Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 890 – 903

doi:10.3906/elk-1905-177

Dynamic software rejuvenation in web services: a whale optimization algorithm-based approach

Kimia REZAEI KALANTARI[1], **Ali EBRAHIMNEJAD**[2,*], **Homayun MOTAMENI**[3]
[1]Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran
[2]Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran
[3]Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

Received: 30.05.2019 • **Accepted/Published Online:** 25.12.2019 • **Final Version:** 28.03.2020

Abstract: In this paper, we suggest a method for determining the restarting time for web services to increase availability, known as rejuvenation. We consider different parameters such as number of users, maximum service request number, response time, and throughput of a web service to determine its restarting time. Software rejuvenation is an effective technique to counteract software aging in continuously running applications such as web service-based systems. In these systems, web services are allocated based on the needs of the receivers and facilities of servers. One of the challenges while assigning web services is selecting the appropriate server to reduce faults. Since the selection of a server among candidates while maintaining the optimal quality of service is an NP-hard problem, metaheuristics seem to be suitable. In this paper, we propose dynamic software rejuvenation as a proactive fault-tolerance technique based on the whale optimization algorithm. The threshold for the rejuvenation of each of the web services is considered and training is done based on the features of the service providers as well as the needs of the receivers. The whale optimization algorithm with the criterion of movement radius is utilized for flexibility of web service provider selection. Here, we detect and rejuvenate systems that required rejuvenation before the occurrence of a fault. The simulation results reveal that our strategy can decrease the failure rate by an average of 30 percent in comparison with state-of-the-art strategies and improve the system availability in web services.

Key words: Software aging, software rejuvenation, web service, whale optimization algorithm

1. Introduction

Web services are used for developing client server applications for communication and provide flexible solutions for integration of software under web environments [1]. Several web services provide similar functionality with different nonfunctional properties, specified as quality of service (QoS) attributes. Reliability, availability, and deadline guarantees are the most important metrics to measure the QoS of long-running and real-time applications. Producing a service composition with an optimal QoS value that satisfies the customer requirements is a complex and essential task [2]. For example, web services without QoS technology in e-business could negatively impact business operations such as no guarantee of profits and longer outages and downtimes. Several studies [3, 4] have reported that one of the causes of performance degradation and unplanned software outages is the software aging phenomenon. Software aging observed in software as long-running software systems suffer from abnormal states, performance degradation, and even hang or failure. The reasons for software aging are consumption of operating system resources, data corruption, and rounding error accumulation, which could be accompanied by memory leaks, data fragments, unreleased file locks, and database connections. Software

*Correspondence: aemarzoun@gmail.com

rejuvenation is one of the most commonly used approaches to handle issues caused by software aging. This process removes the accumulated errors and frees operating system resources [5].

However, rejuvenation of software does not solve the root cause of software aging. As a result, software aging will continue from the system's startup; therefore, software rejuvenation should be run periodically at predetermined or scheduled times to maintain software system firmness. Additionally, the software system may experience system performance downtime and even system faults during rejuvenation procedures. Meanwhile, it may bear a number of costs (such as the cost of losing business due to inaccessibility at the time of rejuvenation). Nevertheless, the costs incurred due to software rejuvenation are less than those incurred due to system faults. The most important problem is adopting an affordable rejuvenation policy to ensure system reliability, reduce maintenance and system inactivity costs, and improve system availability.

In previous works, in order to improve software reliability, software defect prediction was applied to the process of software maintenance to identify potential bugs [6]. Li et al. [7] suggested a hybrid method called the software process risk optimization method that integrates risk management into the software process model. This technique resembles a heavyweight approach, which might not be suitable for web service environments. Ning et al. [8] studied multigranularity software rejuvenation policy and assumed inspection intervals to follow exponential distribution. The limitation of these works is based on the consideration of the constant time and distribution that decrease the performance of systems.

Optimal scheduling of services in web service environments has been proven to be an NP-complete problem, hence the need for the application of heuristic method seems to be necessary [9]. In addition, intelligent optimization approaches or metaheuristic algorithms such as the genetic algorithm (GA), particle swarm optimization (PSO), and whale optimization algorithm (WOA) have addressed the QoS issue in recent years [10]. In [11], a hybrid optimization algorithm for enhancing the scheduling process in web services was suggested that combined the GA and ant colony optimization (ACO) algorithms. However, the difficulty of improving the operation steps and parameter selection are the shortcomings of such methods.

In this paper, we investigate a new software rejuvenation policy in web service-based systems. We analyze the system model to derive the optimum policy by evaluating the reliability and extending the threshold for rejuvenation in servers during web services allocation. We rejuvenate servers with reliability values that are less than the threshold extension. However, determining the explicit threshold extension or a specified value might decrease the system efficiency. Thus, we use a whale optimization algorithm in the selection of the suitable web servers for the requested web services.

The rest of this paper is organized as follows. Section 2 reviews previous related works. A review of the whale optimization algorithm (WOA) is presented in Section 3. In Section 4, the proposed dynamic software rejuvenation based on the WOA is presented. Experimental results are described in Section 5. Finally, conclusions and future work are discussed in Section 6.

2. Literature review

With the rapid growth of Internet technology, web application reliability is the main concern. The important issues to make them reliable are the performance and availability of the web application or server. Software aging can be defined as a growing weakening of the internal state of the software. Aging will affect the performance of a device and ultimately cause the application to fail. Therefore, such characteristics are used in the determination of web application candidates for software aging. Software rejuvenation is one of the dynamic techniques of fault tolerance and is used to solve the software aging problem.

Reliability is one of the essential requirements for cloud computing systems, which must provide services with high availability, high fault tolerance, and dynamic deployment capabilities. Sun et al. [12] examined the effectiveness of software in cloud computing systems. Based on that study, software rejuvenation has a great effect on the performance of communications among the dimensions of cloud computing as well as the reliability of their components. Since the software and hardware of the cloud may have limited reliability, the use of multiple clusters may be required to achieve the expected level of dependability. Dantas et al. [13] presented availability models for private cloud architectures based on the Eucalyptus platform. The needs for reliability, accessibility, and high performance have increased in modern software applications that are meeting fast-growing needs and providing endless services. Araujo et al. [14] proposed a software framework that is mainly used as a service for private cloud computing. Their research studied the effectiveness of software within the framework of Eucalyptus in terms of system workloads and high demands for remote storage and virtual models. They also proposed an approach that offers time series analysis for rejuvenation scheduling to reduce the fault time by predicting the appropriate time for rejuvenation. Guo et al. [15] used discrete web services to measure software aging failures and a software rejuvenation strategy based on the view that discrete web services provide a loose connection feature to the server. In their research, they used several linear regression methods to calculate the aging of individual web services.

Kulkarni [16] proposed a method of shifting applications across a virtual machine (VM) by using time-based rejuvenation. In order to overcome internal software rejuvenation, Torquato et al. [17] presented availability models based on stochastic petri nets to evaluate two VM live migration approaches. The method was based on redundancy schemes called warm-standby and cold-standby. As decision-trees are among the most popular machine learning algorithms, Ghayathri et al. [18] presented a web services quality prediction model based on the decision-tree approach. They applied classification of web services based on the decision-tree approach. The modified algorithm of the C5 classifier was used to do the classification based on QoS parameters. Umesh et al. [19] proposed an adaptive genetic algorithm (A-GA) for rejuvenation. They used their method to perform live migration to avoid downtime. Mooij et al. [20] carried out cost-effective industrial software rejuvenation with domain-specific models. They showed that semiautomatic software rejuvenation in industrial operations is feasible and cost-effective. They used specific domain models that use abstract implementation details and applied a practical combination of manual and automated techniques. Sumathi and Raju [21] used rejuvenation that should be finely planned and scheduled. They carried out the rejuvenation process for the Apache web service provider with a feedforward neural network.

Designing a dynamic routing algorithm to satisfy QoS requirements is a challenging task. Additionally, multiconstrained QoS routing aims to optimize multiple QoS metrics. Providing the required network resources is an admittedly complex problem [22]. The fault tolerance criterion for web services is considered an important research topic in choosing a web service. The existing fault tolerance web service models have limitations. For example, various important sources of fault tolerance do not integrate or do not focus on various attributes of service quality such as performance, accessibility, and so forth to satisfy user requirements.

Liu et al. [23] unified a fault tolerance management module with standard service architecture by converting a web service network into a small global network based on the fault tolerance relationships of the service entities; they introduced a fault tolerance evaluation model with a reasonable correction logic. Baek et al. [24] proposed an approach to improve the fault traceability in web applications by utilizing software revision information and presented a behavior model to reduce the developer's effort. Gupta et al. [25] presented a QoS-enabled approach to tolerate faults in the composition process. In their approach, a trust-based web service

filtering mechanism is used, and whenever a fault is detected in a process, a decision for dynamic recovery is made. That is based on the optimum QoS ranking of the web services. The limitation lies in detecting only an incorrect value and the service unavailability fault in this algorithm.

In general, most of the methods previously proposed to optimize software rejuvenation in various software systems have shortcomings. In this study, it has been tried to overcome most of these shortcomings. Some of the most important of these shortcomings are summarized below:

1. Most of the proposed methods for optimizing software rejuvenation in software systems are single-dimensional. That is, only one aspect of software rejuvenation has been considered. For example, Meng et al. [26] presented nonsequential inspection intervals to optimize software rejuvenation. However, they did not consider other effective dimensions in software rejuvenation. In their study, Machida and Miyoshi [27] considered an optimal stopping problem for software rejuvenation. However, they did not investigate sequences with different time intervals.
2. Most of the proposed methods for software rejuvenation have static management in software systems. That is, they have considered a static structure for software rejuvenation. However, they have not considered different methods for software rejuvenation at the time of running the software under different circumstances [26, 28].
3. One of the main problems in optimizing software rejuvenation is the dependence of optimum time for rejuvenation on the distribution of software system failures. Often, the proposed methods for optimizing the software rejuvenation have included a particular probability distribution for failure events. For example, one of the Poisson, Gaussian, or Weibull distributions has been used. On the contrary, the effects of different distributions of failure events in optimizing software rejuvenation were not evaluated [29].
4. Many software rejuvenation methods stop the software system at the time of software rejuvenation; users' requests are not answered at the time of rejuvenation. This policy of the implementation mechanism simplifies the procedure but reduces the system's availability [30].
5. Most presented software rejuvenation methods include a single copy for the software. Therefore, the fault tolerance is low and the system availability is reduced during the rejuvenation [30, 31].
6. One of the problems in optimizing software rejuvenation is to select a suitable server for web application allocation that can reduce fail frequency. The thresholds for the rejuvenation of web services and servers were not considered to date.

3. Whale optimization algorithm

Metaheuristic algorithms impersonate biological or physical phenomena to handle complex real-world optimization problems [32]. Metaheuristic algorithms are widely used to find the global optimum of real engineering problems. The disadvantages of existing numerical methods concerning factors such as simplicity, efficiency, and accuracy encourage researchers to rely on metaheuristic algorithms based on methods that are inspired by nature or different branches of science to solve engineering optimization problems [33, 34]. Some of the most prominent nature-inspired metaheuristic algorithms are particle swarm optimization (PSO) [35], the butterfly optimization algorithm (ALO) [36], genetic algorithms (GAs) [37], and the whale optimization algorithm (WOA) [38]. It is proven that this algorithm has better or comparable performance as compared to some existing algorithmic techniques [38]. WOA is a simulation of the hunting behavior of humpback whales. Humpback whales

encircle their prey and update their position towards the best search agent (whale) with increasing numbers of iterations. This problem is made up of encircling prey, spiral bubble-net attacks, and searches for prey of the WOA. Mathematical models of these subproblems are described below.

3.1. Encircling prey

This behavior is showed by the following equations:

$$\overrightarrow{D} = |C \cdot \overrightarrow{X^*}(t) - \overrightarrow{X}(t)|, \tag{1}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X^*}(t) \cdot \overrightarrow{A} \cdot \overrightarrow{D}, \tag{2}$$

where $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, t indicates the current iteration, X^* is the position vector of the best solution obtained so far, $\overrightarrow{X}$ is the position vector, $\|$ is the absolute value, and . is an inner product.
The vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated as in Eq.(3) and (4):

$$\overrightarrow{A} = 2\overrightarrow{a} \cdot \overrightarrow{r} - \overrightarrow{a}, \tag{3}$$

$$\overrightarrow{C} = 2 \cdot \overrightarrow{r}, \tag{4}$$

where $\overrightarrow{r}$ is a random number in [0,1] and $\overrightarrow{a}$ is linearly decreased from 2 to 0 over the course of the iteration.

3.2. Bubble-net attacking method

This method is a hybrid form of two approaches that can be mathematically modeled as follows:

3.2.1. Shrinking encircling mechanism

This behavior is achieved by decreasing the value of a from 2 to 0 in Eq. (3) over the course of iterations. The new position of a search agent can be defined anywhere between the original position of the agent and the position of the current best agent by setting a random value for vector $\overrightarrow{A}$ in $[-1, 1]$.

3.2.2. Spiral updating position

The spiral equation between the position of the prey and the whale to simulate the helix-shaped movement of humpback whales is formulated as in Eq. (5):

$$\overrightarrow{X}(t+1) = \overrightarrow{D'} \cdot e^{bL} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t), \tag{5}$$

where $\overrightarrow{D}$ is the distance between the whale and prey, b is a constant defining the logarithmic shape, l is random in $[-1, 1]$, and $\cdot$ is an inner product.

Indeed, humpback whales swim along a spiral-shaped path and at the same time in the shrinking circle's direction. Assuming a probability of 50%, selection of either the shrinking encircling movement or the spiral model movement is simulated during iterations of the algorithm as in Eq. (6). Here p is a random number in $[0, 1]$:

$$\overrightarrow{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \overrightarrow{A} \cdot \overrightarrow{D} & \text{if } p < 0.5 \\ \overrightarrow{D'} \cdot e^{bL} \cdot \cos(2\pi L) + \overrightarrow{X^*}(t) & \text{if } p \geq 0.5. \end{cases} \tag{6}$$

3.3. Search for prey

Humpback whales search for prey randomly. The WOA uses $\overrightarrow{A}$, random values in $[-1,1]$. With this assumption, a search agent is able to move far away from a reference whale. In return, the position of a search agent will be updated with respect to randomly chosen input from a search agent, instead of the best search agent found so far, as shown in Eqs. (7) and (8):

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X_{rand}} - \overrightarrow{X}|, \tag{7}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_{rand}} - \overrightarrow{A} \cdot \overrightarrow{D}, \tag{8}$$

where $\overrightarrow{X}$ rand is a random position vector.

4. The proposed method

Here a novel method based on the WOA is presented addressing the shortcomings of previous works as mentioned at the end of Section 2. For the first problem that was mentioned, the server's reliability and time threshold based on the requested web services have been considered. In order to overcome the second drawback, the proposed method uses dynamic rejuvenation by measuring the reliability of web services and time threshold factors. In the proposed method the systems requiring rejuvenation do not receive any request from users but some alternative systems are available that respond to the user's request. This causes the system to never stop. In the suggested method for overcoming the last shortcoming mentioned, several web service providers were considered for responding to receivers. According to the proposed method, the whole system will not crash when a server fails, but it will work at a lower power, which will increase the accessibility of rejuvenation time. In this study, single web services are used. For this web service, we set criteria such as run-time, threshold of reliability, and throughput on the receiver side, which can be requested from the web service. On the server side, we consider criteria such as reliability, failure rate, and rejuvenation rate. The proposed method calculates the mean time to failure (MTTF) for all web services in order to calculate the inspection periods for rejuvenation and set the inspection period to be the lowest MTTF for all web services.

In this paper, the servers provide web services based on receivers' requests for web services under two conditions. First, they need to meet the least reliability requirements for those services. Second, as much as possible, there should be maximum matching between the web service requested and the web service allocated.

The reliability of web services in this research is considered exponentially according to Eq. (9), where λ denotes the rate of failure:

$$R(t) = e^{-\lambda t}. \tag{9}$$

One of the problems in optimizing software rejuvenation is to select a suitable server for web application allocation that can reduce the fail rate. Metaheuristic algorithms such as the WOA are now becoming powerful, simple, and robust approaches to solve this problem. A good rate of convergence and rapid discovery of good solutions that have higher probability and efficiency in finding global optima and are easy in concept and coding implementation compared to other heuristic optimization techniques are the main reasons for selection of the WOA [39,40]. These advantages cause the WOA to be an appropriate algorithm for solving different constrained or unconstrained optimization problems for practical applications without structural reformation in the algorithm.

The main characteristic of the algorithm is the dynamic balance of diversification and intensification in the gradient-free search space. The WOA with the criterion of movement radius is suitable for web services selection. In the suggested method, the criterion of radius is utilized for flexibility of web service provider selection. In this way, if the criterion of the web service provider is not equal to the service request, with the r distance a web service provider can be selected for responding to service requests. According to this, most of the requests that have failed in traditional algorithms can be selected with little decrease in performance and, in general, this solution can reduce the mean failure rate.

The pseudocode for dynamic software rejuvenation in web service systems by using the WOA is illustrated in Table 1.

In the beginning of the algorithm, some of the web servers should be noticed randomly according to requested web services (line 2). At iteration t (line 3), each solution i (line 4) is evaluated according to the fitness function. To achieve this purpose, the Euclidean distances of each data vector $X_{p_{ij}}$ to all web service types are calculated. Each data vector $X_{p_{ij}}$ can be represented as a vector $X_{p_{ij}} = [rt_{ij}, tp_{ij}]$ and includes response time and throughput of the web service j assigned to user i. Moreover, $Z_l = [rt_l, tp_l]$ is the centroid of the web service type l. The Euclidean distance (line 6) can be defined as the difference between each data vector $X_{p_{ij}}$ and all web service types. Then the data vector $X_{p_{ij}}$ is assigned to the web service type l, which results in the minimum Euclidean distance (line 7). When each data vector is assigned to one web service type, the fitness value of each search solution is calculated according to the fitness function (line 8). After evaluating all search agents (whales), the global best solution X^* is updated (line 11).

In lines 12 to 21 the parameters of the mentioned algorithm are updated. Before beginning the subsequent stage, the algorithm convergence is determined. If the result is positive, the algorithm is stopped, or else the subsequent stage is begun (the constant value of X^* in sequential stages is the convergence condition).

Since in the proposed algorithm the radius parameter was considered as the distance of reliability criterion between the web service provider and web service request and because in our studied system the acceptable maximum threshold of this criterion was ten percent, the radius in our proposed method was considered in $[0, 0.1]$.

The used GA for obtaining quality of service in our system was presented in [41]. In our work, for all comparison aims, we used the defined fitness function that is suitable for the proposed rejuvenation method. Each whale in the WOA and each chromosome in the GA show a run mode of the problem that is quantified randomly in the beginning (the web service is allocated to a web service requester randomly in initializing the random population stage). According to Eq. (11), the objective is to minimize the number of failures of web services assigned to all users. Since lower response time and higher throughput result in lower probability of a web service failing, the fitness function of the WOA and GA can be represented as Eq. (13). The objective is to minimize the average response time and simultaneously maximize the average throughput of the assigned web services to all users:

$$\text{fitnessfunction} = \frac{\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m} rt_{ij})}{\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m} tp_{ij})}, \tag{13}$$

where n indicates the number of users and m demonstrates the maximum service request number. rt_{ij} represents the response time of service j by user i and tp_{ij} represents the throughput of service j by user i. It is worth mentioning that with the increasing of the availability, the throughput will strengthen in system. As the suitable result has minimum response time and maximum throughput, the fitness function must be minimized

Algorithm 1 Pseudocode of dynamic software rejuvenation in web services using WOA.

Load requested web services dataset
Initialize each search solution to contain m randomly web service providers
while $t <$ maximum number of Iteration **do**
  for each search solution i **do**
    for each data vector X_P **do**
      Calculate the Euclidean distance of X_P to all web service types
      Assign X_P to the web service type l such that

$$|X_{P_{ij}} - Z_l| = \min |X_{P_{ij}} - Z_{ws}|, \quad ws = 1, 2, \cdots, k \tag{10}$$

      Calculate the fitness using

$$fitness = \sum_{j=1}^{k} \sum_{i=1}^{n} W_{ij} |X_{P_{ij}} - Z_l| \tag{11}$$

      Where W_{ij} is:

$$\begin{cases} 1 & \text{if } jth \text{ service of } ith \text{user belongs to failure service type} \\ 0 & else \end{cases} \tag{12}$$

    end for
  end for
  X^* is the best search solution
  for each search solution **do**
    update a, A, C, L, and p
    if $p < 0.5$ **then**
      if $|A| < 1$ **then**
        update search solution by Eq. (1)
      else if $|A| \geq 1$ **then**
        select random search solution
        update current search solution by Eq. (8)
      else if $p \geq 0.5$ **then**
        update the position of current search solution by Eq. (5)
      end if
    end if
  end for
  $t = t + 1$
  if (the algorithm convergence) **then**
    break;
  end if
end while
return X^*
Assigning web services to the user requests according to X^*
Rejuvenation process according to Eqs. (14) & (15)

in the suggested method. To continue, all whales/chromosomes are sorted in ascending order based on their fitness function value. This means that after sorting, the first one has the most appropriate performance at the current stage (the position vector of the best solution that was considered as a leader in the proposed method). The method assumes this situation as a food position. To continue, the whales/chromosomes should get closer to this position.

Now, for each search agent, the coefficients are updated and the random numbers are generated. Explore/exploit phases are done according to the cited formula in the WOA until the stopping of the maximum number of whales' criteria happens. Then we check for the allowed range for each search agent and check the best solution's agent until max iteration. Finally, the proposed method returns the leader. Now, at this moment, after this suitable process for allocation, in the next steps all services without appropriate conditions go to the rejuvenation process. The method determines the time for the next restart for a web service, considering its duration since the last restart. For this, we take a critical time value that is accepted as the threshold. The web service should be restarted if it has run long enough to exceed the threshold value. The value of the time-out threshold is application-dependent and could be set by the users of the dataset based on the needs of their applications. The threshold determines the condition of rejuvenation for each web service, considering its response time and throughput. The web service should be restarted if it has run long enough to exceed the threshold value. More specifically, each web service i should be rejuvenated if $\frac{rt_i}{tp_i} >$ threshold.

The response time and throughput of aged services are updated according to the following rates:

$$rt_i^{new} = rt_i^{old} * \left(1 - e^{-\left(\frac{tp_i^{old}}{rt_i^{old}}t\right)}\right), \tag{14}$$

$$tp_i^{new} = tp_i^{old} * \left(1 + e^{-\left(\frac{tp_i^{old}}{rt_i^{old}}t\right)}\right), \tag{15}$$

where rt_i^{old} and tp_i^{old} are the current response time and throughput of the aged service i, respectively. Moreover, rt_i^{new} and tp_i^{new} are the new response time and throughput of the aged service i (after rejuvenation process), respectively.

Then the process and allocations will be performed again from the beginning.

5. Simulations and tests

In this paper, we conduct simulations on Windows 7 and a core i5 system. We simulate the structures of single web services based on the various fault tolerance features described in Section 3 using the MATLAB programming language. Meanwhile, we simulate fault tolerance features for servers using the MATLAB programming language. In this simulation, we make it possible to evaluate it based on scenarios of web service requests that allow the scenarios to be created randomly as well as implementing the scenarios that exist in the previous dataset.

One of the most important issues of software rejuvenation is increasing availability and throughput in web service providers with reduction of failed web service requests. In this research, several methods based on the scenarios presented in Table 1 were conducted and the numbers of failed web services were noticed.

In the simulations conducted, we tested receivers' request scenarios based on the selection of appropriate web services with the WOA, GA, and decision tree algorithms. GAs possess certain traits that make them popular among the different metaheuristics. A GA uses both crossover and mutation operators, which make its population more diverse and thus more immune from being trapped in local optima. These attributes make it a good competitor for comparing. Table 1 lists some of the scenario values considered in the simulations.

Figure 1 shows the convergence speed of the proposed method and the GA. By increasing the number of implementations, the fitness function value is first decreased and then fixed.

Table 1. Specifications of different scenarios to run in simulations.

Dataset	Different types of web services	Number of different web services	Number of receivers
Dataset1	10	50	100
Dataset2	5	50	100
Dataset3	10	50	80
Dataset4	5	50	80
Dataset5	10	40	60
Dataset6	5	40	60
Dataset7	10	40	50
Dataset8	5	40	50

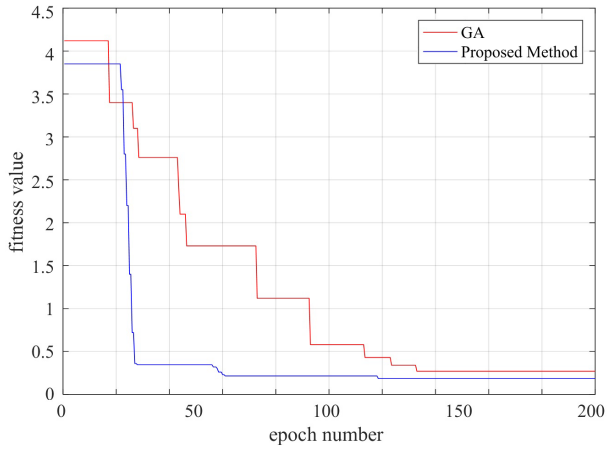

Figure 1. Convergence speed graph of the proposed method and GA.

In this approach, each implementation of the algorithm was iterated 200 times. Moreover, for each one of these iterations, 100 search agents (population size) were considered. The NFE (number of function evaluations) was 20100. The best cost function was 0.184 in the proposed method while it was 0.246 in the GA. In order to evaluate the performance of the proposed rejuvenation method, we considered failed web services to determine the reliability of a system. The number of failures for each scenario is displayed in Figure 2. Moreover, the mean value of failed web services is shown in Figure 3.

To continue, we illustrate the performance of the WOA-based method. As Figure 2 depicts, the WOA has the most flexibility and the lowest failed web service numbers in different scenarios. For instance, according to dataset 8, the WOA-based method has six failed web services, which is two and three less than the GA and decision tree, respectively. As Figure 3 demonstrates, the WOA method has on average 24 percent and 37 percent fewer failed web services than the GA and decision tree, respectively.

In what follows, the proposed method is compared with the existing algorithms in terms of runtime. Runtime is the amount of time taken for a computer program to perform a task. In Figure 4, the mentioned algorithms were studied based on runtimes (x-axis shows the dataset number in Table 2 and y-axis shows time in seconds). In Table 2 the optimization time and overall runtime (on average for all datasets) of the proposed method are compared with the GA. According to Figure 5, the runtime of the proposed method is 9 percent

less than that of the GA and 35 percent more than that of the decision tree. However, the proposed algorithm has high fault tolerance compared to both the GA and decision tree algorithms.

Table 2. Comparison of average runtime (s) of the proposed method and GA.

	GA	Proposed method
Optimization time (assignment)	9.5	8.3
Overall time (assignment + rejuvenation)	12.9	11.7

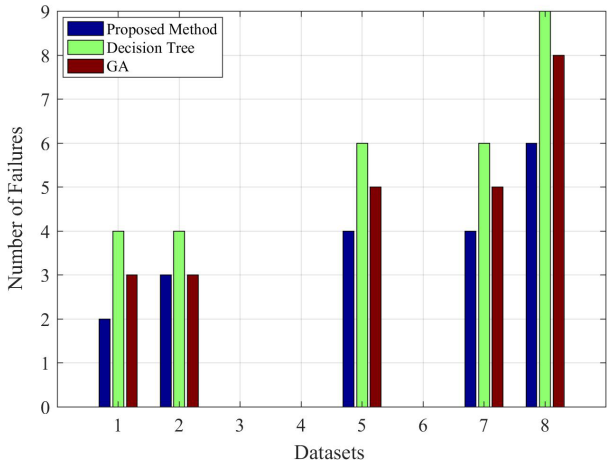

Figure 2. The numbers of failed web services in WOA-based proposed method, GA, and decision tree.

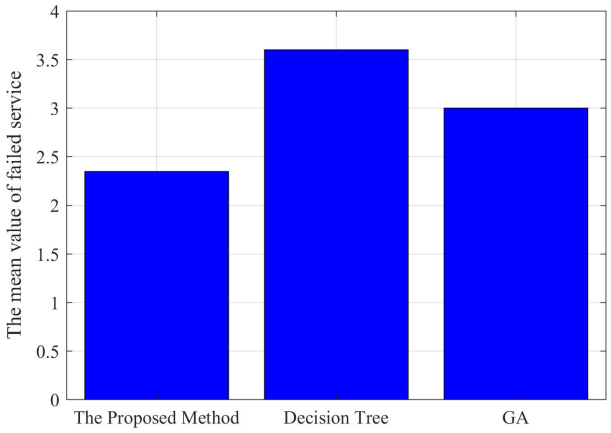

Figure 3. The mean value of failed web services in WOA-based proposed method, GA, and decision tree.

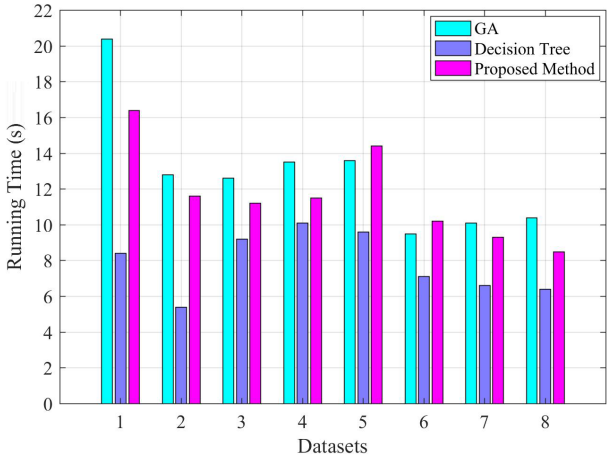

Figure 4. Performance of the methods in terms of run-time.

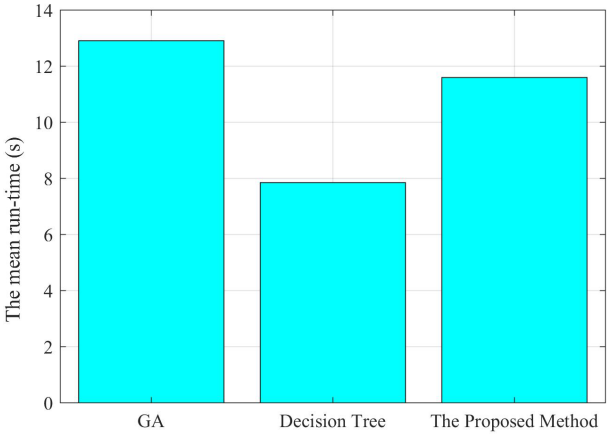

Figure 5. The mean runtime of proposed method and existing algorithms.

6. Conclusions

Software rejuvenation is helpful in preventing the failures in the running process of software execution. Rejuvenation policies are powerful tools for the improvement of the performance of web service-based systems.

In this paper, the effective use of the WOA was investigated for dynamic software rejuvenation based on the characteristics of service providers and receiver requirements. In addition, the allocation of web services and the threshold for rejuvenation were checked dynamically. The systems requiring rejuvenation were detected. As a result of implementation of the proposed technique, there was a considerable improvement in performance compared to well-known algorithms such as the GA and decision tree. The obtained results confirm that the WOA method has on average 24 percent and 37 percent fewer failed web services than the GA and decision tree, respectively. Also, the runtime of the proposed method is 9 percent less than that of the GA and 35 percent more than that of the decision tree method. Despite this increase in runtime compared to the decision tree method, about 37 percent reduction in the number of failed web services shows the high fault tolerance of the proposed method. This indicates that the proposed method based on the WOA can be used for environments that require high fault tolerance.

For future research, in order to improve the accuracy of the presented method, other criteria such as trustworthiness and dependability can be considered for selection of the right web service providers.

References

[1] Beron M, Bernardis H, Miranda E, Riesco D, Pereira M et al. Measuring the understandability of WSDL specification, Web service understanding degree approach and system. Computer Science and Information Systems 2016; 13 (3): 779-807.

[2] Grottke M, Nikora A, Trivedi K. An empirical investigation of fault types in space mission system software. In: IEEE Dependable Systems and Networks Conference; 2010. pp. 447-456.

[3] Torres E, Callou G, Andrade E. A hierarchical approach for availability and performance analysis of private cloud storage services. Computing 2018; 100 (6): 621-644.

[4] Yang M, Min G, Yang W, Li Z. Software rejuvenation in cluster computing systems with dependency between nodes. Computing 2014; 96: 503-526.

[5] Levitin G, Xing L, Huang H. Optimization of partial software rejuvenation policy. Reliability Engineering & System Safety 2019; 182: 63-72.

[6] Mahmood Z, Bowes D, Hall T, Lane P, Petric J. Reproducibility and replicability of software defect prediction studies. Information and Software Technology 2018; 99: 148-163.

[7] Masadeh R, Sharieh A, Sliet A. Grey wolf optimization applied to the maximum flow problem. International Journal of Advanced and Applied Science 2017; 14: 95-100.

[8] Ning G, Trivedi KS, Hu H, Cai KY. Multi-granularity software rejuvenation policy based on continuous time Markov chain. In: IEEE Workshop on Software Aging and Rejuvenation; Hiroshima, Japan; 2011. pp. 32–37.

[9] Fung S, Poon CH, Zheng F. Improved randomized online scheduling of intervals and jobs. Theory of Computing Systems 2014; 55: 202-228.

[10] Hu B, Chicano F. Special issue on meta-heuristics for combinatorial optimization. Journal of Heuristics 2018; 24: 239-242.

[11] Yang YY, Yang B, Wang SH. A dynamic ant-colony genetic algorithm for cloud service composition optimization. International Journal of Advanced Manufacturing Technology 2019; 102 (1-4): 355-368.

[12] Sun D, Chang G, Guo Q, Wang C. A dependability model to enhance security of cloud environment using system-level virtualization techniques. In: Proceeding of 1st Conference on Pervasive Computing, Signal Processing and Applications; Washington, USA; 2010. pp. 305-310.

[13] Dantas J, Matos R, Araujo J, Maciel P. Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. Computing 2015; 97 (11): 1121-1140.

[14] Araujo J, Matos R, Alves V, Maciel P, Souza F et al. Software aging in the eucalyptus cloud computing infrastructure: characterization and rejuvenation. ACM Journal of Emerging Technology Computer System 2014; 10 (1): 1-22.

[15] Guo J, Song X, Wang Y, Zhang B, Li X. The measurement of software aging damage and rejuvenation strategy for discrete web services. Advanced Materials Research 2012; 2: 432-437.

[16] Kulkarni P. Software rejuvenation and workload distribution in virtualized system. International Journal of Innovated Research in Computer and Communication Engineering 2015; 3 (6): 5966-5973.

[17] Torquato M, Umesh IM, Maciel P. Models for availability and power consumption evaluation of a private cloud with VMM rejuvenation enabled by VM live Migration. Journal of Supercomputing 2018; 74 (9): 4817-4841.

[18] Ghayathri J, Pannirselvam S. Categorization of web services based on QOS constraint using decision tree classifier. International Journal of Innovative Technology and Creative Engineering 2016; 6 (3): 338-344.

[19] Umesh IM, Srinivasan GN. Optimum software aging prediction and rejuvenation model for virtualized environment. Indonesian Journal of Electrical Engineering and Computer Science 2016; 3 (3): 572-578.

[20] Mooij A, Eggen G, Hooman J, Wezep H. Cost-effective industrial software rejuvenation using domain-specific models. In: Proceedings of 8th Conference on Theory and Practice of Model Transformations; Berlin; Germany; 2015. pp. 66-81.

[21] Sumathi G, Raju R. Software aging analysis of web server using neural networks. International Journal of Artificial Intelligence & Applications 2012; 3: 11-21.

[22] Nivetha SK, Asokan R. Energy efficient multiconstrained optimization using hybrid ACO and GA in MANET routing. Turkish Journal of Electrical Engineering & Computer Sciences 2016; 24: 3698-3713.

[23] Liu M, Wang L, Gao L, Li H, Zhao H et al. A Web Service trust evaluation model based on small-world networks. Knowledge-Based Systems 2014; 57: 161-167.

[24] Baek S, Lee J, Lee B. Improving fault traceability of web application by utilizing software revision information and behavior model. KSII Transactions on Internet and Information Systems 2018; 12 (2): 1-20.

[25] Gupta R, Kamal R, Suman U. A QoS-supported approach using fault detection and tolerance for achieving reliability in dynamic orchestration of web services. Journal of Information Technology 2018; 10 (1): 71-81.

[26] Meng H, Liu J, Hei X. Modeling and optimizing periodically inspected software rejuvenation policy based on geometric sequences. Reliability Engineering and System Safety 2015; 133: 184-191.

[27] Machida F, Miyoshi N. An optimal stopping problem for software rejuvenation in a job processing system. In: Workshop on Software Reliability Engineering; Washington, USA; 2015. pp. 139-143.

[28] Ning G, Zhao J, Lou Y, Alonso J, Matias R et al. Optimization of two-granularity software rejuvenation policy based on the Markov regenerative process. IEEE Transactions on Reliability 2016; 65 (4): 1630-1646.

[29] Cotroneo D, Iannillo AK, Natella R, Pietrantuono R, Russo S. The software aging and rejuvenation repository. IEEE Transactions on Software Reliability Engineering 2015; 2: 108-113.

[30] Dohi D, Okamura H, Trivedi KS. Optimizing software rejuvenation policies under interval reliability criteria. In: Proceedings of 9th Conference on Ubiquitous Intelligence and Computing; Fukuoka, Japan; 2012. pp. 478-485.

[31] Agepati R, Gundala N, Amari S. Optimal software rejuvenation policies. In: Reliability and Maintainability Symposium; Orlando, FL, USA; 2013. pp. 341-347.

[32] Kaur C, Arora S. Chaotic whale optimization algorithm. Computational Design and Engineering 2018; 5: 275-284.

[33] Tanyildizi E. A novel optimization method for solving constrained and unconstrained problems: modified golden sine algorithm. Turkish Journal of Electrical Engineering & Computer Sciences 2018; 26: 3287-3304.

[34] Karakuzu C. On the performance of newsworthy meta-heuristic algorithms based on point of view fuzzy modeling. Turkish Journal Of Electrical Engineering & Computer Sciences 2017; 25: 4706-4721.

[35] Eberhart R, Kennedy J. Particle swarm optimization. In: IEEE International Conference on Neural Network; Perth; Australia; 1995. pp. 1942-1948.

[36] Mirjalili S. The ant lion optimizer. Advances in Engineering Software 2015; 88: 80-98.

[37] Holland JH. Genetic algorithm. Scientific American 1992; 267: 66-72.

[38] Mirjalili S, Lewis A. The whale optimization algorithm. Advances in Engineering Software 2016; 95: 51-67.

[39] Gharehchopogh F, Gholizadeh H. A comprehensive survey: whale optimization algorithm and its applications. Swarm and Evolutionary Computation 2018; 48: 1-24.

[40] Mohammad HM, Umar S, Rahid T. A systematic and meta-analysis survey of whale optimization algorithm. Computational Intelligence and Neuroscience 2019; 2019: 1-25.

[41] Mardukhi F, Nemat Bakhsh N, Zamanifar K, Barati A. QoS decomposition for service composition using genetic algorithm. Applied Soft Computing 2013; 13: 3409-3421.