International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Genetic Algorithm based Optimal Service Selection of Composition in Middleware using QoS Correlation

Yashwant Dongre*[1], Pramod Patil[2]

Submitted: 12/11/2022 **Accepted:** 14/02/2023

Abstract: A key role is played by service composition, a critical technique for integrating sophisticated web applications in service oriented architecture. The service selection procedure highlighted Quality of Service as a necessary criterion for the optimum selection of services for the composition process. It can be challenging to find functionally equivalent services that meet the user's nonfunctional requirements. Web services employ Software as a Service to create web applications. When selecting the best services from the input set, we first apply the minimal services technique to decrease the quantity of unsuitable services in the candidate services set. The suggested Genetic Algorithm (GA) correlation-based methodology has a shorter calculation time than the conventional GA-based approach, and it outperforms the current GA-based method, according to the findings of the experimental implementation and statistical analysis.

Keywords: *Optimal selection, Correlation, Service composition, Middleware.*

1. Introduction

Over the past decade, there has been a significant increase in the publication and deployment of web services using distributed networks. A web service is a functional component of the web that can be realized at any time by web-reachable software, DB's, hardware sensors, and a wide range of some actual devices. It is recognized by its Uniform Resource Identifier (URI)[1].

In the service computing environment, it can be challenging to select the web service that will best help build a complex application. In addition, it's crucial to take the end user's wants into consideration. Nowadays, a lot of providers are offering online resources capability as a web service due to the rising demand for online technology. In order to be distributed across a network, a web service exports a description of an application's capabilities [2]. The execution of business operations is supported by a distributed environment through web services [3]. For every comparable service capacity, there are a huge variety of services with varying service quality accessible. The capacity of services belongs in the abstract task that manages functionally equivalent services for every assignment. A provision class called abstract task performs the same tasks as the services in groups [4][5].

[1] *Department of Computer Engineering , Dr. D Y Patil Institute of Technology, SPPU, Pune, India*
ORCID ID : 0000-0002-4312-8704
Department of Computer Engineering , Dr. D Y Patil Institute of Technology, SPPU, Pune, India
ORCID ID : 0000-0002-4073-1428
** Corresponding Author Email: yashwant.dongre@email.com*

The process of creating a composite service involves various steps, including service selection. Selecting one service from a set of functionally equal services with varied the Quality of Service (QoS) is required by the selection problem, which is very difficult. QoS characteristics include, for instance, response time, reliability, availability, reputation, execution time, cost, and pricing etc.[6]. The service quality level is conflicting, entangled, and dependent on other factors. Service providers now have additional options to present to customers as a result of the expanded number of services available to them [7]. Finding the accurate web service to solve a problem has become challenging, though, with so many options available. In this case, the user compels QoS to pick the best service as described by vendors who can satisfy the non-functional needs of the request.

Five important activities construct the service composition process: composition design, service discovery, service selection, resource allocation, and composition execution [8]. The order in which actions or activities are carried out is decided upon by the composition's design. Candidates for services are linked to functional qualities in the repository. The candidate services for the composition design must all meet the requirements for the composite service's functional attributes. All candidate services needed for composite services are found using the discovery of service stage. In the second step of service selection, non-functional attributes are taken into account. By selecting services for composition process, the selection is leveraged to obtain the best values for non-functional qualities (such as response time) [9].

Five operations are divided into runtime and design time events are shown in a service composition process in Fig. 1. The first three processes, such as composition design, service discovery, and service selection, are part of the design role. The latter two steps, such as resource allocation and composition execution, are runtime roles.

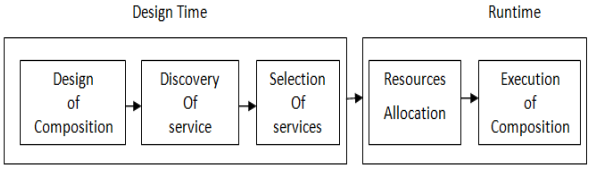

Fig.1 Service composition process

The outline of the paper is structured as follows: The second section provides a list of the related work. Section 3 explains the problem's basic definitions. Section 4 provides a comprehensive description of the proposed approach. Section 5 discusses experimentation. Section 6 discusses the results and discussions. Finally, you'll reach some conclusions in Section 7.

2. Related Work

F. Wang et al. explore the composability-oriented and quality-oriented correlations in [8], respectively, two different types of correlations. Through composability-oriented correlation, the viability of the solution is assessed. The effect of two services' correlation on a composition's overall QoS is measured using a discount percentage. The mathematical model for correlation-aware services was presented by the authors in this study. This work developed the mimetic meta heuristic-based hyper volume estimation algorithm. To achieve better trade-offs among numerous conflicting evaluation criteria, correlation-based local search approaches are also made available.

A service selection technique that considers QoS correlations between services and user needs was introduced in [9] by D. Li et al. The use of decentralized topology prevented performance degradation. Purohit et al. suggested a method for choosing QoS services in [10]. The classification technique was employed in this strategy to monetize potential web services. PROMETHEE Plus is also employed for professional services. A hybrid weight evaluation mechanism is used to maximize variance.

In order to avoid the unreliable real-world process of procuring services, W. Wang et al. offer a service selection technique in [11]. To increase reliability, a two-phase decision is used. The optimization issue is resolved by adapting a convex hull-based method. Large-scale and practical testing are conducted to show the merits of this approach.

Z. Yasmina et al. [12] offer a method for identifying the Top K service compositions that best meet the requirements of the user. This approach addresses the user's overall constraints as well as the dynamic nature of QoS data. The authors rank the services of each abstract class using the probabilistic dominance heuristic. The compositional search space is also explored by the authors using backtracking search.

A rapid heuristic method for multi-criteria service selection is provided by A. S. Kurdija et al. [13] and is designed for multi-user composite workflows with the objective of satisfying all or the majority of the stated QoS requirements. This approach divides the issue into numerous distinct transportation issues using a global-aware utility cost based on anticipated compositional QoS and iterative solution improvements.

M. Moghaddam et al. improved composition of services in [14] to enable the simultaneous submission of several composite service requests. The authors suggest a service choice method centered on combinatorial auctions to address this problem. This method simultaneously matches accessible service deals and demands and does thorough estimations in some parameters with differing economic quantities and request complexity. The findings show that compared to sequential matching systems, the suggested simultaneous approach has a greater success rate in providing services to clients.

To obtain a actual near match to the best arrangement in a fair quantity of period, M. E. Khanouche [15] suggested the teaching and learning oriented composing of services approach (ITL-QCA). This services composition algorithm offers a significant examination ability of the planning search space and only a few tuning parameters, in contrast to bio inspired algorithms. As a result, compositions with a high level of QoS optimality can be produced without the use of any parameters. According to claim of author, this method achieves better than the competition in relations of composition execution time in a large-scale scenario, as authors claimed.

R. Boucetti [16] suggested a method based on GA and Neural Networks (NN) for constructing services in the case of extensive ecosystems. Merging GA with NN results in the sub-optimal composition. The Quality spans are divided into Quality thresholds in order to fit into the hypothetical composition. The GA and a thorough Quality modification are used to arrive at the ideal hypothetical composition. The inaccurate empirical services are removed using NN, keeping only those services that have the same partitions as the atomic hypothetical services that make up the ideal hypothetical composition. In comparison to earlier techniques, they claimed that this method has the fastest composition time and composition effectiveness. In this work, correlation is not used at all.

With the use of computing techniques inspired by nature, X. Zhool [17] reviews techniques for choosing and composing web services. This work reviews and analyses a variety of bio-inspired service selection and arrangement methodologies for QoS-based services. It includes the difficulties, criticisms, and debates surrounding QoS-based composition of services. The authors asserted that effective composition approaches are required for composite services, which might include heterogeneous services.

To accomplish the given objectives, S. Sefati [18] suggests utilizing a genetic algorithm with an artificial bee colony. If the equation of the services chosen by the Genetic Algorithm is enough, the Artificial Bee Colony technique then presents a range of services from which to choose, based on the requirements of each user. Through studies utilising a cloud emulator, this approach's value is evaluated in terms of reliability, availability, and cost; however, the link between qualitative qualities is not taken into account for the elimination of undesired services.

In [19], Y. Zang et al. defined and covered long- and short-term utility, along with their preferences as parties. The research also developed a model that accounts for both the suppliers' and consumers' long- and short-term utility. When determining whether a work will be valuable in the long run, service providers are taken into account. In this research, a non-dominated sorting genetic algorithm is also suggested to deal with the optimization issue in service composition. The suggested method uses tabu search and the k-means mechanism to produce a set of optimal solutions.

Only a few studies address service correlation among candidates, while the majority of the work in the survey takes QoS into account for service computation. However, no attempt has been made to look into the connection between candidate services and QoS restrictions in order to trim candidate services for composition selection, and no GA-based correlation method is successfully employed to determine the best combination of web services.

3. Basic Definitions

In this part, the fundamentals of the issue are defined. The terms "web," "service," "web service," "service execution plan," and "service composition" all have definitions.

3.1 (Web). Hyper Text Markup Language (HTML) or advanced HTML pages are retained on servers connected to the internet, such as those running Apache Tomcat, BEA Weblogic, IIS Server, and others, and make up the World Wide Web (WWW).

3.2 (Service). Any physical action or activity needs to be functional in order to be completed. Examples of services include those offered by banks, telecommunications firms, hospitals, hotels, and railways etc.

3.3 (Web Service). It is a tuple with the four parameters WS=(PID, FDES, INF, QoS), where (1) PID is the primary identification number of the web service, (2) FDES is a description of the functionalities offered by the service, including the input required, the output received, the pre-condition required, and the visible result of the web service, and (3) INF is the web service's mandatory information, including its full name on the WWW, URL location, and service provider. (4) Response time, cost, availability, reputation, and other quality attributes are all part of QoS.

3.4 (Execution Plan). An execution plan is a triple EP=(Ts, Lh, Sd), where Ts stands for a list of tasks, Lh for the probability that the plan will be carried out, and Sd for the execution plan's structural information. An execution plan provides a description of a business process. To do practically any jobs, there could be a sizable number of online services with comparable functionality and various QoS and suppliers.

3.5 (Composite Web Service). A Triple CWS= (WS, SW, QoS) is a composite web service, where WS stands for "Candidate Web Service" and SI for "Structure of the Web Service Plan." QoS stands for the quality of service attribute of the composite web service.

In order to generate composite services, base service is chosen from among many others, each of which has a different QoS and the same functional description for each task, all of which are provided in accordance with the structure of the service execution plan. At the beginning of the service composition process, the user sets a service execution plan. Then, while keeping in mind the layout of the service execution plan, the development of selection of services is carried out to choose web component services. The composition of the selected web services is then completed. The composition process[20] involves numerous steps, with service selection being the most crucial one. Because this phase determines the composite service's quality. When indicating the relationship between two services, the symbol "." is used.

The QoS characteristics of some online services are in real-time communication with the QoS characteristics of other services[21]. Two or more services' QoS qualities may be related to the quality attributes of more than one other service, and the quality attributes of one web service may be related to the quality attributes of two or more other services.

3.6 (QoS). The values of the quadruple attribute set are as follows: (1) IV is the web service's initial quality of service value; (2) CV is its quality of service value with correlation; (3) DEST is services to which the significance of QoS is linked; and (4) SOURCE is services from which the significance of QoS is linked.

Each QoS characteristic has four pieces that characterize it, and different QoS attributes may apply to different services. The correlated value is only utilized if one or more services from particular providers are invoked first; otherwise, the default value for the service is used. When one service is chosen, all of the services in that service's from same providers are shown along with their associated values.

3.7. (Correlation). The act of presuming a relationship between two features is known as correlation [22]. This is represented by the triple Qc= (QoS, DEST, SOURCE), where the SOURCE (source) QoS property is linked to the DEST (destination)..”

4. Proposed Approach

In this section, we outline the stages of GA-based correlated service selection. There are three distinct phases of our approach as shown in Fig 2.

Task Design: Assigning component services to tasks is the goal of the task management module. Using the functional properties of the service, this is possible. As an illustration, consider the bank's credit card service, hotel booking services, and map route services.

GA-based Correlation: This stage enables the persistence of the correlated QoS metrics for each service. From the QoS repository, one can obtain the correlation value of the QoS service. As an illustration, the cost of service A is 15 and the linked value of the QoS cost attribute is 20. To achieve the highest degree of solution fitness, this phase is repeatedly iterated using genetic algorithm processes such as selection, crossover, mutation, and iteration.

Optimal Service Selection: For each user's service request, it offers the most prevalent service composition. Algorithms make up this portion. The algorithm for reducing the number of prospective services in the search space

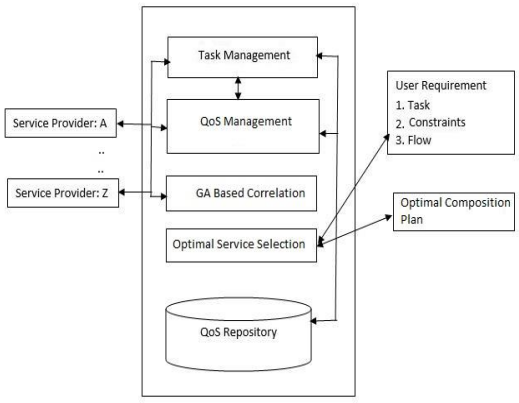

Fig. 2: Service Selection Framework.

4.1 Minimal Services

By choosing services from the whole list of candidate services without using the minimization procedure, the best optimal solution for a combination of services is obtained [23,24]. When there are many potential services, it will take exponential time to find a solution. For instance, there may be m^ndifferent solution forms for a business process with *n* tasks and *m* candidate services per task. However, a workable solution does not include all correlating potential services that are available for a certain task. We suggest minimal services on associated candidate services to address this scenario. The minimization approach guarantees the finding of the optimal solution while reducing the number of linked candidate services per job and, consequently, the number of possible solution combinations. In a word, minimization techniques improve the effectiveness of the service selection process' search space reduction.

The objective is to keep only those candidate services that adhere to global limitations and to discard those that do not. Some global constraint-based strategies for eliminating such candidate services are provided in [25]. Any service with a response time more than or equal to 5 units of time, for instance, can be deleted if the user specifies that the global response time of the composite service should not be larger than 5 units of time. The QoS constraint-based pruning mechanism chooses acceptable services while ensuring an ideal outcome. It is dependent on local and global candidate service thresholds. The computation of the local QoS of the candidate service's threshold also determines whether or not the global constraints are satisfied.

The composition of a composite service will break global limits if any one candidate service does not meet the local criterion. As a result, the search space can be reduced by removing the candidate service from the task's list of candidate services.

Let's take a process with *p* sequential tasks, $\mathrm{T} = \{\mathrm{T}_1, \mathrm{T}_2, .., \mathrm{T}_p\}$ and assume that there are two global QoS constraints: *Q(Response time)=10.0* units time and *Q(cost)=5.0* units cost. It is necessary to determine the lower bound of QoS values for response time and cost for all p tasks when computing local thresholds of QoS for all p tasks. In our example, $\mathrm{C(T1, Response\ time)}_{min=}0.1$, $\mathrm{C(T2, Response\ time)}_{min=}0.1$, $\mathrm{C(T3, Response\ time)}_{min=}0.1$, $\mathrm{C(T1, Cost)}_{min=}1.0$, $\mathrm{C(T2, Cost)}_{min=}1.0$, $\mathrm{C(T3, Cost)}_{min=}1.0$, for p=3.

For each of the three activities, we are now able to calculate local QoS thresholds. Since the lowest response time value of the task T_1 is equal to 0.1 in the best scenario and the response time is an additive attribute, for example, no service with a cost larger than 9; *(C(Response time)* -

$C(T_1, Response\ time)_{min}) = 9.9$ of the task T_1 can be included in a workable solution. The created service will go against international laws. A candidate service's value for any QoS attribute may be removed from the collection of tasks for that service if it does not meet the requirements of a local threshold. Only the candidate services that breach the global limits will be eliminated from the list of candidate services for the tasks according to our proposed method.

The minimum and maximum values of this QoS attribute that can be provided by services (i.e., $C(T_i; Q_p)_{min}$ *and* $C(T_i; Q_p)_{max}$) as well as the value of the needed global constraint $C(Q_y)$ determine the value of the local threshold $C_{Lth}(T_i, Q_p)$ for the P_{th} attribute of the task T_i.

To do this, we suggest calculating local thresholds for each business task using a constraint optimization model. This model can be used to create generic business process architectures that include sequential, parallel, choice, and loop patterns with a variety of quality attribute categories. The proposed approach relies on the atomic nature of business operations in order to keep things simple. Alternately to reduce the services to minimal level may include approaches like service skyline which are mentioned [29-32].

Here is the proposed threshold model:

$$F(Q) = Minimize \sum_{Ti=1\ and\ i=!j}^{i=p} C(T_i, Q_p) - C(T_j, Q_p) \quad \text{--- (1)}$$

4.2 Genetic Algorithm

The primary application of genetic algorithms is resource optimization. This refers to maximizing or limiting qualities in accordance with a function, also known as an objective function from the possible attributes pool in a certain context. Our ability to choose wisely is now enabled by this. [33] First, higher value is better, and second, lower value is better qualities, are the two primary categories. Second should be minimized, while first should be highlighted. One programmed with biological underpinnings in artificial intelligence is the genetic algorithm.

Compared to traditional optimization methods, a genetic algorithm has a number of benefits [34]. The chromosomes and genetic patterns that may be seen in many groups are fundamental to the basic idea of GA, which is founded on genetics. GA emphasizes the persistence of those people who, over many generations, are best qualified to handle the issue. There are several strings that resemble chromosomes in every generation. Depending on the value in the search space, each generation represents a potential resolution that might be found. Individuals in the population are then exposed to evolutionary processes after that.

Once the population reaches the stage when offspring are not generated differently from the previous generation, fitness values are analyzed to determine the optimal solution. In order to address the problem, we first establish a population in GA, with each member representing a workable suitable solution. Each person is represented by a single constant, typically a binary range approach, with a finite length and a defined range. As a result, chromosomes, a solution, are composed of many genes.

Each response that reflects a unique solution is given a fitness value that is calculated using the objective's utilization [35]. The best fitness value is then determined. GA aims to leverage selective "reproduction" of solutions by merging chromosomal data, elevating the child population over the parent. Three operators—crossover, mutations, and selection—are used to create subsequent populations once the initial one is produced at random.

4.3 The algorithm for Minimizing Services

This section introduces and explains the QoS constraint-based minimization algorithm. These steps follow Algorithm 1's format. The collection of potential services (CSi), free of any quality restrictions, serves as the algorithm's input. After the pruning process, a reduced collection of services called a set of candidate services is produced. Line 1 sets the list of candidate services for pruning to zero. Line 3 receives the minimal value with the quality attribute, and Line 2 sets the service threshold to 0. When it comes to quality, Line 3 is most valuable. Response time and cost are the values that line 5 sets for Q. Response Time, the first item in the collection of Q, is retrieved at line 6. In line 7, there is a for loop that checks that the condition is empty.

For both the cost and the quality attribute, such as response time, Line 8 adds local threshold values to all tasks. For the threshold computation, line 9 receives the following quality characteristic from the collection of Q. Lines 10 and 11 add the currently selected candidate service for the i_{th} job and check for an empty condition. Each service is checked on lines 12, 13, and 14 to see if it meets the threshold values or not. CS_i will be added if the constraint is not met by the service. It will be added to PCS_i as line 16 if the constraint is satisfied by the service. If the PCS_i indicator is empty, no service has violated the threshold-imposed constraint, according to lines 17 and 18.

Algorithm 1:	*Minimization of candidate web services for Task T_i*
Input:	*The set of candidate web services CWS_i*
Output:	*The set of minimal candidate web services $MCWS_i$*

1	$MCWS_i$=null;
2	C_{Lth}=0;
3	Min=minValues(CWS_i);
4	Max=maxValues(CWS_i);
5	Q ← {Response time, Cost};
6	Q_p ←first(Q);
7	For Q_p not empty do
8	C_{Lth} (T_i, Q_p) ←localThreshold(T_i, Q_p)
9	Q_p ←next(Q);
10	$MCWS_{ij}$ ←first (CWS_{ij})
11	For $MCWS_{ij}$ not empty do
12	For each Q_p in Q do
13	If C (CWS_{ij}, Q_p)> CLth (T_i, Q_p) Then
14	CWS_i=CWS_i Union CWS_{ij}
15	Break;
16	$MCWS_i$=$MCWS_i$ Union CWS_{ij}
17	If $MCWS_i$=null;
18	Then no web services removed.

4.4 The algorithm for selection using GA

Algorithm 2:	*GA for Candidate service selection for task T_i*
Input	*Set of minimal candidate web services CWS_i*
output	*Optimal candidate web services $SCWS_i$*
1	*For each $SCWS_{ij}$ from set of $SCWS_i$ (set of minimal services)*
2	*Compute objective function of web service OF ($SCWS_{ij}$) using equation 1.*
3	*Preform ranking of web services using objective function values obtained from eq.1.*
4	*For each $SCWS_{ij}$ from set of CWS_i*
5	*For each Q_p in Q*
6	*Sort services by ascending order with the priority of correlated services.*
7	*Create initial population using selected $SCWS_{ij}$*
8	*Calculate fitness of each population using eq.2*
9	*Select best individual from population with maximum fitness*
10	*Perform Crossover operation*
11	*Perform Mutation operation*
12	*Repeat goes to step 8 till maximum fitness.*
13	*Return best composition plan with selected services with MAX fitness value.*

To begin, each possible service's fitness is determined using the objective function F outlined in equation 2. (Lines 3 and 4). Each candidate service's fitness is represented by its normalised QoS values, which are computed. Instances of local optimum are prevented by using Qmax and Qmin when determining normalised values. To rank using normalisation of the quality of service, use equation 2. Lines 5 and 6 illustrate how we arrived at the ranking of services based on normalised values by computing the QoS correlation for each service and sorting the QoS values of each prospective service in ascending order. The steps necessary to apply the GA method to the connected services and deliver the optimal composition plan are carried out in lines 7 through 12.

$$F(Sij) = \sum_{i=0}^{p} Wi * \frac{Qmax - Q(Sij)}{Qmax - Qmin} \text{-----(2)}$$

5. Experimentation

On a computer system with an 8GB RAM, 64-bit Intel Core i3, and Fedora 35 operating system, experimental simulation is carried out. We employed the Java 17 programming language to put the suggested algorithms into practice. Using the Maven repository and the Spring Boot framework (2.7), candidate web services were generated. In our hypothetical situation, there were a total of 15000 candidate web services created. They randomly determined how their QoS correlated with one another for each abstract task. The correlation proportion used is between 0% and 50%, meaning that for each alternate candidate service we generated, the service's QoS offering was connected with values that were half as high as the original values. Utilizing poison distribution, the response time and cost QoS attributes are taken with their minimum and maximum values. Both the cost restriction and the reaction time limitation were created at random.

Assuming a poison uniform distribution over the intervals [0.1, 5.0] and [1.0, 10.0], respectively, the response time and cost are generated. The implementation work takes into account composite services that have a sequential structure. By altering the amount of composite services per task (i.e. 100, 200, 300, 400, and 500), the experiment is conducted 20 times on average to determine the computation time (selection time) in milliseconds. Changing the amount of tasks per activity, such as to 5, 10,

or 15, also allows for the computation time (selection time) to be noted. Response time and cost each have a set weight of 0.5, with the requirement that the total of all weight values should equal 1.

The GA parameters were configured to produce 200 people as the initial population, 100 generations, a roulette wheel for selection, a one-point crossover with a crossover chance of 0.9, and random (single bit) mutations with a mutation probability of 0.1.

5.1 Model of Population as Solution

The best possible set of tasks with the least amount of service cost and fastest possible response time should be included in the solution to this issue. As a result, each chromosome symbolises a potential component of the service. An array of integer numbers serves as the representation for each solution. Task numbers are represented by locations in an array, and candidate service identification numbers are indicated by data. Table 1 exhibits the representational model.

Table 1: Tabular representation of formation of composite service

	Task 1	Task2	Task 3	Task (n-1)	Task n
Selected Service Location Index(**SSLI**)	SSLI(1 to m) i.e. [**2**]	SSLI(1 to m) i.e**. [3]**	SSLI(1 to m) i.e. [**m**]	SSLI(1 to m) i.e. **[m-1]**	SSLI(1 to m) i.e. **[1]**
1	CS11	CS21	CS31	CS(n-1)1	**CS(n)1**
2	**CS12**	CS22	CS32	CS(n-1)2	CS(n)2
3	CS13	**CS23**	CS33	CS(n-1)3	CS(n)3
--	--	--	--	--	--
m-1	CS1(m-1)	CS2(m-1)	CS3(m-1)	**CS(n-1)(m-1)**	CS(n)(m-1)
m	CS1(m)	CS2(m)	**CS3(m)**	CS(n-1)m	CS(n)(m)

Whereas the initial solution is an array with n integer members and CS is the Candidate Service: A[n], has n total tasks. A[0]=2 indicates that the second candidate web service is chosen for the first task. A[n]=4 denotes that 4 candidate web service from a set are chosen for the (n+1)th task. In this case, there are n tasks in activity or given business process and m is the maximum number of candidate web services that can be made available for each task, so the solution is: A[2,3,m,m-1,1].

6. Results and Discussions

Utilizing computation time (selection time) and the effectiveness of the proposed and current methods is assessed. The selection time of the suggested algorithm is represented as computation time which is given in table 2. The ratio between the quality value of the composite service produced using the algorithm and the composite service's optimal quality value achieved using the algorithm is called the optimality ratio.

Calculation Time vs the Amount of Services: The implementation is used to verify the algorithm's scalability. The initial number of tasks is 5, and it is then increased by 5 steps until it reaches 15. There are between 100 and 1000 potential candidates for the task, with each number increasing by 100.

According to Fig. 4, as more candidate services are considered to satisfy the QoS threshold values, the computation time grows. Then, there will be a total of 5 tasks, with 100–1000 candidate services for each job. According to Fig. 5, as more candidate services are considered to satisfy the QoS threshold values, the computation time grows. Figures 4, 5, and 6 illustrate how our methodology requires less computation time for selection than approaches without a correlation method for varying number of tasks per activity.

Table 2: Computational time of proposed and existing approach

Sr no	No. of Services per task	Computational time of Proposed GA Approach with Correlation in ms			Computational time of GA Approach in ms		
		Tasks =5	Tasks =10	Tasks =15	Tasks =5	Tasks =10	Tasks =15
1	100	14236	20776	30690	17668	24529	37699
2	200	16770	31545	32355	23211	39314	41381
3	300	29337	42367	34400	30357	49278	42901
4	400	29670	45081	37438	37120	53578	47988
5	500	28238	48980	33093	38347	58212	55003
6	600	32519	49953	41277	41526	57486	59174
7	700	29388	50384	44987	39485	58684	62889

8	800	35825	51122	50067	42686	59681	62522
9	900	38234	51997	53590	46318	59848	66921
10	1000	36731	48295	60126	48088	62088	70091

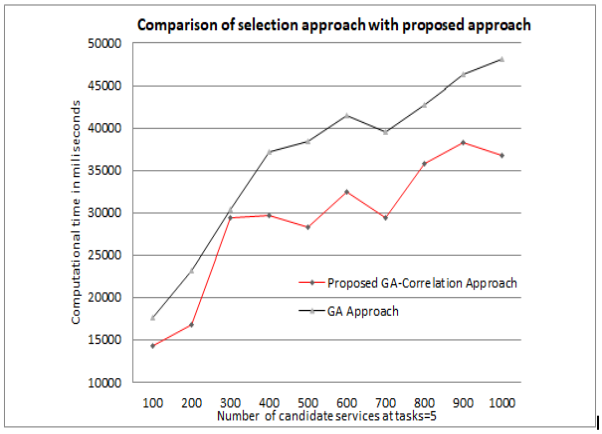

Fig.4: The comparison of proposed algorithm with existing approach for tasks=5

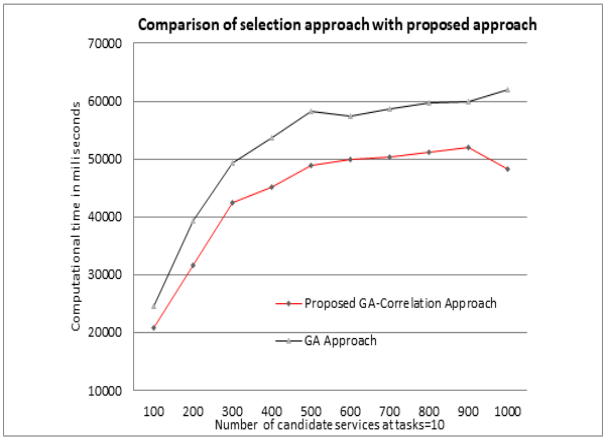

Fig.5: The comparison of proposed algorithm with existing approach for tasks=10

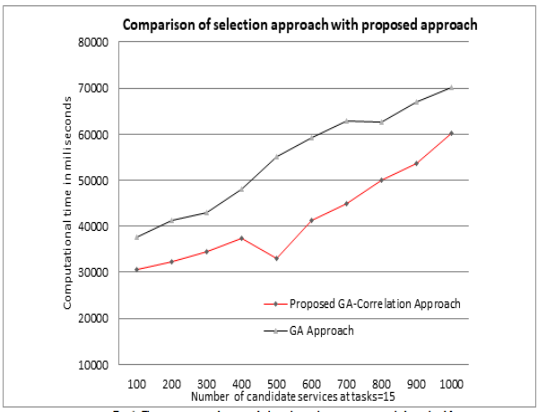

Fig.6: The comparison of proposed algorithm with existing approach for tasks=15

7. Conclusion

In this paper, work offers a method for dealing with the QoS correlation and quality restrictions of candidate services. A GA-based optimization model that enables the choice of the optimal configuration of candidate services is the foundation of the proposed method for choosing candidate services to be included in a composite service. It is shown, using the outcomes of an experimental simulation, that the strategy including quality restrictions and GA correlation outperforms the current regular GA methodology. The findings demonstrate that, with a step increase of 100, the suggested method scales for 100 to 1000 services per task, indicating a large environment.

But in this study, planning is carried out while taking the sequential process into consideration. As a result, in our future work, we will also take alternative workflow designs into account.

Author contributions

Yashwant Dongre: Conceptualization, Methodology, Software, Field study

Pramod Patil: Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, “Book on Web Services: Concepts, Architectures and Applications”, Springer, 2010, pp. 1-369.

[2] X. Liang, A. Qin, K. Tang, K.Tan, “QoS-aware Web Service Selection with Internal Complementarity”, IEEE Transactions on Services Computing, 2016, pp-14, vol. x, no. x.

[3] R. Micillo, S. Venticinque, N. Mazzocca, R. Aversa, “An Agent-Based Approach for Distributed Execution of CompositeWeb Services”, IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2008, pp.1-6.

[4] Y. Liu, A. Ngu, L. Zeng, “Qos Computation and Policing in Dynamic Web Service Selection”, Proceedings of 13th International Conference on World Wide Web, 2004, pp. 66-73.

[5] V. Cardellini, E. Casalicchio, V. Grassi, F. Presti, “Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes, 2007, pp. 743-750.

[6] A. Masri and Q. Mohmoud, “QoS based discovery and Ranking of Web Services.” Proceedings of IEEE conference on computer comm and networks, pp. 529-534, 2007.

[7] W. Ahmed, Y. Wu, W. Zheng, “Response Time based Optimal Web Service Selection”, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, 2013, pp. 1-11.

[8] F. Wang, Y. Laili and L. Zhang, “A many-objective memetic algorithm for correlation-aware service composition in cloud manufacturing,” INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH, vol x, no. x, pp. 1-20, 2020.

[9] D. Li, D. Ye, N. Gao and S. Wang, "Service Selection With QoS Correlations in Distributed Service-Based Systems," in IEEE Access, vol. 7, pp. 88718-88732, 2019, doi: 10.1109/ACCESS.2019.2926127.

[10] L. Purohit and S. Kumar, "A Classification Based Web Service Selection Approach," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2018.2805352, pp.1-14, 2018.

[11] W. Wang, Z. Huang and L. Wang, “ISAT: An intelligent Web service selection approach for improving reliability via two-phase decisions,” Journal of Information Sciences, vol 433, no 434, pp. 255-273, 2018.

[12] R. Z. Yasmina, H. Fethallah, and D. Fedoua, “Selecting Web Service Compositions Under Uncertain QoS,” Springer International Publishing, CIIA 2018, IFIP AICT 522, pp. 622-634, 2018.

[13] A. S. Kurdija, M. Silic, G. Delac and K. Vladimir, "Fast Multi-Criteria Service Selection for Multi-User Composite Applications," in IEEE Transactions on Services Computing, vol. x, no. x, pp. 1-14, 2019, doi: 10.1109/TSC.2019.2925614

[14] M. Moghaddam and J.G. Davis, “Simultaneous service selection for multiple composite service requests: A combinatorial auction approach,” Decision Support Systems, 2019

[15] M. E. Khanouche, N. Atmani and A. Cherifi, "Improved Teaching Learning-Based QoS-Aware Services Composition for Internet of Things," in IEEE Systems Journal, vol. 14, no. 3, pp. 4155-4164, Sept. 2020, doi: 10.1109/JSYST.2019.2960677.

[16] R. Boucetti, O. Hioual, and S. Hemam, “An approach based on genetic algorithms and neural networks for QoS-aware IoT services composition”, Journal of King Saud University –Computer and Information Sciences vol. 34, (2022)pp. 5619–5632

[17] X. Zhaol, R. Li1 and X. Zuo, ” Advances on QoS-aware web service selection and composition with nature-inspired computing”, IET Journal and CAAI Transactions on Intelligence Technology, 2019, Vol. 4, Iss. 3, pp. 159–174.

[18] S. Sefati and S. Halunga, “A Hybrid Service Selection and Composition for Cloud Computing Using the Adaptive Penalty Function in Genetic and Artificial Bee Colony Algorithm”, Sensors 2022, 22, 4873 pp. 1-22

[19] Y. Zhang, F. Tao, Y. Liu, P. Zhang, Y. Cheng, and Y. Zuo, “Long/short-term utility aware optimal selection of manufacturing service composition towards Industrial Internet platform”, IEEE Transactions on Industrial Informatics, 2019. pp. 1-11.

[20] C. HANG and M. SINGH. “Trustworthy Service Selection and Composition” ACM Trans on Autonomous and Adaptive Sys., Vol. 5, No. 4, October 2010, pp. 1-18.

[21] S. Deng, H. Wu, D. Hu, and J. Zhao, “Service Selection for Composition with QoS Correlations”, IEEE TRANSACTIONS ON SERVICES COMPUTING, vol. x, no.x, 2014, pp-1-14.

[22] Y. Du, H. Hu, W. Song, J. Ding and J. Lu, “Efficient Computing Composite Service Skyline with QoS Correlations”, 15th IEEE International Conference on Services Computing, 2015, pp.41-48.

[23] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. “Qos-aware middleware for web services composition.” IEEE Transactions on Software Engineering., vol. 30, no. 5, pp. 311-327, 2004.

[24] A. Hassine, S. Matsubara, and T. Ishida. “A constraint-based approach to horizontal web service composition.” Proceedings of International Semantic Web Conference, pp. 130-143, 2006.

[25] Q. Yu and A. Bouguettaya. “Computing service skyline from uncertain qows”. IEEE Transactions on Services Computing, 3(1):16-29, 2010.

[26] M. Moradi, and S. Emadi, “A Review of Service Skylline Algorithms”, Journal of Soft Computing and Decision Support Systems, Vol. 3, No. 3, 2016, pp.55-58.

[27] Q. Yu and A. Bouguettaya. “Efficient Service Skyline Computation for Composite Service Selection”. IEEE Transactions on Knowledge and Data Engineering, 2011, 268- 281.

[28] H. Ayed, F. Dahan, T. Alfakih, H. Mathkour, and M. Arafah, “Enhancement of Ant Colony Optimization for QoS-Aware Web Service Selection” IEEE Access, DOI 10.1109/ACCESS.2019.2927769 2017, pp.1--12.

[29] L. Purohit and S Kumar, “Exploring K-Means Clustering and skyline for Web Service Selection”, 11th International Conference on Industrial and Information Systems, 2016, pp. 603-607.

[30] L. Purohit and S Kumar, “Clustering based Approach for Web Service Selection using Skyline Computations”, IEEE International Conference on Web Services, 2019, pp. 260-264.

[31] Y. Yang, F. Dong, J. Luo, “Computing Service Skycube for Web Service Selection”, Proceedings of

IEEE 19th International Conference on Computer Supported Cooperative Work in Design, 2015, pp.614-619.

[32] A. Ouadah, A. Hadjali, F. Nader, and K. Benouaret, "SEFAP: an efficient approach for ranking skyline web services", Springer-Journal of ambient Intelligence and Humanized Computing, Vol. x, no. x, 2018, pp.1-16.

[33] D.E. Goldberg, and J.H., Holland, Genetic algorithms and machine learning. Machine learning, 1988, Vol. 3, no. 2, pp.95-9.

[34] M. Gen, R. Cheng, Genetic Algorithms & Engineering Design, John Wiley& Sons, Inc., New York, 1997.

[35] F. Mardukhi, N. NematBakhsh, K. Zamanifar, A. Barati, QoS decomposition for service composition using genetic algorithm, Applied Soft Computing, vol. 13 no. 7, pp. 3409-3421, 2013.