ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 2, 412-422 (2021)

www.astesj.com

TTASTESJ

ISSN: 2415-6698

Development and Improvement of Web Services Selections using Immigrants Scheme of Multi-Objective Genetic Algorithm

Khalil Ibrahim Mohammad Abuzanouneh[1,*], Khalil Hamdi Ateyeh Al-Shqeerat[2]

[1]*Department of Information Technology, College of Computer, Qassim University, Buraydah, 52211, Saudi Arabia*

[2]*Department of Computer Science, College of Computer, Qassim University, Buraydah, 52211, Saudi Arabia*

ARTICLE INFO

Article history:
Received: 02 January, 2021
Accepted: 08 March, 2021
Online: 17 March, 2021

Keywords:
Composite Web Services
Quality of Service
Non-functional Factors
Multi-Objectives Algorithm
Immigrant Schemes
Dynamic Optimization Problem

ABSTRACT

Quality of service is a significant part of formulating a composing of web services to satisfy the user's request, especially when several services exist and have been implemented in the same field and functionality. The selection process of web services among many options can be taken based on the quality of service considerations, fitness parameters, multiple objectives, and constraints. Moreover, some non-functional factors such as quality of service indicators such as integrity, cost, availability, reliability, security, and response time. Several composite services are required to achieve the user's requirements. This paper presents an integrated approach based on immigrant schemes and multi-objective genetic algorithm (ISMOGA) to specify and maintain the diversity of composite services more efficiently. Research experiments evaluate the performance of elitism-based ISMOGA based on different probabilities of immigrant mutation to produce an adaptive mechanism and to improve the performance in the dynamic optimization problem. The proposed method is compared to the standard technique of genetic algorithms and multi-objective algorithms. The experimental results show that ISMOGA outperforms other algorithms and improves the searching process. Moreover, the proposed algorithm can quickly adapt to the different resources and environmental changes as well as increases the high-quality solutions to satisfy QoS requirements in different configuration networks and communications channels.

1. Introduction

Distributed services based on systems and functions targeted in several stages of implementation and deployment have differences in non-functional features related to availability, reliability, response time, and performance are used as measures tools of Quality of Service (QoS) [1]. There are several selections to substitute each service with others when a user is mapping a service chain to a specific one by using a request and a response method.

Decision selections affect the behavior of workflow events assigned to QoS that require the deployment phase not to be chosen randomly. The selections should be considered according to QoS requirements such as security process, commercial integrity, main objectives, and constraints. The main challenge is the selection problem of the QoS.

The QoS-based web service contains many challenges that can be represented by different types of optimization methods. The QoS of hybrid web services were used in different fields such as business, academic, transport, monitoring, exchange information, service registering, and web services industrial.

The hybrid web services and location systems introduce a set of advantages related to the mobile application and customers to get benefit information about scheduling problems, resource allocation, and analysis data to retrieve essential and useful information to customers [2]. GPS and GPRS systems in mobile phone and web service applications are used to support location-based services for providing high-quality services. The location-based services are implemented via Google Web Service. GPS System is proposed to provide a high scale of QoS for web services [3]. Creating a web service is an attractive feature of web services that have been integrated into existing web services to

*Corresponding Author Khalil Ibrahim Mohammad Abuzanouneh, Buraydah, Saudi Arabia, ka.abuzanouneh@qu.edu.sa

create a new value-added set of web services to meet user needs and customer requirements.

Web services architectures can be used as a functional application to be implemented in developing composition web service programs. A service application can be integrated into a single process that is included as a set of web services applications to run automatically together. The process of developing technologies to create compositions of web services from many candidate services is a significant cost and time-consuming.

The solution to creating composition problems based on multi-objective characteristics is evolutionary computing. Therefore, researchers in [4] proposed applying Multi-Objective Genetic Algorithms (MOGA) for optimizing service quality using effective solutions. Researchers focus on the main vital QoS issues that rely on web services as follows.

- Web services problems can be increased when creating a composite service with individual services used by external providers.
- QoS problems can arise when using cloud services that are part of hybrid cloud services to construct composite web services.
- QoS problems can arise when scheduling and resource allocation are used in multiple objectives of QoS for web services systems.
- Scheduling composite web services with multi-objectives is very complicated. Scheduling results could be sophisticated and inaccurate with respect to the QoS of web services.
- Planning and organizing composite web services will be very hard. The number and size of composite web services increase the problem.

In the general perspective and according to the research theory of composite web services, the complex problem is related to a scheduling problem, resource allocation, and QoS attributes, so the problem, in this case, is NP-hard problems.

MOGA is used to generate acceptable solutions to optimize composite problems when the search space cannot be determined efficiently using methods of traditional optimization, such as heuristic methods. MOGA operates on an individual's population, for each population could be generated as a possible solution for the optimization problem. Individuals are evaluated by applying their fitness function [5]. The fitness level is used to indicate how well a population individual can be solved the optimization problems [6]. Researchers develop and enhance web service technologies that can provide a way to use various applications based on web service composition [7].

This paper aims to achieve web services based on the highest level of quality to satisfy multi-objective QoS-based constraints and reduce response time at the lowest cost and maximum reliability. Besides, the quality criteria of web services will be optimized at the same time. Moreover, this study focuses on how to apply the proposed ISMOGA algorithm to complex web services. It can be used to develop and improve service quality based on dynamic web optimization problems to increase throughput and reduce execution time and costs.

2. Literature Review

In [8], the authors proposed an algorithm to ensure that software distribution providers are able to manage the dynamic requirements of customers. Besides, they suggested building a map of customer demands to meet various levels of infrastructure in homogeneity and heterogeneity systems. Scheduling mechanisms are designed and implemented to ensure that the scheduling mechanism determines, which type of VM can be initiated to incorporate the heterogeneity of VMs. The approach did not consider the service discovery process, but rather suggested methods of allocating resources to software as a service (SaaS) providers to reduce costs of infrastructure and service level breaches.

Several solutions related to the service selection problem have been formulated in [9]-[11]. This problem consists of locating the group of concretizations to satisfy all QoS constraints that are required by customers.

In [9], a new approach uses Genetic Algorithms (GAs) was proposed to identify a collection of concrete services to restrict abstract services and workflow stages of a composite service to perform QoS constraints established in the service-level agreement (SLA). This approach aims to optimize the function of QoS parameters. The authors did not determine the necessary number of resources selection while the service discovery is discarded in the mapping resource selection. Authors in [10] have proposed a service selection approach without determining the resources set to run the selected services to satisfy the QoS restriction and constraints, therefore the proposed research is not achieved in the services of cloud computing. An optimization evolutionary multi-objective related to service composition as a framework has been proposed in [11]. The proposed framework illustrates a service deployment model to apply two multi-objective genetic algorithms: Extreme-E3 (X-E3) and E3-MOGA. These algorithms have been produced a set of Pareto solutions to satisfy SLAs of service compositions. X-E3 and E3-MOGA locate the number of instances for each concrete selected service to satisfy a certain SLA during the workflow definition and a series of abstract services.

3. Genetic Algorithm

A standard Genetic Algorithm (GA) is a search heuristic of natural selection based on the fittest of individuals. GA has a set of parameters such as a population and candidate solutions, called individuals or creatures. The optimization problems of a genetic algorithm are the better solutions based on a fitness function. The candidate solutions have a set of genotypes or chromosomes, which are mutated. The population individuals of the genetic algorithm can be represented as concrete services. Therefore, the selection individual problem represents a candidate of abstract service. Fitness function is an evaluation measure of a genetic algorithm to select population individuals to get more evolution and feasible solutions of the composite web services [12].

The optimization problems of GA are defined as a constrained and dynamic optimization problem to select the highest QoS of composite web service, the customer's preferences, and to ensure

that a composite web service satisfies the cloud customer resources to guarantee the QoS constraint. This paper uses the MOGA algorithm to solve multi-constraint problems and to apply immigrant schema based on MOGA to produce feasible solutions. It is used to achieve high quality after each change as well as to satisfy all required constraints.

3.1. Genetic Algorithm steps

Five steps are considered in a genetic algorithm:

- Creating an initial population randomly, which consists of a set of individuals according to the available data that must be resolved.
- The fitness value is computed for each individual and then compared to the best fitness value to replace the best value in case the obtained fitness value is better than the previous value.
- Selecting the fittest individuals based on their fitness scores. The individual's migration is performed based on fitness values.
- Crossover processes are performed on parents, which leads to the generation of new individuals.
- The mutation operator is applied randomly to one of the individual characteristics; therefore, the status value is changed to a random value within the terms of population individuals.

3.2. Fitness Functions Calculations

There are a set of individual instances in GAs. The individuals are instances of the Combination of Services (CS), so the fitness values of individuals will affect the probability of being selected individuals. Therefore, the main technique of GAs is the calculations of fitness functions, and the selection of individuals based on global QoS constraints dependent on the user's preferences. The global constraints of QoS are significant references for the fitness function [13]. The user's preferences and the selection rules must be considered to build fitness values, such that the individual fitness value and the individual's selected probability are maximum.

3.3. Developed Genetic Algorithm

This paper improves GA to perform the selection of service composition. GA includes creating and encoding processes using a tree graph for selection parameters and applying the fitness function strategy. The tree encoding method represents a tree graph of services composition between the source of abstract service and multi-destination of concrete services.

There are different types of service composition such as series composition, parallel composition, probability composition, and circulation composition, which are included in the service combination process. A structure of a special tree is used to present the service combination, which is called a tree combination.

4. Design GAs for Dynamic Optimization Problem

This section describes the proposed ISMOGA algorithm for the Dynamic Optimization Problem (DOPs) in the web services composition. The ISMOGA design involves a set of key functions such as individuals' representation, initialization values, fitness functions, selection parameters, immigrant scheme, crossover, and mutation. The abstract service consists of a sequence of concrete services such as adjacent services in the composite services. Hence, it is a natural choice to adopt the services selection parameters using a tree graph and encoding method. For the dynamic optimization problem, the selection parameters are encoded based on crossover, mutation, and the immigrant scheme of ISMOGA. ISMOGA is suggested to solve Composite Services Optimization Problem (CSOP) in dynamic environments and to satisfy QoS requirements. CSOP can be embedded in different group communications and collaborative network applications that are based on real-time delivery such as the video conference service and distance learning applications [14], [15].

Information transmitted between the source and the destination needs to be activated and guaranteed. Therefore, QoS classification of real-time delivery is required for multimedia applications, and tree paths have to be constructed [16]. The cost of tree paths should be calculated to evaluate network resources and VMs configurations.

In ISMOGA design, all parameters and functions are related to the composite services. We used several immigrants of memory schemes and their combination into the GA to enhance its searching capacity for the QoS and user's preferences in dynamic environments. When the configuration of topology is changed, new immigrants or valuable information will be stored in the memory that can help guide the search for feasible solutions in the new dynamic environment [17]. In the simulation experiments, the experiments evaluated these ISMOGA in a dynamic environment under different parameters and configurations to find the best solutions. ISMOGA is used as a composite algorithm that is applied to work with composite services to evaluate their performance.

5. Problem Formulation

ISMOGA is applied to the concrete services package for building abstract services. It is used to determine the QoS constraints for cloud customer services [18]. These constraints (attribute values) are related to a composite service, for example, minimum cost, minimal response time, maximal resource availability, and reliability at the same time.

5.1. Series Services Reliability (SSR)

SSR contains a set of independent services (n).

Figure 1: General structure of Series Services

Let us assume the service event S_i has functioned properly $P(S_i)$. $R_i = P(S_i)$, it stands for series services reliability.

$$Rs = P(S_1 \cap S_2 \dots Sn) = P(S_1) * PS(S_2) * \dots P(S_n),$$

The product law of reliabilities applies to series services of independent systems as shown in Equation (1), where, R_i – the reliability of service *i*.

$$R_s(t) = \prod_{i=1}^{N} R_i(t) \quad (1)$$

5.2. Series Services Availability (SSA)

To compute the availability of independent repair services, we need to apply the product law as shown in Equation (2), where, A_i – the stable-state or temporary availability of service *i*.

$$A_s(t) = \prod_{i=1}^{N} A_i(t) \quad (2)$$

A(s) can be calculated by Equation (3), where, $MTTF_i$ – Mean time to failure, *MTTRi* – Mean time to repair.

$$A(S) = \prod_{i=1}^{n} \frac{MTTF_i}{MTTF_i + MTTRi} \quad (3)$$

5.3. Parallel Services Reliability (PSR)

The complex system consisting of *n* independent parallel services. The system fails if all *n* services fail. Services status can be represented as follows.

S_i – service *i* is working properly, $\overline{S_i}$ – service *i* is not working properly, S_p – parallel services of *n* are working properly.

$$P(\overline{S_p}) = P(\overline{S_1} \cap \overline{S_2} \ldots \cap \overline{S_n})$$

Therefore:

$$P(\overline{S_p}) = P(\overline{S_1})P(\overline{S_2}) \ldots P(\overline{S_n})$$

Parallel services are a system that has *n* independent parallel services.

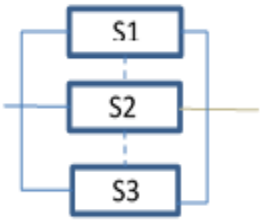

Figure 2: General structure of Series Services.

Parallel services reliability can be defined as follows.

$$R(t) = 1 - \prod_{i=1}^{n} (1 - Ri(t)) \quad (4)$$

Parallel services availability as defined in equation (5).

$$A(t) = 1 - \prod_{i=1}^{n} (1 - Ai(t)) \quad (5)$$

5.4. Series-Parallel Services

Reliability of abstract service $R(A_c)$ has the reliability of each concrete service $R_i(C_s)$. Figure 3 shows Series-Parallel services consist of 2 abstract services and 5 concrete services.

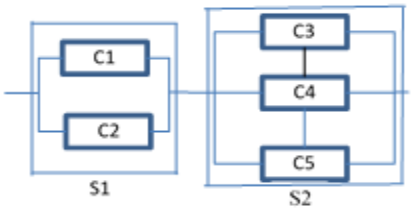

Figure 3: Series-Parallel services

Parallel systems are working if at least one $R_i(C_s)$ of A_{c1} is up as shown in Equation (6).

$$R = [1 - (1 - R1)^2][1 - (1 - R1)^3] \quad (6)$$

5.5. Concrete Service Availability

Concrete Service Availability (CSA) is a metric to measure the probability that a service is not available when it is needed to be used. Qualifications of concrete service that must be available:

- Functioning equipment: service is not out of work for fixing or inspections.
- Service is operating under standard conditions: it operates in a trusted environment and at an expected rate.
- Service works when it is needed: it can be launched at any scheduled time.

5.6. Concrete Service Reliability

Concrete Service Reliability (CSR) is the probability rate that a concrete service carries out correctly in specific time duration.

In the cycle life of the correct process, no repair is required and the system service appropriately follows the required performance specifications.

5.7. Concrete Service Failure Rate

Concrete Service Failure Rate (CSFR) is the frequency of concrete service failure per unit time. It is usually denoted by Lambda (λ).

To calculate the reliability of concrete services, the rate for the severity of the concrete service's failure is predicted as the concrete service is fully functional at the initial stage.

The formula of service reliability is calculated for repairable and non-repairable system services respectively as follows.

$$\text{CSFR} = \frac{1}{\text{MTBCSF}} \quad (7)$$

$$\text{FRCS} = \frac{1}{\text{MTTFCS}} \quad (8)$$

where, MTBCSF – the mean time between concrete services failure (Non-Repairable), while MTTCSF – the mean time to concrete services failure (Repairable).

5.8. Concrete Service Repair Rate

Concrete Service Repair Rate (CSRR) is the rate successful frequency of repair operations achieved in a failed concrete service per unit time The CSRR is calculated in (9).

$$\text{CSRR} = \frac{1}{\text{MTTFCS}} \quad (9)$$

5.9. Mean Time to Concrete Services Failure

Mean Time to Concrete Services Failure (MTTCSF) is the average time before non-repairable CS downtime. Equation (10) computes MTTCSF as follows.

$$\text{MTTCSF} = \frac{\text{Total Hours (CS Operation)}}{\text{Total Number (CS Units)}} = \frac{1}{\lambda} \quad (10)$$

5.10. Mean Time between Concrete Services Failure

Mean time between concrete services failure (MTBCSF) is the average time among inherent failures of repairable concrete services.

The MTCSBF can be calculated using equation (11).

$$MTBCSF = \frac{Total\ Hours\ (CS\ Operation)}{Total\ Number\ (CS\ failures)} = \frac{1}{\lambda} \quad (11)$$

5.11. Mean Time to Concrete Services Recovery

Mean time to concrete services recovery (MTTCSR) is the average time to fix a failed state of concrete service and back to an operational state. This stage includes time spent on the diagnostic and alert process, and before repair activities.

$$\text{MTTSCR} = \frac{\text{Total Hours (CS Maintenance)}}{\text{Total Number (CS Repairs)}} = \frac{1}{\mu} \quad (12)$$

5.12. Mean Time to Concrete Services Detection

Mean time to concrete services detection (MTTCSD) is the average time elapsed between the occurrence of concrete services failure and its detection.

$$\text{MTTCSD} = \frac{\text{Total Hours (CS Incident Detection)}}{\text{Total Number (CS Incident)}} \quad (13)$$

5.13. Availability and Reliability calculations

Calculations are calculated for the availability and reliability attributes of concrete service. The failure rate of CS can be calculated interchangeably based on MTBF and MTTF according to the above calculations.

Reliability is exponentially computed for the decay probability function, which depends on the failure rate of the concrete service, as the failure rate may change over the lifecycle of the operational concrete services.

The average time-based quantities such as MTBCSF or MTTCSF can be used to compute Reliability.

The mathematical formula used to calculate MTBCSF is represented in Equation (14).

$$\text{Reliability (t)} = e^{-\lambda t} \quad (14)$$

Availability determines the component's immediate performance during any given time based on the period between its failure and its recovery. Equation (15) can be used to calculate the Availability.

$$\text{Availability (t)} = \frac{\text{MTBCSF}}{\text{MTTCSF} + \text{MTBCSF}} \quad (15)$$

5.14. Web Service Systems

Web Service Systems (WSS) include multi-services linked together as a complex architecture of the system.

The effective availability and reliability of the WSS depend on the specifications of the individual components as network configurations and service redundancy models.

The network configuration can be parallel, sequential, or mixed communications between network components.

Redundancy model service may cause the abortion of internal service components and change the availability of effective service and reliability performance.

5.15. Reliability Block Diagram

Reliability Block Diagram (RBD) can be used to illustrate the interconnection among individual services. Another option, the analytical techniques can also be applied to perform the large calculations of the complex system. RBD illustrates a hybrid system that contains series and parallel service connections among components of the complex systems as shown in Figure 4.

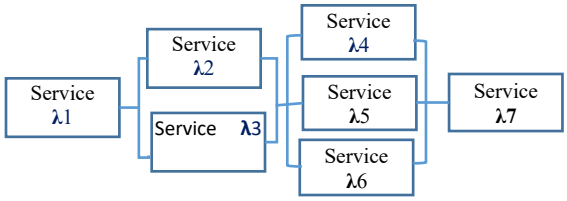

Figure 4: Series and Parallel Service Connections a Hybrid System

The formulae of effective failure rates are used to calculate the availability and reliability of a hybrid system.

The real failure rate associated with the series concrete services (*Cs*) is defined as the overall failure rates per *S*(λc).

$$\lambda(Cs) = \sum_{c=1}^{n} \lambda c \quad (16)$$

MTTF is defined in parallel concrete services, as the mutual total of failure rates of each of *Cs*.

$$\text{MTTCSF} = \frac{1}{\lambda 1(Cs)} + \frac{1}{\lambda 2(Cs)} \quad \ldots + \frac{1}{\lambda n(Cs)} \quad (17)$$

In hybrid systems, the CS must first be collected to parallel or series concrete services. Calculations of hybrid components should be reduced for parallel or series configurations.

It is important to note a few caveats regarding these incident metrics and the associated reliability and availability calculations.

Reliability and availability calculations can be understood in relative terms, for example, in different network applications the failure of the same components can be described differently.

In cloud applications, the measurement values may be inconsistent. Therefore, measurement values such as MTTR, MTTF, MTTD, and MTBF have been calculated in experimental results under controlled environments as follows.

- The functionality of QoS characteristics has been optimized. The customer can reduce response time and cost as possible. QoS characteristics are characterized by different preferences of the customer and can be expressed based on the preference weight.
- Resource constraints must be available in the infrastructure of the IT provider. Therefore, the proposed research will select several services based on available resources.
- Web service composition is the main force for developing services, and thus complex services have specific functions divided into different service components.
- Component functions are fulfilled by several candidates for concrete services. Therefore, service providers provide services with the same functionality, but according to changing circumstances, these services could be combined with a set of combination plans for the same functions while the contents differ [19], [20].
- The VM must contain all requirements of service components so that the total requirements of the composite service must match the configuration and capabilities of VM that are implemented by processing units, the capacity of memory cards, network configurations, supporting devices, and any other components.
- Servers and IT resources are required for users and companies to obtain high quality service during peak load times. The peak capacity is often used to evaluate server usage levels, so we have to find the percentage of CPU utilization.

5.16. Virtual Machine Resources Problem

Virtual Machine (VM) in cloud computing has a set of resources R and its attributes A. V_{Rt} represents VM resources V at time t. We consider A={*Mrt, Srt, Prt, Urt*}, and R={$r_1, r_2, r_3, \ldots r_i$} where, $r_i \in R$. Each resource r_i is defined based on its attributes as follows. V_{Mt} is the VM memory capacity at time t, V_{St} is the storage capacity of VM at time t, V_{Pt} is the VM processing capacity at time t, and V_{Ut} is the VM utilization rate at time t.

We define the fitness function $f(V)$ to evaluate the VM resources (VMR), which has two sections. The first section is an objective function to integrate QoS attributes. The second is an immigrant function that has constrained for optimization problems and selects the individuals obeying constraints and user's preferences to immigrants in the next generation, as calculated by Equation (18).

$$f(V) = \sum_{i=1}^{m}(w_i * q_i') - \beta \sum_{j=1}^{n}(w_j * \Delta q_j) \quad (18)$$

Where, m – the number of QoS attributes, n – the number of QoS constraints, β – a coefficient of an experience rate, w_j and w_j – fitness values and QoS attributes, q_i' – a part of selection parameters to integrate the QoS attribute values. Δq_j – user preferences or global constraints.

The subtraction in the equation shows the immigrant function based on the survival probability of individuals who satisfy user's preferences.

Web Service Systems (WSS) include multi-services linked together as an architecture of complex systems.

The combination services problem between the abstract nodes and concrete services is represented in the tree graph G (N, C).

$N = \{A_i, c_1, c_2, c_3 \ldots c_j\}$, where $A_i = \{a_1, a_2, a_3, \ldots a_i\}$ is a set abstract service nodes, and $C_j = \{c_1, c_2, c_3, \ldots c_j\}$ is a set of concrete services.

Concrete services have a set of resources $C_i(R_n)= \{r_1, r_2, r_3, \ldots r_n\}$ where, R is the resources of concrete services.

Services cost can be defined as,

$$C(S) = c_1(a_1) + c_2(a_2) + c3(a_3) + \ldots + C_i(a_i). \quad (19)$$

while, cost of resource nodes,

$$C(R_i)= c_1(r_1, r_2, r_n) + c_2(r_1, r_2, r_n) + c_3(r_1, r_2, r_n). \quad (20)$$

where, $c_1, c_2 \ldots$ is the total cost of concrete services.

In combination services tree, T is defined as below:

$$T(A_i, C_j) = \{a_i, c_1, c_2, c_3 \ldots c_j\}. \quad (21)$$

ISMOGA optimizes and improves QoS of composition services by minimizing the following objectives i.e.

Definition 1. The total cost of all services nodes (N) along with $C_T(a_i, c_j)$ is equal to the total response time of services $T(a_i, c_i)$.

$$T\left(N_T(a_i, c_j)\right) = \sum_{s \epsilon N_T(a_i, c_j)} (T(s) \quad (22)$$

The total cost of QoS is defined as the total cost of all graph nodes in the composition services as follows.

$$C(N_T(a_i, c_j)) = \sum_{s \epsilon N_T(ai, c_i)} (C(s) \quad (23)$$

Availability of the services nodes A (a_i, c_j) is defined as the maximum available of concrete services nodes at any time associated with the abstract node as shown in Equation (24).

$$A(N_T(a_i, c_j)) = \max\{A(s_i), s \in N_T(a_i, c_j)\} \quad (24)$$

The reliability of the services nodes R (a_i, c_j) is defined as the maximum available concrete services nodes at any time associated with the abstract node as defined in Equation (25).

$$R(N_T(r_i, c_j)) = max\{R(s_i), s \in N_T(r_i, c_j)\} \quad (25)$$

The services selection scheme uses the fitness function to express the required relationships among structure resources of VM based on different types of candidate services as follows.

Series Services refers to activities of services arranged sequentially, Services are executed one by one, so the activity output of one will be used as input to the next service. Figure 5, shows series services are executed P1= {S1, S2, S3}. Besides,

there are controls for data flow from one service to the next service as illustrated below.

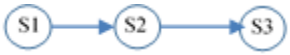

Figure 5: Series Services of VMs

Services wrap defines a structure in which a hidden services activity is nested into services activity (wrapping). It is used to implement a service that needs to execute another service to complete a special subtask.

Figure 6: Wrapped Services (S3) of VMs

In Figure 6, service S2 is a wrapping service that needs an activity of wrapped service S3 to complete a subtask, and then subtask S3 retires its data to wrapping service, this will be repeated many times as necessitated based on the control of the workflow and tasks needed.

Finally, the data flow will exit from service S2 to the next service (S4). The output of the data flow can be expressed as a longer sequence structure, for example, P1= {S1, [S2, S3], [S2, S3], [S2, S3], S2, S4}. In this case, there are three calls to the wrapped service S3, and three are many cycles among the wrapped and wrapping service.

Parallel activities describe a structure formed of a parallel service, included n of activities performed as services parallel and synchronization. The data flow and control continue from the first synchronization point to terminate all tasks of parallel branches as illustrated below.

Figure 7: Parallel Services (S2, S3) of VMs

Figure 7 shows two services S2 and S3 are performed as parallel tasks P1= {S1, [S2, S3], S4}, and service S4 is performed after finishing both S2 and S3.

Conditional Choice is a component composed of an exclusive selection with *n* conditional sections to perform one of these sections and followed by a merge process. In this case, the workflow is supposed into a sequence after calculating the number of loops such as P1= {S1, S2, S4}, P2= {S1, S3, S4}.

Figure 8 illustrates a combination pattern of conditional choices.

Figure 8: Conditional Services (S3, S4) of VMs

At the workflow level, the service attributes must be aggregated for all workflow deployments as shown in Figure 9.

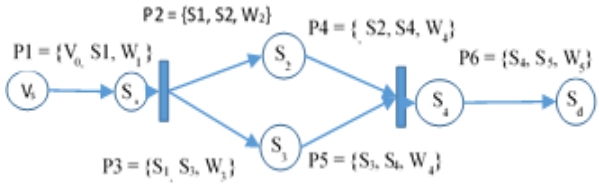

Figure 9: Composition Services Workflow Mapping

The complex architecture of combination services contains sequential and general flow structures. The general flow structure includes non-sequential patterns such as parallel and conditional paths. Services loop can be transformed into sequences by unfolding known cycles. Figure 10 illustrates the VMs' complex architecture of composition services and resources.

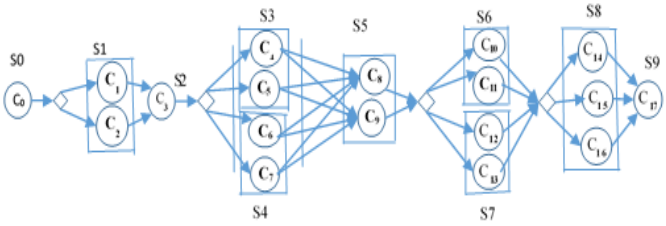

Figure 10: Illustrates the complex architecture of composition services

The matrix representation $M(S_i, C_j)$ is used to illustrate the tree nodes of abstract and concrete services for composition services and calculate the fitness values of candidate services. The tree scheme (T) has four methods to evaluate the fitness values for each service as follows:

$$F(T) = f_1(c_{ij}) + f_2(t_{ij}) + f_3(a_{ij}) + f_4(r_{ij}); S, C \in M(s_i, c_j) \quad (26)$$

The following rules should be applied to get the candidate services.

- The tree service that doesn't satisfy the QoS constraints and users' preferences should be eliminated.
- If there one single service between two candidate services and this service satisfies the QoS constraints, it will be selected automatically.
- Among multiple trees of candidate service, the best service will be selected.
- Satisfy all resources constraints $R_i = \{r_1, r_2, r_3 \ldots r_i\}$ of services infrastructure $S_i = \{s_1, s_2, s_3 \ldots s_i\}$. Requirements of service components located in VM must be less than VM capacities.
- To build one complex system, all required services should be collected together and ordered by tree services $T(S_i, C_j)$.
- In the final step, we built one-line services instead of multi-distribution services after calculating fitness values for all trees.

The fitness method can be used to evaluate all services nodes components. We assume the cost range is $10 \geq c \geq 1$, the minimum response time is $1 \leq t \leq 10$, the maximum resource

availability is $90 \leq a \leq 100$, and reliability at the same time is $80 \leq r \leq 100$. In this example, if the cost constraint c=4, time constraint t = 5, resources availability a = 95, and resources reliability r = 85. Resources constraints of IT infrastructure for VMs are represented as follows, $R = \{r_1, r_2, r_3\}$, where, r_1 – CPUs processing capacity per VM is 32, r_2 – memory capacity per VM is 1024 G, and r_3 – storage capacities per VM is 100 TB.

In all component service paths and their dependencies, various QoS states and constraints are included from abstract service to its concrete services. The tree scheme can illustrate all nodes of services that satisfy users' preferences in the same period.

The proposed algorithms can be applied to sequential and general flow structures. The general flow structure includes non-sequential patterns such as parallel and conditional paths. Loops can be transformed into sequences by unfolding the known number of cycles.

In merged nodes, QoS values, as well as utility values, need to be merged. In parallel merged nodes, since all services in all branches preceding the join are executed, QoS values need to be aggregated based on the properties of specific attributes. For example, the cost and response time for each path are summed, whereas the minimum values are chosen, while for the availability and reliability the maximum values are chosen. At conditional joins, the true conditions of the chosen branch are selected. Figure 11 shows the service candidate graph and all possible paths from the source to the destination services.

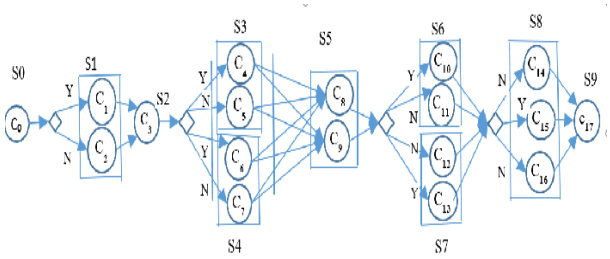

Figure 11: The service candidate graph.

Table 1 shows the matrix representation used to represent all path trees of composite services.

The QoS attributes have various aggregation functions per workflow pattern. The QoS has a set of standard functions such as min, max, sum, and product.

The fitness functions for service cost, time, reliability, and availability are defined respectively.

$$f(c) = \sum_{c \epsilon \mathrm{Ti}(s,d_i)} P_c(s,d_i)\ P_c \in P \qquad (27)$$

where, $f(c)$ – the total cost for each service in the path tree T_i (s, d_i).

$$f(\mathrm{t}) = \sum_{t \epsilon \mathrm{Ti}(s,d_i)} P_t(s,d_i)\, P_t \in P \qquad (28)$$

Table 1: The matrix representation of composite services.

Probability All path										= 1*2*1*4*1*4*3*1=96 paths									
Pi	Concrete Services Path								Di	Pi	Concrete Services Path								Di
P1	c_0	c_1	c_3	c_4	c_8	c_9	c_{10}	c_{14}	c_{17}	P49	c_0	c_1	c_3	c_4	c_8	c_9	c_{12}	c_{14}	c_{17}
P2	c_0	c_1	c_3	c_5	c_8	c_9	c_{10}	c_{14}	c_{17}	P50	c_0	c_1	c_3	c_5	c_8	c_9	c_{12}	c_{14}	c_{17}
P3	c_0	c_1	c_3	c_6	c_8	c_9	c_{10}	c_{14}	c_{17}	P51	c_0	c_1	c_3	c_6	c_8	c_9	c_{12}	c_{14}	c_{17}
P4	c_0	c_1	c_3	c_7	c_8	c_9	c_{10}	c_{14}	c_{17}	P52	c_0	c_1	c_3	c_7	c_8	c_9	c_{12}	c_{14}	c_{17}
P5	c_0	c_2	c_3	c_4	c_8	c_9	c_{10}	c_{14}	c_{17}	P53	c_0	c_2	c_3	c_4	c_8	c_9	c_{12}	c_{14}	c_{17}
P6	c_0	c_2	c_3	c_5	c_8	c_9	c_{10}	c_{14}	c_{17}	P54	c_0	c_2	c_3	c_5	c_8	c_9	c_{12}	c_{14}	c_{17}
P7	c_0	c_2	c_3	c_6	c_8	c_9	c_{10}	c_{14}	c_{17}	P55	c_0	c_2	c_3	c_6	c_8	c_9	c_{12}	c_{14}	c_{17}
P8	c_0	c_2	c_3	c_7	c_8	c_9	c_{10}	c_{14}	c_{17}	P56	c_0	c_2	c_3	c_7	c_8	c_9	c_{12}	c_{15}	c_{17}
P9	c_0	c_1	c_3	c_4	c_8	c_9	c_{10}	c_{15}	c_{17}	P57	c_0	c_1	c_3	c_4	c_8	c_9	c_{12}	c_{15}	c_{17}
P10	c_0	c_1	c_3	c_5	c_8	c_9	c_{10}	c_{15}	c_{17}	P58	c_0	c_1	c_3	c_5	c_8	c_9	c_{12}	c_{15}	c_{17}
P11	c_0	c_1	c_3	c_6	c_8	c_9	c_{10}	c_{15}	c_{17}	P59	c_0	c_1	c_3	c_6	c_8	c_9	c_{12}	c_{15}	c_{17}
P12	c_0	c_1	c_3	c_7	c_8	c_9	c_{10}	c_{15}	c_{17}	P60	c_0	c_1	c_3	c_7	c_8	c_9	c_{12}	c_{15}	c_{17}
P13	c_0	c_2	c_3	c_4	c_8	c_9	c_{10}	c_{15}	c_{17}	P61	c_0	c_2	c_3	c_4	c_8	c_9	c_{12}	c_{15}	c_{17}
P14	c_0	c_2	c_3	c_5	c_8	c_9	c_{10}	c_{15}	c_{17}	P62	c_0	c_2	c_3	c_5	c_8	c_9	c_{12}	c_{15}	c_{17}
P15	c_0	c_2	c_3	c_6	c_8	c_9	c_{10}	c_{15}	c_{17}	P63	c_0	c_2	c_3	c_6	c_8	c_9	c_{12}	c_{15}	c_{17}
P16	c_0	c_2	c_3	c_7	c_8	c_9	c_{10}	c_{15}	c_{17}	P64	c_0	c_2	c_3	c_7	c_8	c_9	c_{12}	c_{15}	c_{17}
P17	c_0	c_1	c_3	c_4	c_8	c_9	c_{10}	c_{16}	c_{17}	P65	c_0	c_1	c_3	c_4	c_8	c_9	c_{12}	c_{16}	c_{17}
P18	c_0	c_1	c_3	c_5	c_8	c_9	c_{10}	c_{16}	c_{17}	P66	c_0	c_1	c_3	c_5	c_8	c_9	c_{12}	c_{16}	c_{17}
P19	c_0	c_1	c_3	c_6	c_8	c_9	c_{10}	c_{16}	c_{17}	P67	c_0	c_1	c_3	c_6	c_8	c_9	c_{12}	c_{16}	c_{17}
P20	c_0	c_1	c_3	c_7	c_8	c_9	c_{10}	c_{16}	c_{17}	P68	c_0	c_1	c_3	c_7	c_8	c_9	c_{12}	c_{16}	c_{17}
P21	c_0	c_2	c_3	c_4	c_8	c_9	c_{10}	c_{16}	c_{17}	P69	c_0	c_2	c_3	c_4	c_8	c_9	c_{12}	c_{16}	c_{17}
P22	c_0	c_2	c_3	c_5	c_8	c_9	c_{10}	c_{16}	c_{17}	P70	c_0	c_2	c_3	c_5	c_8	c_9	c_{12}	c_{16}	c_{17}
P23	c_0	c_2	c_3	c_6	c_8	c_9	c_{10}	c_{16}	c_{17}	P71	c_0	c_2	c_3	c_6	c_8	c_9	c_{12}	c_{16}	c_{17}
P24	c_0	c_2	c_3	c_4	c_8	c_9	c_{10}	c_{16}	c_{17}	P72	c_0	c_2	c_3	c_7	c_8	c_9	c_{12}	c_{16}	c_{17}
P25	c_0	c_1	c_3	c_4	c_8	c_9	c_{11}	c_{14}	c_{17}	P73	c_0	c_1	c_3	c_4	c_8	c_9	c_{13}	c_{14}	c_{17}
P26	c_0	c_1	c_3	c_5	c_8	c_9	c_{11}	c_{14}	c_{17}	P74	c_0	c_1	c_3	c_5	c_8	c_9	c_{13}	c_{14}	c_{17}
P27	c_0	c_1	c_3	c_6	c_8	c_9	c_{11}	c_{14}	c_{17}	P75	c_0	c_1	c_3	c_6	c_8	c_9	c_{13}	c_{14}	c_{17}
P28	c_0	c_1	c_3	c_7	c_8	c_9	c_{11}	c_{14}	c_{17}	P76	c_0	c_1	c_3	c_7	c_8	c_9	c_{13}	c_{14}	c_{17}
P29	c_0	c_2	c_3	c_4	c_5	c_6	c_{11}	c_{14}	c_{17}	P77	c_0	c_2	c_3	c_4	c_8	c_9	c_{13}	c_{14}	c_{17}
P30	c_0	c_2	c_3	c_5	c_5	c_6	c_{11}	c_{14}	c_{17}	P78	c_0	c_2	c_3	c_5	c_8	c_9	c_{13}	c_{14}	c_{17}
P31	c_0	c_2	c_3	c_6	c_5	c_6	c_{11}	c_{14}	c_{17}	P79	c_0	c_2	c_3	c_6	c_8	c_9	c_{13}	c_{14}	c_{17}
P32	c_0	c_2	c_3	c_7	c_8	c_9	c_{11}	c_{14}	c_{17}	P80	c_0	c_2	c_3	c_7	c_8	c_9	c_{13}	c_{14}	c_{17}
P33	c_0	c_1	c_3	c_4	c_8	c_9	c_{11}	c_{15}	c_{17}	P81	c_0	c_1	c_3	c_4	c_8	c_9	c_{13}	c_{15}	c_{17}
P34	c_0	c_1	c_3	c_5	c_8	c_9	c_{11}	c_{15}	c_{17}	P82	c_0	c_1	c_3	c_5	c_8	c_9	c_{13}	c_{15}	c_{17}
P35	c_0	c_1	c_3	c_6	c_8	c_9	c_{11}	c_{15}	c_{17}	P83	c_0	c_1	c_3	c_6	c_8	c_9	c_{13}	c_{15}	c_{17}
P36	c_0	c_1	c_3	c_7	c_8	c_9	c_{11}	c_{15}	c_{17}	P84	c_0	c_1	c_3	c_7	c_8	c_9	c_{13}	c_{15}	c_{17}
P37	c_0	c_2	c_3	c_4	c_8	c_9	c_{11}	c_{15}	c_{17}	P85	c_0	c_2	c_3	c_4	c_8	c_9	c_{13}	c_{15}	c_{17}
P38	c_0	c_2	c_3	c_5	c_8	c_9	c_{11}	c_{15}	c_{17}	P86	c_0	c_2	c_3	c_5	c_8	c_9	c_{13}	c_{15}	c_{17}
P39	c_0	c_2	c_3	c_6	c_8	c_9	c_{11}	c_{15}	c_{17}	P87	c_0	c_2	c_3	c_6	c_8	c_9	c_{13}	c_{15}	c_{17}
P40	c_0	c_2	c_3	c_7	c_8	c_9	c_{11}	c_{15}	c_{17}	P88	c_0	c_2	c_3	c_7	c_8	c_9	c_{13}	c_{15}	c_{17}
P41	c_0	c_1	c_3	c_4	c_8	c_9	c_{11}	c_{16}	c_{17}	P89	c_0	c_1	c_3	c_4	c_8	c_9	c_{13}	c_{16}	c_{17}
P42	c_0	c_1	c_3	c_5	c_8	c_9	c_{11}	c_{16}	c_{17}	P90	c_0	c_1	c_3	c_5	c_8	c_9	c_{13}	c_{16}	c_{17}
P43	c_0	c_1	c_3	c_6	c_8	c_9	c_{11}	c_{16}	c_{17}	P91	c_0	c_1	c_3	c_6	c_8	c_9	c_{13}	c_{16}	c_{17}
P44	c_0	c_1	c_3	c_7	c_8	c_9	c_{11}	c_{16}	c_{17}	P92	c_0	c_1	c_3	c_7	c_8	c_9	c_{13}	c_{16}	c_{17}
P45	c_0	c_2	c_3	c_4	c_8	c_9	c_{11}	c_{16}	c_{17}	P93	c_0	c_2	c_3	c_4	c_8	c_9	c_{13}	c_{16}	c_{17}
P46	c_0	c_2	c_3	c_5	c_8	c_9	c_{11}	c_{16}	c_{17}	P94	c_0	c_2	c_3	c_5	c_8	c_9	c_{13}	c_{16}	c_{17}
P47	c_0	c_2	c_3	c_6	c_8	c_9	c_{11}	c_{16}	c_{17}	P95	c_0	c_2	c_3	c_6	c_8	c_9	c_{13}	c_{16}	c_{17}
P48	c_0	c_2	c_3	c_7	c_8	c_9	c_{11}	c_{16}	c_{17}	P96	c_0	c_2	c_3	c_7	c_8	c_9	c_{13}	c_{16}	c_{17}

where, $f(t)$ is the total of times for each service along with path tree T_i (s, d_i).

$$f(r) = \sum_{r \epsilon \mathrm{Ti}(s,d_i)} P_r(s,d_i)\ P_r \in P \qquad (29)$$

where, $f(r)$ is the total reliability for each service along with path tree T_i (s, d_i).

$$f(a) = \sum_{a \epsilon \mathrm{Ti}(s,d_i)} P_a(s,d_i)\, P_a \in P \qquad (30)$$

where, $f(a)$ is the total availability for each service along with path tree $T_i(s, d_i)$. Table 2 shows the candidate path services based on the proposed ISMOGA algorithm. ISMOGA used the fitness function (*F3*) to evaluate the solution quality of the path among a set of candidate path services (candidate solutions). The maximum fitness value of candidate solutions is chosen.

Table 2: Candidate path Services based on ISMOGA.

Path	Candidate Path Services								Des
P1	C_0	C_1	C_3	C_4	C_8	C_9	C_{10}	C_{15}	C_{17}
P2	C_0	C_1	C_3	C_4	C_8	C_9	C_{13}	C_{15}	C_{17}
P3	C_0	C_2	C_3	C_4	C_8	C_9	C_{10}	C_{15}	C_{17}
P4	C_0	C_2	C_3	C_7	C_8	C_9	C_{13}	C_{15}	C_{17}

The fitness value of cost and time is calculated and converted to a maximum value as shown in Equation (31).

$$F1 = 100 - \left(\frac{1}{2}\left[f_1(t)_{min} + f_2(c)_{min}\right]\right) \quad (31)$$

The fitness value of availability and reliability is calculated by Equation (32).

$$F2 = \frac{1}{2}\left(f_3(a)_{\max} + f_4(r)_{\max}\right) \quad (32)$$

The fitness function *F3* is used to evaluate the cost of solution quality in the *F1* and *F2* as calculated in Equation (33).

$$F3 = \frac{1}{2}(F1 + F2) \quad (33)$$

Table 3 shows the fitness values of the candidate services paths calculated by the ISMOGA.

Table 3: fitness values of the candidate services calculated by ISMOGA.

Path	Candidate Services								Des	$f_1(t)$	$f_2(c)$	$f_3(r)$	$f_4(a)$
P1	C_0	C_1	C_3	C_4	C_8	C_9	C_{10}	C_{15}	C_{17}	%91	%94	%94	%97
P2	C_0	C_1	C_3	C_4	C_8	C_9	C_{13}	C_{15}	C_{17}	%94	%93	%93	%95
P3	C_0	C_2	C_3	C_4	C_8	C_9	C_{10}	C_{15}	C_{17}	%95	%92	%94	%97
P4	C_0	C_2	C_3	C_7	C_8	C_9	C_{13}	C_{15}	C_{17}	%92	%93	%97	%96

The average fitness value for response time and cost

$$F1_{avg} = (f1(Wi) + f2(Wi)) / 2$$

The average of the fitness value for availability and reliability

$$F2_{avg} = (f3(Wi) + f4(Wi)) / 2$$

Table 4 shows the best path (*P3*) of the candidate services based on the ISMOGA. The best fitness value = 94.25%. P3= {C0, C2, C3, C4, C8, C9, C10, C15, C17}.

Table 4: Candidate services path based on the ISMOGA.

Path	f1(t)	f2(c)	f3(r)	f4(a)	F1	F2	F3
P3	95%	92%	94%	97%	93.50%	95%	94.25%
P2	94%	93%	93%	95%	93.50%	94%	93.75%
P4	92%	93%	97%	96%	92.50%	92%	92.25%
P1	91%	94%	94%	97%	92.50%	91%	91.75%

6. Simulation Experimental Results and Analysis

Simulation comparisons were performed on the QoS of composite web services. All experimental results were taken on the same software and hardware to test the studied algorithms. The experiments were executed using the python programming language to determine the efficacy of the web service composition selection.

The experiment analyzed and evaluated the tree paths of combination services and structure resources of an abstract that satisfy users' preferences and QoS constraints using the proposed algorithm (ISMOGA), Standard Genetic Algorithm (GA), and Multi-Objective Genetic Algorithm (MOGA).

The QoS and costumer's constraints are met in composite web services based on abstract services and concrete services satisfying QoS constraints, achieving the resource constraints of the VMs. Figure 15 shows the computation cost in seconds for composite services and one execution path using MOGA and ISMOGA. In both algorithms, the computation cost increases with the task number and the candidate services number. However, the computation cost of the ISMOGA (70 seconds) is less than in the MOGA (95 seconds).

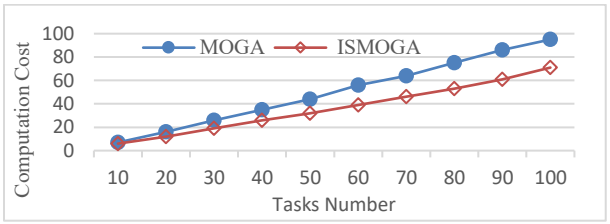

Figure 15: Computation cost of MOGA and ISMOGA algorithms.

Figure 16 shows the relationship between the fitness average and the generation number for MOGA and ISMOGA algorithms using the same number of resource constraints and user preferences.

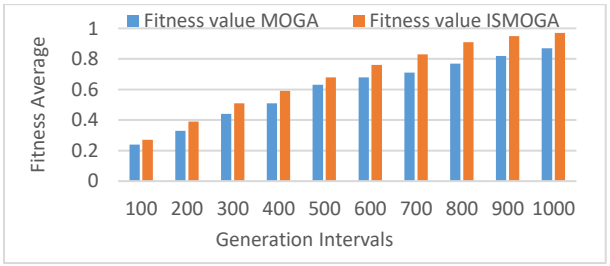

Figure 16: Fitness values and generation number of MOGA and ISMOGA.

In the experiment, we have implemented both algorithms MOGA and ISMOGA according to their techniques. Resource constraints and user preferences were equivalent for both algorithms, as the experimental parameters were the same, including population size = 100, generation number = 1000, point crossover probability = 0.5, point mutation probability = 0.1. Figure 17 shows the comparison results for minimum execution times based on the number of composite services used ISMOGA to get high-quality visual solutions and perform QoS requirements.

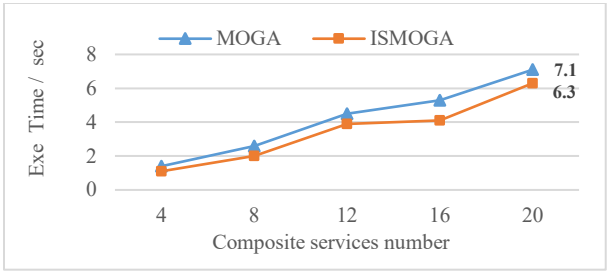

Figure 17: The execution time of the MOGA and ISMOGA

Figure18 shows the results obtained from the search algorithms based on the number of concrete services to obtain the best and most effective solutions.

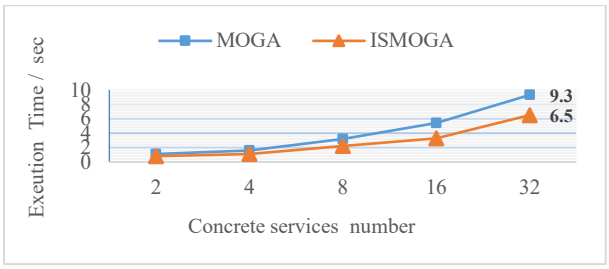

Figure 18: Comparison between search algorithms

Figure 19 shows the characteristics and number of the required resources for VMs in a dynamic environment that is changing rapidly. ISMOGA algorithm is more adapted and efficiently based on changed characteristics in the environmental dynamics (ENDY).

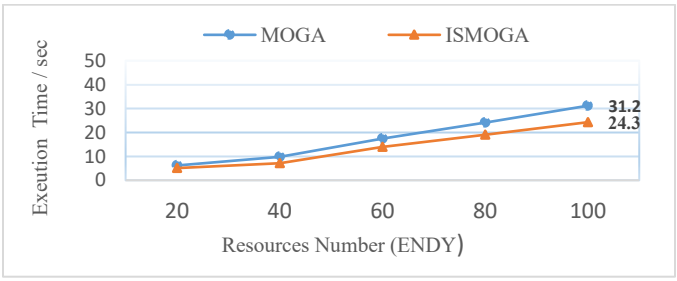

Figure 19: MOGA versus ISMOGA in environmental dynamics.

Figure 20 illustrates a comparison of the average fitness values for availability and reliability obtained for MOGA and ISMOGA. The results show the capability of ISMOGA is more efficient as it can be seen that the obtained time value for execution time is small.

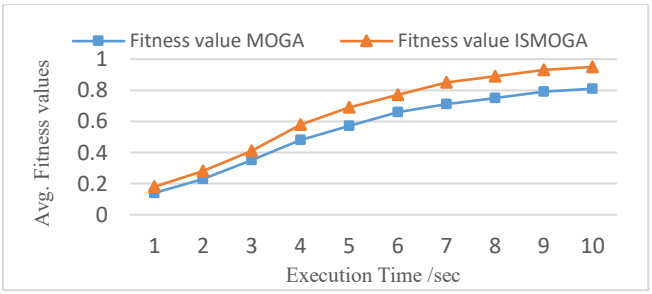

Figure 20: Availability and reliability of MOGA and ISMOGA.

Figure 21 shows a comparison of specific cost values obtained for each algorithm between (0-1000) generation intervals. The result of the analysis explains the cost paths of MOGA and ISMOGA that are satisfied with the resource constraints of the VMs and QoS that are the main references for the fitness function.

The experiment results show that applying ISMOGA search method to get the best solutions for composite web services problems is feasible and effective. The experimental results illustrate the best cost of the proposed algorithm = 260 while the cost of MOGA = 320. ISMOGA adapts quickly compared to MOGA and obtains high-quality visual solutions to satisfy QoS requirements. ISMOGA applied to work with DOPs to evaluate their performance and obtain the best solutions. Furthermore, the immigrant scheme was organized in the ISMOGA to improve its searching capacity.

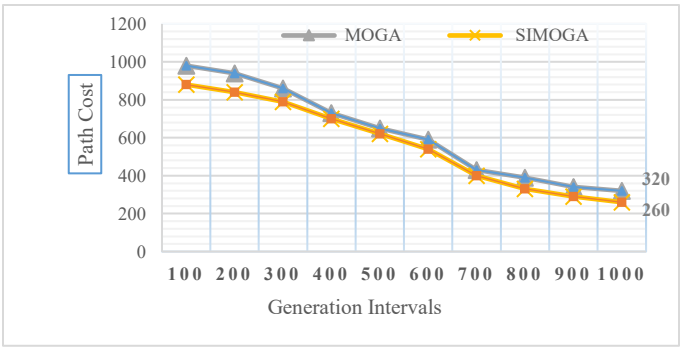

Figure 21: The comparison of cost values in different intervals generation.

Figure 22 shows the result of three types of techniques of genetic algorithm (GA, MOGA, ISMOGA) and their fitness functions. The parameters used in the experiment are population size – 400, crossover probability – 0.7, mutation probability – 0.1, generation – 500, and running times is 50/ms.

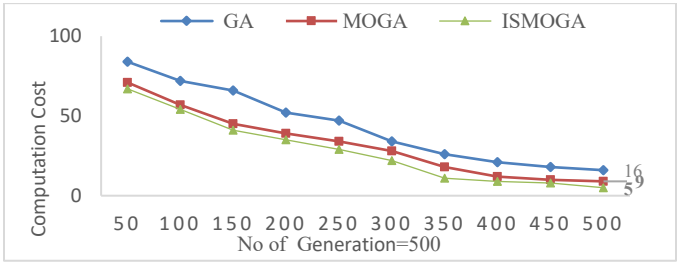

Figure 22: The comparison results of three studied algorithms

The experiment result shows that the proposed ISMOGA algorithm more effective than other algorithms by using the immigrant scheme to improve its searching process.

The experiments were implemented to evaluate the total reliability for each service, where the number of VMs =100, the best value of fitness function = 0.979 of the proposed algorithm, and the number of generation = 500. Figure 23 shows the comparison results of the three algorithms and the maximum value of VM reliability. The experiments are implemented to evaluate the solution quality of availability for each service of composite services and structure resources. The best value of fitness function = 9.66 of the proposed algorithm, and the number of generation = 500.

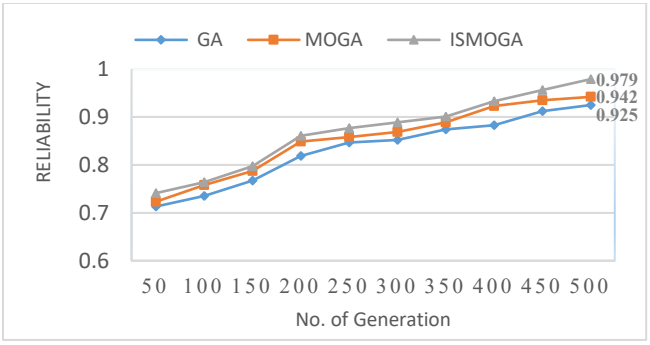

Figure 23: The maximum VM reliability of the proposed algorithm.

Figure 24 shows the comparison results of the genetic algorithms and their maximum value of availability with high quality of services at the path of candidate services.

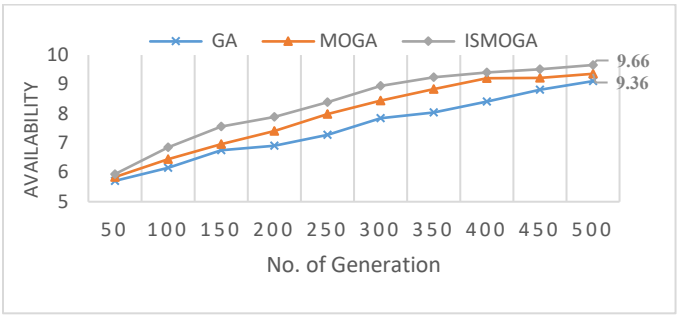

Figure 24. Maximum availability value of the proposed algorithm.

7. Conclusion

In this study, we have proposed and developed a multi-objective genetic algorithm using the immigrant schemes in selecting search methodology to get the best solutions that meet the requirements of QoS in solving the composite services problems. A multi-objective genetic algorithm has been applied to develop and improve the problem of the web service composition to obtain the best services and recourses that are available in the dynamic environment. The study contribution helps a service provider to find the best solutions based on determining a set of concrete services efficiency and satisfy user's preferences and QoS of concrete services. The Interdependence between concrete services and abstract services is built to improve and provide a function with high quality and characteristics in the service requested by the customer, taking into consideration the available preferences and capabilities. The available resource constraints are a significant part, therefore the best paths of the multicast tree in the network environment were analyzed and determined to achieve the highest levels for all service components available in VMs based on configuration, capabilities, and resources to be compatible with VMs in the cloud environment. The experiments are performed using the fitness function of ISMOGA, which gives indications for selecting a group of desired individuals and excluding other groups that do not achieve the multi-objective and constraints. Experimental results show that the proposed algorithm was one of the best scientific solutions in dealing with complex and NP problems.

Acknowledgment

The authors gratefully acknowledge Qassim University, represented by the Deanship of Scientific Research, on the material support for this research under the project number: 5250-COC-2018-1-14-S, during the academic year 1439 AH/2018 AD.

References

[1] P. Wang, Z. Ding, C. Jiang, and M. Zhou, "Constraint-aware approach to web service composition," IEEE Transactions on Systems, Man, and Cybernetics: Systems, **44**(6), 770–784, 2014.

[2] M.E. Ash, "Determination of Earth Satellite Orbits," Defence Technical Information Center, 1972.

[3] D.A. Menasc´e, R. Honglei, and H. Gomaa, "QoS management in service-oriented architectures," Performance Evaluation, **64**, 646–663, 2007, doi: 10.1016/j.peva.2006.10.001.

[4] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in 2005 conference on Genetic and evolutionary computation (GECCO '05), 1069–1075, New York, NY, USA, 2005.

[5] K. Bessai, S. Youcef, A. Oulamara, C. Godart, S. Nurcan, "Bicriteria workflow tasks allocation and scheduling in cloud computing environments," in 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), 638–645, 2012.

[6] V. Bennett, A. Capella, "Developing and deploying a location-based service application," IBM, developerWorks, 2002.

[7] Z. Ding, J. Liu, Y. Sun, C. Jiang, M. Zhou, "A transaction and qos-aware service selection approach based on genetic algorithm," IEEE Transactions on Systems, Man, and Cybernetics: Systems, **45**(7), 1035–1046, 2015. DOI: 10.1109/TSMC.2015.2396001

[8] W. Linlin, S.K. Garg, R. Buyya, "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing , 195-204, Newport Beach, CA, USA, 2011, doi: 10.1109/CCGrid.2011.51.

[9] G. Canfora, M.D. Penta, R. Esposito, M.L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in 7th annual conference on Genetic and evolutionary computation, 1069-1075, Washington DC, USA, 2005.

[10] S. Wang, Z. Zibin, S. Qibo, Z. Hua, Y. Fangchun, "Cloud model for service selection," in IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 666-671, 2011.

[11] H. Wada, J. Suzuki, Y. Yamano, K. Oba, "E3: A Multiobjective Optimization Framework for SLAAware Service Composition," IEEE Transactions on Services Computing, **5**(3), 358-372, 2012.

[12] K. Abuzanouneh, "Develop and Design Hybrid Genetic Algorithms with Multiple Objectives in Data Compression," International Journal of Computer Science and Network Security, **17**(10), 32-39, 2017.

[13] M. Ben Othman, "Survey of the use of the genetic algorithm for multiple sequence alignment," Journal of Advanced Computer Science & Technology, **5**(2), 28-33, 2016. Doi: 10.1016/j.ygeno.2017.06.007

[14] L. Qi, Y. Tang, W. Dou, J. Chen, "Combining local optimization and enumeration for qos-aware web service composition," in 2010 IEEE International Conference on Web Services (ICWS), 34–41, July 2010. DOI: 10.1109/ICWS.2010.62

[15] Y. Feng, L. D. Ngan, and R. Kanagasabai, "Dynamic service composition with service-dependent qos attributes," in 2013 IEEE 20th International Conference on Web Services (ICWS), 10–17, 2013. DOI: 10.1109/ICWS.2013.12

[16] H. Guo, F. Tao, L. Zhang, S. Su, and N. Si, "Correlation-aware web services composition and qos computation model in virtual enterprise," International Journal of Advanced Manufacturing Technology, **51**(5-8), 817–827, 2010.

[17] K. Abuzanounneh, "New Image Processing Techniques Using Elitism Immigrants Multiple Objective of Genetic Algorithms for Disease Detection," International Journal of Computer Science and Information Security, **15**(12), 252-260, 2017.

[18] S. Deng, H. Wu, D. Hu, J.L. Zhao, "Service selection for composition with qos correlations," IEEE Transactions on Services Computing, **9**(2), 291–303, 2016.

[19] M.C. Jager, Optimizing Quality-of-Service for the Composition of Electronic Services. Ph. D. thesis, Berlin University of Technology, 2006.

[20] X. G. Wang, J. Cao, and Y. Xiang, "Dynamic cloud service selection using an adaptive learning mechanism in multi-cloud computing," Journal of Systems and Software, **100**, 195– 210, 2015. Doi: 10.1016/j.jss.2014.10.047