on Q-learning that we integrated with K-means clustering. We called this hybrid approach PGAQK. In this current paper, we propose a self-healing model that integrates our PGAQK approach with IBM's autonomic Monitor, Analyse, Plan, Execute, and Knowledge (MAPE-K) reference model to fulfill the QoS goals of WSC when changes occur in the operational environment. The main focus of this model is to keep WSC up to date of QoS changes that occur at design and run times and satisfy global optimization.

The rest of this paper is organized as follows. Section 2 gives a brief review of the related literature. Section 3 presents our proposed new self-healing model. Section 4 gives the evaluation of our model. Finally, section 5 gives the conclusion and future work.

2. Related Work

In this section, we review the related work of self-healing WSC available in the literature. Angarita *et al*. [4] analysed the impact on the execution time of WSC using different Fault Tolerant (FT) techniques in different execution scenarios. This study focuses on backward, forward, and replication recovery techniques. The selection of these recovery techniques is dynamic based on context information, the effect of QoS attributes, and the execution state of composite WSs when the failure occurs. Gupta and Bhanodia [11] proposed a subset replacement mechanism for fault tolerance in the WSC process. The replacement policy of their mechanism replaces the subset of WSs that contains the failed WS with another equivalent subset. During the execution, their mechanism identifies subsets of the failed WSs, and the subsets of the equivalent ones. Then, the equivalent subsets are ranked, and the best subset among them is selected. Their mechanism, as published, was the prototype, and there was no implementation. Rajendran *et al*. [18] proposed a Dynamic Self-Healing (DSH) method that uses the substitution healing method to heal a WS in a dynamic environment. This method focuses on the response time programming and the availability of the WSs. Karray *et al*. [14] proposed an approach to enhance the reliability of WSs based on aspect-oriented programming and case-based reasoning. Their approach implementation is in weather WSs. Li *et al*. [15] also proposed a framework based on case-based reasoning. Their framework is composed of a business process part, an extractor part, and a case-based reasoning part. The business process achieves the users' requirements. The extractor part extracts the Functional Requirement (FR), Non-Functional Requirement (NFR), and fault information. The faulty information is classified into a case with a solution and case without a solution. Their framework is not efficient in complex WSC structures due to the case-base becoming very large, and it doesn't have techniques to maintain it.

Boumhamdi and Jarir [5] presented the state of the art of various types of failures and recovery methods. They also proposed architecture for detecting and dynamically recovering failure of the Business Process Execution Language (BPEL) process based on the FR of a user's preferences. This architecture is composed of five components. These components are request analyser, discovered component, constraints manager, selection manager, and orchestrator. When the composite WS failure occurs, the BPEL adapter substitutes the failure WS with alternative WS. Jayashree and Anand [13] proposed a runtime fault detection process for static, semi-dynamic, and dynamic for WSC. BPEL is used to implement and test static and semi-dynamic WSs; Web Ontology Language for Services (OWL-S) is used to implement and test dynamic WSs. Once faulty WSs are detected, the meaningful error message displays to the user to understand where and why the fault has occurred. The user takes the necessary decision that resubmits the service request or substitutes the failure WSs with equivalent ones. Their approach is tested by use a sample WS application. Subramanian *et al*. [20] extend BPEL with a self-healing policy to enhancement it. This policy monitors the BPEL activities due to unexpected failures in the WS process, diagnoses the cause of failure, and suggests a solution to this failure. Diagnoses part policy uses the database to store failure information. If failure information is not in the database, human intervention is required. Poonguzhali *et al*. [17] presented a self-healing approach which focuses on WS unavailability. Their approach is composed of BPEL monitor, diagnoser, and path substituter. Their approach alternates the routing path, which contains the failure WS to the nearby router to the composer. Their mechanism, as published, is a prototype, and there was no implementation.

The substitution process in most self-healing models suffers from time consumption because these models have to search all WSs in the search space to find the corresponding WSs to the faulty WS. These models also do not consider WS selection processes to achieve global optimization. Therefore, we integrate our PGAQK approach with IBM'S MAPE-K to adjust WSC to appropriate a variable environment where the

properties of the composite WSs continue being variable and also achieve better global optimization.

3. Our Proposed Self-Healing Model

Figure 1 gives an illustration of our proposed self-healing model. The inputs to this model are sequential AWS workflow and the candidate WSs that can use to construct the concrete WSC. The components of this model organize according to the MAPE-K loop. The explanation that underpins the components of this model describes in more detail below in the following subsections.

3.1. The Execute Component

The Execute component involves the PGAQK module and the adaptation manager module. PGAQK is a global optimization approach, which comprises PGA, Q-learning, and K-means clustering, as mentioned before. These techniques are integrated to generate a global optimal WS selection plan that results in concrete WSC, which can ensure the users' QoS requirements. The general process of WS selection using PGAQK illustrates in Figure 2. The PGAQK is composed of creating the Q-table using Q-learning, encoding the Q-table to the initial population, and applying the master-slave PGA on the population. Our recent publications [6, 7] describe how to create Q-table and encode it to the initial population. In the master-slave PGA, the single initial population stores in the master, and the fitness function is distributed and evaluated in the slaves. After the master receives the fitness value for all populations from the slave, the roulette wheel selection is applied to select the parent chromosomes. Then the crossover is applied to the parent chromosomes depended on the crossover rate. Finally, the K-means clustering mutation is applied to the new offspring from the crossover. PGAQK in detail is described in our recent publication [14].

Figure 1. The flowchart of our proposed self-healing model.

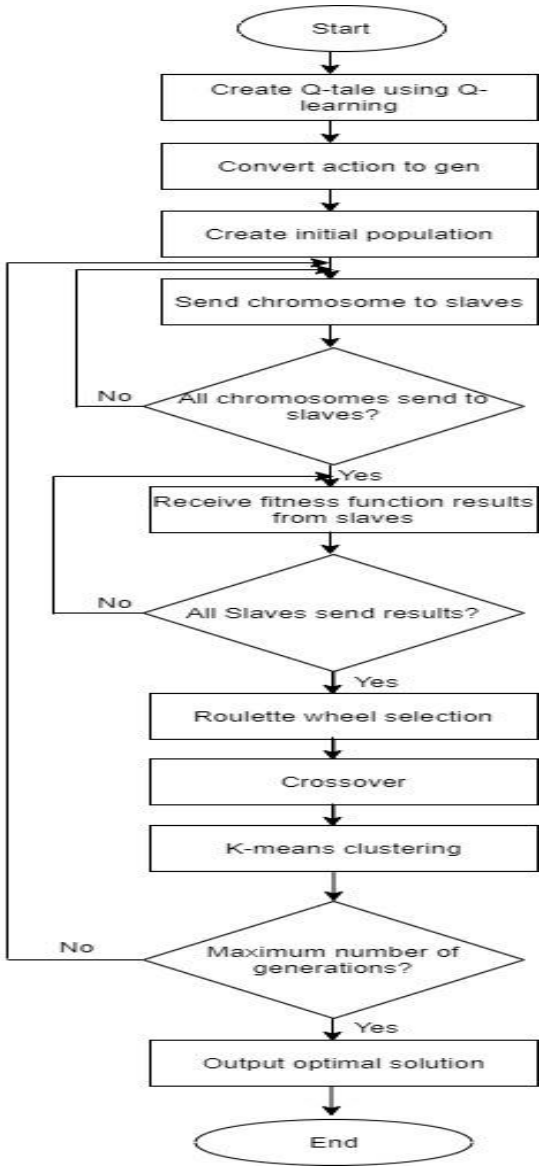

Figure 2. The flowchart of PGAQK architecture.

The adaptation manager receives notification from the adaptation policy module in the Plan component when a fault detects at the system. Once the fault is detected at design time, the PGAQK process pauses temporarily until the adaption action finish. Then, the PGAQK process resumes. At runtime, the fault can occur in atomic WS during the execution of the WSC. This atomic fault can lead to all WSC failure. It is mandatory to detect this failure, and the adaptation manager works on it based on a specified adaptation policy.

3.2. The Plan Component

The role of the Plan component is to identify the appropriate adaptation policy for the current situation. Adaptation policies correspond to self-healing features that enable configuration to respond to WS failure such as unavailability of WS and poor response. In our case, the adaptation action mainly focuses on substitution strategy in the case of unavailability of service, service failure, and variation in the QoS attribute of WS. The substitution strategy identifies the WS that replaces the faulty WS. The K-means clustering is applied to cluster the candidate WSs to identify the cluster that has the nearly QoS equivalent to faulty WS. In algorithm 1, the pseudocode for the substitution strategy is illustrated.

Algorithm 1: The pseudocode for the substitution strategy

Retrieve the fitness function score of the faulty WS.
Retrieve the cluster number of the faulty WS.
Repeat for each candidate WSs in the cluster number of the failure WS.
Calculate the substitution Coefficient between the faulty WS, and candidate WSs by using (1):

$$Sc(Q_i, Q_{j)} = \frac{F_{Q_j}}{F_{Q_J} - F_{Q_i}} \quad (1)$$

Where $Sc(Q_i, Q_{j)}$ *represents the substitution Coefficient between faulty* $WSj_{,\ and}$ *candidate* WSi. Q_i *represents candidate WS.* Q_J *represents the faulty WS.* F_{Q_j} *represents the fitness function of faulty WS, and* F_{Q_i} *represents the WS of candidate* WSi.
Until all candidate WSs in the cluster have been covered.
Select the candidate WS with the highest substitution Coefficient.
End applying K-means clustering algorithm in the case of faulty WS.

In the case of emerging new WS to candidate WSs, the fitness value of this WS calculate by using Multiple Criteria Decision Making (MCDM) to aggregate all QoS attributes to the same scale. For the negative QoS attributes such as response time and cost, values are scale according to the Equation (2). For the positive QoS attributes such as availability, and reliability the values are scale according to the Equation (3). After that, this WS assigns to the corresponding WS cluster based on fitness value.

$$\begin{Bmatrix} \sum_{j=1}^{r} \frac{(Q_{ij} - L_{ij})}{(U_{ij} - L_{ij})} * w_j \Big| if U_{ij} - L_{ij} \neq 0 \\ \sum_{j=1}^{r} 1 * w_j \Big| if U_{ij} - L_{ij} = 0 \end{Bmatrix} \quad (2)$$

$$\begin{Bmatrix} \sum_{j=1}^{r} \frac{(U_{ij} - Q_{ij})}{(U_{ij} - L_{ij})} * w_j \Big| if U_{ij} - L_{ij} \neq 0 \\ \sum_{j=1}^{r} 1 * w_j \Big| if U_{ij} - L_{ij} = 0 \end{Bmatrix} \quad (3)$$

Equations (2) and (3) assume that each service has r QoS criteria. A Q is a QoS matrix of WSs, in which each row Qj corresponds to a WS i, while each column corresponds to a QoS dimension. U_{ij} and L_{ij} are the upper, and lower values of a QoS criterion in the matrix Q respectively. w_j represents the weight criterion j. These values are provided by the users into the range of 0 to1 based on their preferences.

In the case of variation in QoS attributes of WS, the WS removes from the WS cluster and then assigns it into the corresponding WS cluster based on new fitness value.

3.3. The Monitor, and Analysis Component

The Monitor and Analysis component is responsible for capturing changes in the environment. This component

involves the QoS monitor module and the concrete WS monitor module. The QoS monitor module is responsible for capturing changes in the candidate WSs. Whenever this module detects a change that occurs in candidate WSs, PGAQK in the Execute component suspends until the adaptation policy executes.

The concrete WS monitor is responsible for capturing changes in the concrete WS at the runtime. If they are relevant, the trigger sends to the Plan component to determine the adaptation policy. Currently, tracked changes may include:

1. The availability of WS may evolve as new WS emerge, and old WS replaces or existing WS temporarily disconnected due to maintenance.
2. QoS attribute value such as price is variation.

3.4. The Knowledge Component

All the knowledge that other components need is stored in this component. It considers as a storage layer. After constructing the AWS workflow, the candidate WSs attributes for each AWSs extract from WSDL files and store in candidate WSs storage in the Knowledge component. These candidate WSs use in the PGAQK module in the Execute component to construct concrete WSs workflow. The initial population storage creates by using the Q-learning technique in the PGAQK module, and it changes in each PGA generation. After applying the K-means clustering in the candidate WSs, the K-means centroids and candidate WSs clusters create and store in K-means centroids storage and candidate WSs storage. These storages use in the PGAQK module at K-means clustering mutation operator and in the adaptation policy module in the Plan component.

4. Aluations

In this section, we report the experiments that we carried out for evaluating our new self-healing model. We used the Quality of Web Service (QWS) dataset [2, 3] in our experiments, which include 364 data records. The data records were collected from all kinds of public sources on the Web, such as Universal Description, Discovery, and Integration (UDDI) registries, search engines, and service portals. In our experiments, we consider three QoS attributes for services; namely, cost, response time, and reliability. The weight of the cost, response time, and reliability services are 0.5, 0.2, and 0.3 respectively. In order to evaluate our proposed model, the algorithm was implemented over .NET Framework platform 4.7 using C# programming language. The SQL server 2014 was used to save the Knowledge layer component. The experimental results were conducted on a Dell Laptop with an Intel Core i7 at 2.50 GHz, 8 GB Random Access Memory (RAM), and running Microsoft Windows 10.

We held two experiments for evaluation. The first one compares our new model with the improved GA approach utilized by Liue *et al*. [16] to compare the resulting fitness value results. The second experiment compares our model with the PGAQK approach proposed in the recent paper [14] to compare time consumption to substitute the faulty WSs with another equivalent one.

4.1. Fitness Value Results

The first experiment aims to examine the efficiency of this model in the Execute component to enable WSC to achieve the optimal solution. In a recent paper [14], we showed the effectiveness of the PGAQK module in the Execute component compared to the traditional PGA and Q-learning approaches in terms of fitness values. In this paper, the fitness of the PGAQK module is evaluated by comparing it with the improved GA approach utilized by Liue *et al*. [16] which combines Ant Colony Optimization (ACO), and GA. In [16], GA is improved by utilizing ACO to generate the initial population. We use the parameters given in Table 1 below to test data. We implemented the algorithm and recorded the best fitness value.

Table 1. Parameters setting.

Parameter	Value
Crossover probability	0.8
Mutation	0.2
$\propto$	0.5
γ	0.8
ε **−greedy**	0.85
Number of ants	70
Alpha, and Beta	1
Iteration	72

The fitness function in PGAQK is the total reward for each AWS in a WSC path. The number of the initial population is the same as the episodes' number. Figures 3, 4, and 5 illustrate the experiment's results for 500, 1000, and 2000 chromosomes in the initial population respectively. The number of generations is 10, 20, 50, and 100. The WSC consists of 10 AWSs. The optimal solution is the chromosome with the best cumulative reward. As could be seen from Figures 3, 4, and 5 the optimal solution in the case of using PGAQK is generally better than the optimal solution in the case of using Improved GA. to As could be seen from these figures, the best optimal solution was 8.14. In the case of using PGAQK, the best optimal solution found in 500, 1000, and 2000 chromosomes in the initial population while this optimal solution didn't find in the case of using improved GA.

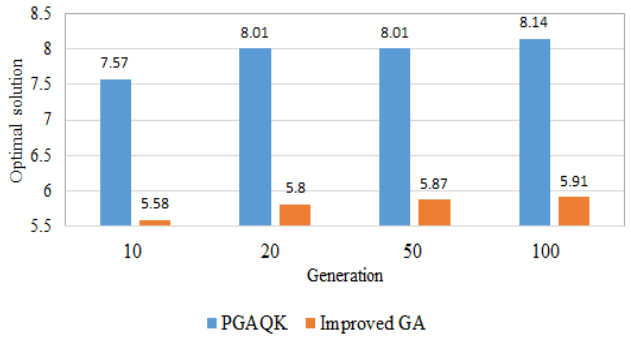

Figure 3. The optimal solutions for 500 chromosomes in the population.

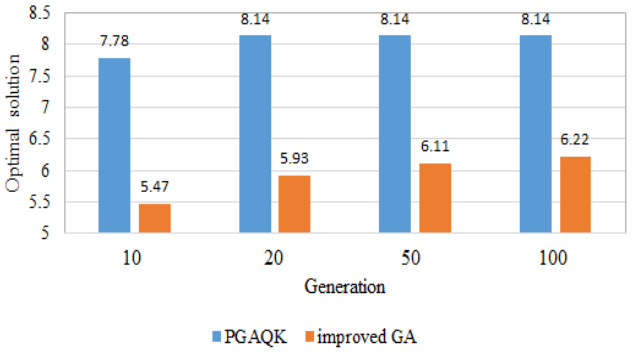

Figure 4. The optimal solutions for 1000 chromosomes in the population.

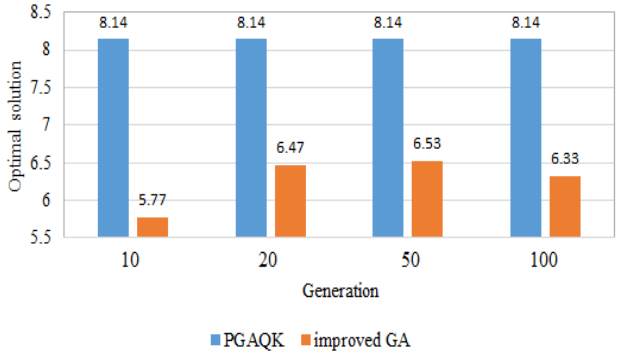

Figure 5. The optimal solutions for 2000 chromosomes in the population.

Figure 6 illustrates the experiment results for 500, 1000, 2000, 2500, 3000, 3500, and 4000 chromosomes in the initial population. The number of generations is 10. In the case of using PGAQK, the optimal solution in the initial population of 500 chromosomes was 7.86. Then the curve progressively increased until it reached the best optimal solution of 8.14 in the cases of 1500, and 2000 chromosomes in the initial population. After that, the curve went down in the case of 2500 chromosomes in the initial population. In the cases of 3000, 3500, and 4000 chromosomes in the initial population, the best optimal solution was obtained again. In using the improved GA approach, the best optimal solution was 5.9, which was obtained in the case of 3000 chromosomes in the initial population. Hence it could be deduced that PGAQK is better than the improved GA approach.

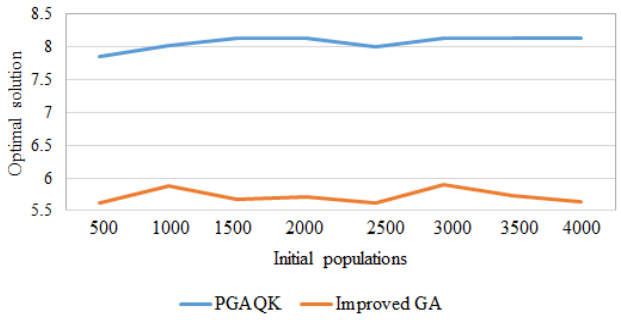

Figure 6. The optimal solution from 500 to 4000 chromosomes in the initial population.

4.2. Execution Time

For the cost of our proposed model, we measured it by using the execution time for substituting the 2% of faulty WSs with another equivalent one. We chose to measure the execution time in seconds. Figure 7 illustrates the computation time for the same 100 chromosomes for the proposed model, and the PGAQK approach proposed in the recent paper [14]. The number of AWS is varied from 10 to 100 AWSs. As could be seen from Figure 5, the time consumption to substitute the 2% of faulty WSs with another equivalent one by using the PGAQK approach is much more effective than our proposed model when the AWS less than 30 AWSs. Otherwise, the time consumption of our proposed model is much more effective than the PGAQK approach.

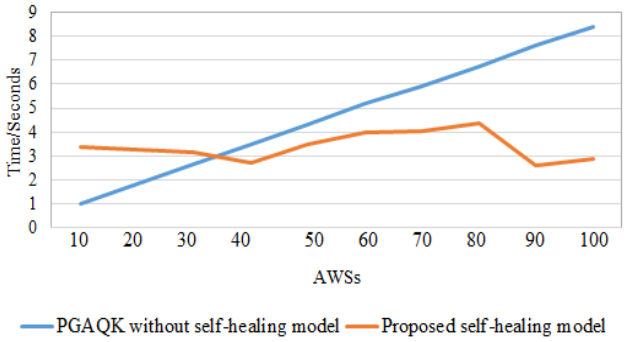

Figure 7. The total time to substitute the faulty WSs with another equivalent one when AWS varied from 10 to 100.

In the second experiment, the number of the initial population is varied from 100 to 1000 chromosomes; the number of AWSs is 10 WSs. As could be seen from Figure 8, the time consumption to substitute the 2% of faulty WSs with another equivalent one by using the PGAQK is much more effective than our proposed model when the chromosome less than 300 chromosomes. Otherwise, the time consumption of our proposed model is much more effective than the PGAQK approach. Hence, we can deduce that the time consumption of our proposed model is much more effective when the number of chromosomes more than 200 chromosomes or in the case of AWS is more than 30 AWSs.

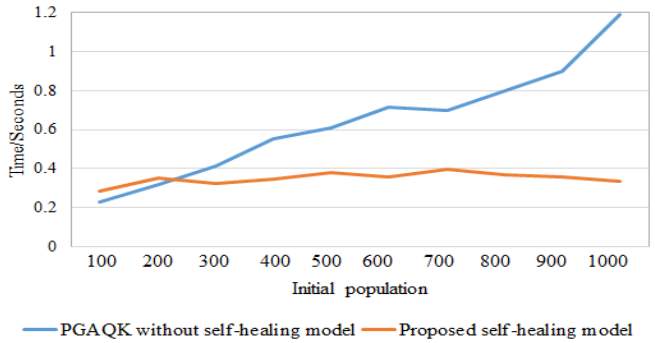

Figure 8. The total time to substitute the faulty WSs with another equivalent one when the initial population varied from 100 to 1000.

5. Conclusions, and Future Work

This paper presents a new self-healing model for QoS-aware WSC, which focuses on adjusting WSC to appropriate a variable environment where the properties of the composite WSs continue being variable and also attend to the WS selection processes to achieve global optimization. Therefore, this model integrates the PGAQK approach with IBM's MAPE-K reference model. The PGAQK approach is a module in the Execute component used to achieve global optimization and fulfill emerging users' requirements. The PGAQK approach in this model was improved by clustering and preserving the candidate WSs in the Knowledge component. When a faulty WS occurs, the most appropriate cluster is determined from the Knowledge component, and then the equivalent WS to the faulty WS is determined from that specific cluster. Experiment results show that the PGAQK in the Execute component is more effective than the improved GA approach in terms of fitness value. In terms of computation time, our model improved WS substitution time, especially when the number of atomic WSs is large. In our future work, we intend to improve the time of our proposed model to achieve global optimization. We intend to investigate other ways of adapting compositions including changes in the structure of the composition's workflow, e.g., replacing one WS with a set of WSs, or vice-versa.

References

[1] Abbassi I., Graiet M., Boubaker S., Mourad K., and Hadj-Alouane N., "A Formal Approach for Verifying QoS Variability in Web Services Composition using EVENT-B," *in Proceedings of the IEEE International Conference on Web Services*, New York, pp. 519-526, 2015.

[2] Al-Masri E. and Mahmoud Q., "Discovering the Best Web Service," *in Proceedings of the 16th International Conference on World Wide Web*, New York, pp. 1257-1258, 2007.

[3] Al-Masri E. and Mahmoud Q., "QoS-based Discovery, and Ranking of Web Services," *in Proceedings of 16th International Conference on Computer Communications and Networks*, Honolulu, pp. 529-534, 2007.

[4] Angarita R., Cardinale Y., and Rukoz M., "Reliable Composite Web Services Execution: Towards a Dynamic Recovery Decision," *Electronic Notes in Theoretical Computer Science*, vol. 302, pp. 5-28, 2014.

[5] Boumhamdi K. and Jarir Z., "An Approach to Support Monitoring, and Recovery of BPEL Processes at Runtime," *International Journal of Computer Applications*, vol. 43, no. 2, pp. 34-41, 2012.

[6] Cardellini V., Casalicchio E., Grassi V., Iannucc S., Presti F., and Mirandola R., "MOSES: a Platform for Experimenting with QoS-driven Self-adaptation Policies for Service Oriented Systems," *in Proceedings of Software Engineering for Self-Adaptive Systems 3. Assurances*, Germany, pp. 409-433, 2013.

[7] Dai Y., Yang L., and Zhang B., "QoS-Driven Self-Healing Web Service Composition Based on Performance Prediction," *Journal of Computer Science, and Technology*, vol. 24, no. 2, pp. 250-261, 2009.

[8] Elsayed D., Nasr E., Ghazali A., and Gheith M., "A New Hybrid Approach using Genetic Algorithm, and Q-learning for QoS-aware Web Service Composition," *in Proceedings of the 3rd International Conference on Advanced Intelligent Systems, and Informatics*, Cairo, pp. 537-546 2017.

[9] Elsayed D., Nasr E., Ghazali A., and Gheith M., "Integration of Parallel Genetic Algorithm, and Q-learning for QoS-aware Web Service Composition," *in Proceedings of the 12th International Conference on Computer Engineering, and Systems*, Cairo, pp. 221-226, 2017.

[10] Elsayed D., Nasr E., Ghazali A., and Gheith M., "PGAQK: An Adaptive QoS-aware Web Service Composition Approach," *International Journal of Intelligent Engineering, and Systems*, vol. 11, no. 4, pp. 231-240, 2018.

[11] Gupta S. and Bhanodia P., "A Fault Tolerant Mechanism for Composition of Web Services using Subset Replacement," *International Journal of Advanced Research in Computer, and Communication Engineering*, vol. 2, no. 8, pp. 3080-3085, 2013.

[12] Jatoth C., Gangadharan G., and Buyya R., "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.

[13] Jayashree K. and Anand S., "Run Time Fault Detection System for Web Service Composition,

and Execution," *Smart Computing Review*, vol. 5, no. 5, pp. 469-482, 2015.

[14] Karray M., Ghedira C., and Maamar Z., "Towards a Self-Healing Approach to Sustain Web Services Reliability," *in Proceedings of the Workshops of International Conference on Advanced Information Networking, and Applications*, Singapore, pp. 267-272, 2011.

[15] Li G., Liao L., Song D., Wang J., Sun F., and Liang G., "A Self-healing Framework for QoS-Aware Web Service Composition via Case-Based Reasoning," *in Proceedings of Asia-Pacific Web Conference*, Sydney, pp. 654-661, 2013.

[16] Liu H., Zhong F., Ouyang B., and Wu J., "An Approach for QoS-aware Web Service Composition based on Improved Genetic Algorithm," *in Proceedings of the International Conference on Web Information Systems, and Mining*, Sanya, pp. 123-128, 2010.

[17] Poonguzhali S., JerlinRubini L., and Divya S., "A Self-Healing Approach for Service Unavailability in Dynamic Web Service Composition," *The International Journal of Computer Science, and Information Technologies*, vol. 5, no. 3, pp. 4381-4383, 2014.

[18] Rajendran V., Chua F., and Chan G., "Self-Healing in Dynamic Web Service Composition," *in Proceedings of the 5th International Conference on Future Internet of Things, and Cloud*, Prague, pp. 206-211, 2017.

[19] Rastegari Y. and Shams F., "A Dynamic Architecture for Runtime Adaptation of Service-based Applications," *The International Arab Journal of Information Technology*, vol. 16, no. 3, pp. 397-406, 2019.

[20] Subramanian S., Thiran P., Narendra N., Mostefaoui G., and Maamar Z., "On the Enhancement of BPEL Engines for Self-Healing Composite Web Services," *in Proceedings of the International Symposium on Applications, and the Internet*, Turku, pp. 33-39, 2008.

[21] Wang H., Chen X., Wu Q., Yu Q., Zheng Z., and Bouguettaya A., "Integrating On-policy Reinforcement Learning with Multi-agent Techniques for Adaptive Service Composition," *in Proceedings of the 12th International Conference Service Oriented Computing*, Paris, pp. 154-168, 2014.

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

[23] Wang H., Wu Q., Chen X., Yu Q., Zheng Z., and Bougu A., "Adaptive, and Dynamic Service Composition via Multi-agent Reinforcement Learning," *in Proceedings of the IEEE International Conference on Web Services*, Anchorage, pp. 447-454, 2014.

[24] Wang H., Zhou X., Zhou X., Liu W., and Li W., "Adaptive, and Dynamic Service Composition Using Q-Learning," *in Proceedings of the 22nd International Conference on Tools with Artificial Intelligence*, Arras, pp. 145-152, 2010.

[25] Xia Y., Chen P., Bao L., Wang M., and Yang J., "A QoS-Aware Web Service Selection Algorithm Based On Clustering," *in Proceedings of the IEEE International Conference on Web Services*, Washington, pp. 428-435, 2011.

[26] Zou G., Lu Q., Chen Y., Huang R., Xu Y., and Xiang Y., "QoS-Aware Dynamic Composition of Web Services using Numerical Temporal Planning," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 18-31, 2014.

Doaa Elsayed received her Ph.D. degree in Information Systems, and Technology from Cairo University, Egypt, in 2019. She is currently a Lecturer at the Computer, and Information Systems Department in Sadat Academy for Management Sciences. Her research interests include web services, requirements engineering, optimization algorithms, and data mining.

Eman Nasr is a Ph.D. degree holder in Computer Science. She is currently an Independent Scholar. Her research interests include software engineering, requirements engineering, and embedded software systems.

Alaa El Ghazali is currently a Professor of Computer and Information Systems at Sadat Academy for Management Sciences, Egypt. His research interests include software engineering economics, decision support systems, business intelligence, and mobile computing.

Mervat Gheith is currently an Associate Professor in the Department of Computer Science in the Faculty of Graduate Studies for Statistical Research at Cairo University, Egypt.