A Study of Web Service Composition Framework with Advantages and Challenges

Manik Chandra* Department of Computer Science and Engineering
Institute of Engineering and Technology Lucknow, Lucknow, India 226021
* manik.chandra@ietlucknow.ac.in

***Abstract*—Assembling of many existing services to work together, is called web service composition. Selection of QoS-aware web service in composite form is a complex task. In this paper, we study the importance of web service composition and challenges. We also explain the description of web services selection problems and apply two existing classical meta-heuristic approaches to demonstrate the web service process and the effect of approaches. These approaches are Genetic algorithms and Grey wolf optimizer. Through experiments, we demonstrate the composition process, its advantages, and challenges.**

***Keywords*: Web-service selection,Web service Composition, Quality of service, Global constraints, gray wolf optimizer, Genetic Algorithm**

I. Introduction

The development of Web 2.0 facilitates the interaction of users with web pages, making it possible to deliver cloud services through the internet as SaaS (Software as a Service). A system that is based on services is known as Service-oriented architecture. A service is a unit of functionality, that exists autonomously, whose access is through a defined interface. The services available through the internet are web services.The web service can be defined as:A web service is self-described, loosely coupled, modularized, self contained and platform-independent software component which can be discovered and invoked via Internet, to implement a particular functionality [1]–[3].

Web services are developed independently and are executed in a distributed manner. Service developer develops the service in WSDL(Web Services Description Language) and publishes it along with additional information such as developer, address and other technical detail about the service in UDDI (Universal description, discovery, and Integration) that is a central service registry. Service user obtains the information and other technical details from the service registry to invoke and bind the web service through SOAP (Simple Object Access Protocol) message exchange protocol. Architecture of web service is shown in fig1. In this service oriented architecture (SOA) there are four components. These are Service itself, Service provider, Service consumer (service requester) and Central registry [4]–[6].

The important characteristics of of service oriented architecture are as follows:

- Interoperability: The service requester considers only the functionality of services and terms and conditions. Any service user having any platform can use the service independently irrespective of any platform.

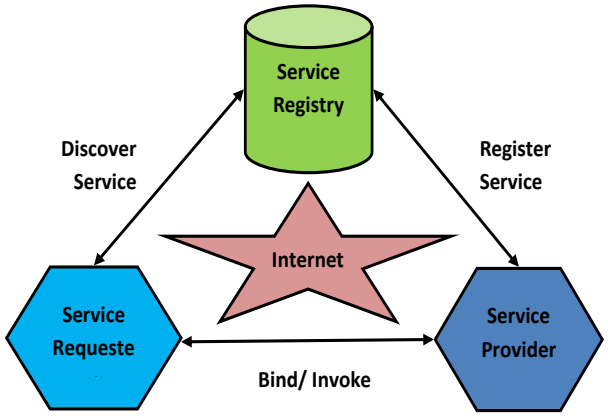

Fig. 1: Architecture of Web Services

- Loosely coupled: The services are developed and hosted on independent computers and the services and service users are situated at different locations and do not have a common shared memory. They communicate through only message passing. The services are stateless, previous information does not affect the current service.
- Abstraction: Services hide the internal details from the user. Since users see the functionality of the service they do not have any concern with the structural details of the service.
- Granularity: The atomic services are the services that have a single functionality. In the services oriented architecture most services are designed for a single functionality. The service oriented architecture allows to combine more than one services to solve the complex problems.

In this paper, we study the various composition models and challenges to be faced during the composition process.

II. Web Service Composition

Web services composition refers to the process of combining multiple web services to create a more complex and value-added service. This approach leverages the capabilities of existing services to build new applications and workflows without having to develop everything from scratch. To fulfill the users complex requirements many web services from different service provider combined to work together. These combined services are known as composite services. Therefore the composite service is to combine the functionality of several

web services and the process of developing a composite web service is called service composition[4], [5], [7].

In the composition user's complex problems are broken into small sub-problems and there is an atomic service available against each sub-problem. The service compositions can be broadly defined into three categories: (i) Static composition, (ii) Dynamic Composition and (iii) Context aware composition [8], [9].

- **Static Composition:** In the static composition, the user knows the number of sub-tasks in the problem and the set of atomic services against each task before starting the composition process.
- **Dynamic Composition:** Dynamic composition refers to the selection of web service for the composition at run time against each sub-task. Dynamic composition is more complex than static composition because at the beginning user does not know which type of services has to choose for composition. It is decided at run time as the requirement occurs.
- **Context Aware composition:** The web services selection and execution process is affected by user's preferences, quality of services and external environments. this eternal environment is known as context. The context can be explain through following sentences:
  - The context are special parameters that do not belongs to any service but make the whole process of composition more efficient and meet user's personalized requirements.
  - Context are non functional attributes which are used to describe the user's preference and external environment that their changes will affect service execution.
  - Context values are ubiquitous and often changed.

III. Composition Models

Some composition arrangements and their attribute aggregation are explained below: Here we are considering two negative (response time and latency) and two positive (availability and reliability) total four attributes/QoS parameter to explain the composition arrangements [10]: $Responsetime$ is the time period from send the request and receive a response. $Latency$ is time the server takes to process a given request. $Availability$ is the ratio between number of successful invocation to total invocation and the $reliability$ is ratio between number of erroneous messages to the total messages.

There are four important types of composition arrangements, which are Sequential, parallel, choice and loop arrangements.

The sequential arrangement is shown in Figure 2(a), there are $s_1, s_2, \therefore s_n$ n numbers of basic web services executed in sequential manner, in this arrangement the input for a service s_k, where $1 < k \leq n$ is the output of its previous service s_{k-1}. In this composition arrangement aggregation of attribute response time and latency is additive and aggregation of attribute availability and reliability is multiplicative.

[5] In case of parallel composition arrangement two or more paths are executed parallel as shown in Figure 2(b). In this composition arrangement the aggregated value of response time, latency is taken as for as maximum and the aggregated value of availability and reliability is multiplicative .

In case of choice composition arrangement, multiple different paths are available but only one path is to be selected for execution purpose. The arrangement is shown in Figure 2(c). Let there are n paths in this arrangement and probability of a path j to be selected for execution is p_j where $\sum_{j=1}^{n} p_j = 1$, the aggregated value for this arrangement has to be chosen after multiplication of corresponding probability in each component of attribute value.

In loop arrangement one or more services are executed repeatedly until a certain condition is achieved, as shown in Figure 2(d). In this composition arrangement each participating basic service, is repeated k times. The aggregated value of attribute response time, latency will be k times, and availability , reliability will be in power of k.

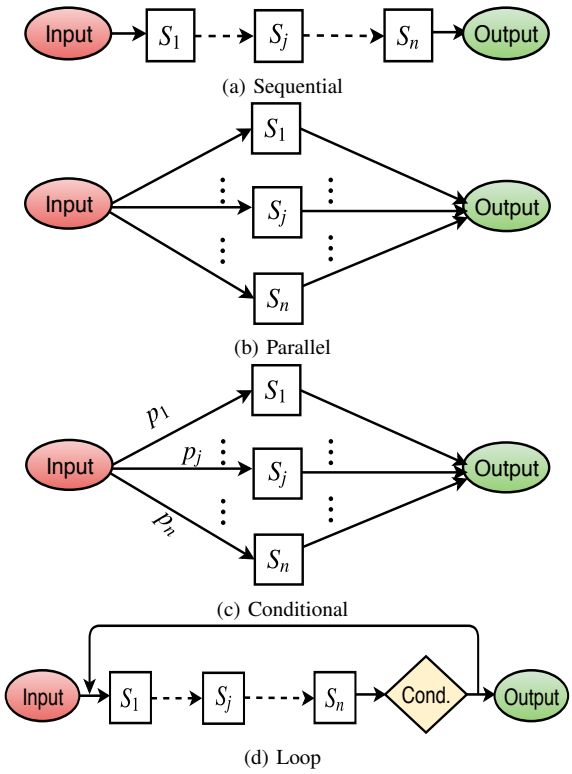

Fig. 2: Diagrammatic representation of different composition structures

Aggregated values of attributes for all types composition arrangements are given in Table I(a),(b),(c) and (d).

IV. Web Service Selection Problem Description and Formulation

A. Web Service Selection

Web service selection is the process of selecting the optimal web services for composition purposes. In this section, we provide the description as well as the mathematical modeling for the Web service composition problem. We first provide some description to set the theoretical background for problem modeling .

Table I: QoS aggregation formulas for different composition structures

Composition Structure	QoS attributes			
	Response time	Latency	Availability	Reliability
Sequential	$\sum_{j=1}^{n} \mathcal{RT}(s_j)$	$\sum_{j=1}^{n} \mathcal{L}(s_j)$	$\prod_{j=1}^{n} \mathcal{A}(s_j)$	$\prod_{j=1}^{n} \mathcal{RL}(s_j)$
Parallel	$\max\{\mathcal{RT}(s_k) : k \in [1,n]\}$	$\max\{\mathcal{L}(S_k) : k \in [1,n]\}$	$\prod_{j=1}^{n} \mathcal{A}(s_j)$	$\prod_{j=1}^{n} \mathcal{RL}(s_j)$
Choice	$\sum_{j=1}^{n} p_j * \mathcal{RT}(s_j)$	$\sum_{j=1}^{n} p_j * \mathcal{L}(s_j)$	$\sum_{j=1}^{n} p_j * \mathcal{A}(s_j)$	$\sum_{j=1}^{n} p_j * \mathcal{RL}(s_j)$
Loop	$k * \mathcal{RT}(s_j)$	$k * \mathcal{L}(s_j)$	$\mathcal{A}(s_j)^k$	$(\mathcal{RL}(s_j))^k$

We consider a universe of web service $\mathbb{S}$ which is defined as the collection of abstract service classes, i.e., $\mathbb{S} = \{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n\}$. Each abstract service class $\mathbf{S}_i$ is a set of functionally equivalent web services, i.e., $\mathbf{S}_i = \{s_i^1, s_i^2, \cdots, s_i^m\}$, here m is the number of web services in an abstract web service and s_i^j, $j = 1, \cdots, m$ denotes the j^{th} web service of class i.

Let us consider a task T which comprises n different atomic tasks, i.e., $T = \{T_1, T_2, \cdots, T_n\}$. Each atomic task T_i, $i = 1, 2, \cdots, n$ is realized by a service of a specific class.

Let $Q = \{q_1, q_2, \cdots, q_r\}$ denotes the QoS attributes. For instance, the $Q_s = \{q_1(s), q_2(s), \cdots, q_r(s)\}$ represents the r attributes of the web service s, where $q_i(s)$ denotes i^{th} attribute of the service s.

Let $W = \{W_1, W_2, \cdots, W_r\}$ denotes the preferences of users for each attribute $q_k \in Q$, where $W_k \in [0,1]$ denotes the weight of k^{th} attribute with $\sum_{k=1}^{r} W_l = 1$.

Let $C = \{C_1, C_2, \cdots, C_r\}$ be the set of constraints given by the user, where C_k is the constraint against QoS attribute k in a composite service and r is the number of constraints.

B. Web Service Composition problem description and formulation

In web service composition many web services are combined together to solve a complex task which is not possible to solve by any basic web service. A composite web service looks like a basic(single) web service and their similar QoS parameters aggregated together. Aggregated value of QoS parameters appears to the user as a basic service parameter. Hence, a composite web service is defined as a abstract representation of $CWS = \{s_1^{j_1}, s_2^{j_2} \ldots, s_n^{j_n}\}$, where $j_i \in [1,m]$.

In Figure 2(a) represents the sequential composition of of web services. A task T is divided into multiple sub-tasks. For example for first sub-task t_1, there are m web services are available and each web service is functionally equivalent. Similarly, for sub-tasks T_2, T_3,and T_4, there are m web services are available and each web service is functionally equivalent In order to compose a web service, only one web service for each sub-task have to be is selected.

In this paper, we are aiming to find a composite service $CWS = \{s_1^{j_1}, s_2^{j_2} \ldots, s_n^{j_n}\}$, where $j_i \in [1,m]$ against each sub-class of class T. Each web service $s_i^{j_i}$ of composite service must satisfy the global QoS constraint, and selected composite service must have the optimal value of global QoS from the user's point of view (preference).

In order to compute the utility of each CWS, we calculate the QoS value of its components. The QoS computation of the components depends on the composition model, i.e., sequential, parallel, conditional and loop. Composition models define the arrangement between these components of services.

C. Utility Function

As we mentioned that a sequential composition model to compute the composite service is considered, the utility function for computation for this model is explained below.

First we normalized the QoS attributes of each component of CWS. The normalization of an attribute q_k of the s is as follows:

- if q_k is positive attribute then the normalized value of attribute q_k is

$$q_k^{norm}(s) = \begin{cases} \frac{q_k(s) - q_k^{min}(s)}{q_k^{max}(s) - q_k^{min}(s)}, & \text{if } q_k^{max}(s) \neq q_k^{min}(s) \\ 1, & \text{if } q_k^{max}(s) = q_k^{min}(s) \end{cases} \quad (1)$$

- if q_k is negative attribute then the normalized value of attribute q_k is

$$q_k^{norm}(s) = \begin{cases} \frac{q_k^{max}(s) - q_k(s)}{q_k^{max}(s) - q_k^{min}(s)}, & \text{if } q_k^{max}(s) \neq q_k^{min}(s) \\ 1, & \text{if } q_k^{max}(s) = q_k^{min}(s) \end{cases} \quad (2)$$

where $q_k^{max}(s)$ and $q_k^{min}(s)$ are the maximum and minimum value of k^{th} attribute of component s.

Let $q_k^{agg}(CWS)$ is the sum of normalized values of the k^{th} attributes of each component of a CWS. The utility function, $\mathcal{U}$, for CWS is defined as:

$$\mathcal{U}(CWS) = \sum_{k=1}^{r} W_k \times q_k^{agg}(CWS) \quad (3)$$

Now using the above-mentioned description of utility function and formula of QoS computation, we formulate the

problem of finding an optimal composite service which meets the global constrains as follows.

$$\text{maximize} \quad \mathcal{U}(CWS) = \sum_{k=1}^{r} W_k \times q_k^{agg}(CWS) \tag{4}$$

$$\text{subject to} \quad q_k^{agg} \leq C_k \quad \text{if } q_k \text{ is positive attribute} \tag{5}$$

$$q_k^{agg} \geq C_k, \quad \text{if } q_k \text{ is negative attribute} \tag{6}$$

In our approach, we determine a single numeric value score derived from the values of different QoS parameters to compare one composite service (CWS) with others. We pass various combinations of basic services through our proposed algorithm to find optimal CWS. In this section, first, we explain score determination method for a single basic service and then we explain score determination for CWS followed by the mathematical formulation of underlying problem.

In the context of web services, a combination of basic web services in composite form acts as a solution, which is represented as an array of n length. Here n is the number of basic web services participating in composition. Let six subtasks are needed to complete the whole task, hence we have to select a basic web service from each six service classes. As it is, shown in figure, we select 7^{th} basic web service from service class S_1, 3^{rd} basic web service from service class S_2, 5^{th} basic web service from service class S_3, 9^{th} basic web service from service class S_4, 4^{th} basic web service from service class S_5, and 8^{th} basic web service from service class S_6. Finally, we get a solution, that is [7,3,5,9,4,8].

V. Applied Approaches for Selection

Many evolutionary algorithms based approaches are applied for the selection of QoS aware web services composition due to there wide applicability and they produce prominent results. These are population-based approximation algorithms but they give results within real-time [11]. Several In [5] authors applied modified grey wolf optimizer (MGWO) for the selection of constraint-based web service selection, in this algorithm genetic algorithm's crossover operator is added with grey wolf optimizer. In [12] authors proposed OBL based differential evolution used for the same problem. In [13] linear programming (LP) based algorithm is proposed, which provides an efficient solution to the problem. In [14], the authors proposed a genetic algorithm (GA) based approach for the selection of web service, in this approach the GA is able to scale-up when the number of concrete services increases. In [15], a genetic algorithm based on simulated annealing. In [16], used hybridization of GRASP and PR meta-heuristic techniques to solve the problem. Some other proposed methods are ant colony algorithms [17], and Artificial bee colony method (ABC) are applied [18]. In [19], the authors proposed improved artificial bee colony algorithm for selection of constraint-based web services selection and justify the obtained the results better than ABC algorithm [18]. In [20], authors proposed a new algorithm that is information entropy immune genetics algorithm (IEIGA), and proves the suitability of solving large scale service selection problem. In this paper we simulated tow approches that Genetic algorithm (GA) and Grey wolf optimizer.

A. Genetic Algorithm

Genetic algorithm is a search based Meta-heuristic algorithm that is able to solve constrained and unconstrained based optimization problems. GA algorithm takes some random solution as input that are known as initial population. It works on the concept of survival of fittest and elimination of unfit based on Darwinian Theory of evolution. Every single solution in GA is a unit of a population which is popularly known as a chromosome in term of genetic algorithm. Initially few solutions are generated for this algorithm. The fitness values of these solutions are evaluated by some fitness function. It does two main operations on existing solutions, crossover and mutation [21], [22]. The basic steps of genetic algorithms are shown in the following flowchart: The first step in this

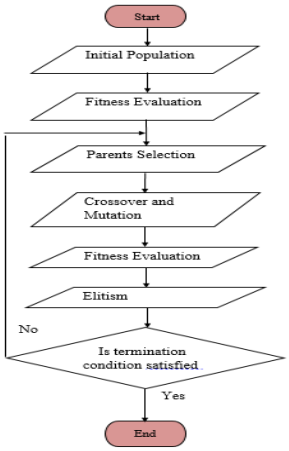

Fig. 3: Flow chart of genetic algorithm

algorithm is the initial population generation. As we previously explained that the population is the set of solutions and a solution is the combination of genes solutions are also known as a chromosome. To complete the execution of algorithm within the feasible time it is not possible to take all possible combinations in the population. So we take some specified number of combinations that are known as population size. There are many methods are available to generate the initial population like a random selection, roulette wheel selection, tournament selection etc. Since there is an objective value associated with each chromosome so the second step which is fitness evaluation is the evaluation of that objective value. In this section, we apply some penalty function which determines some penalty value for each chromosome. The penalty value is added to the existing objective value and the obtained value is known as the fitness value of that particular chromosome. This step aims to encourage the good chromosome and discourage the bad one. In the third step is the selection of chromosome to participate in the next step based on their fitness value. For the maximization, problem chromosomes are selected in the decreasing order of their fitness value and for the minimization problem, they are selected in the increasing order of their

fitness value. Step two and three are the steps of parent selection mechanism of evolutionary algorithms. The fourth step crossover and mutation is the process of creating new permutations of gens among the population in hand which yields a new set of population. Since the GA is the two-phase algorithm, so it has two different evolution operator one for crossover and 2nd for mutation. During the cross over process, we take two parents P1 = 0 0 0 0 0 0 and P2 = 1 1 1 1 1 1. Identify the crossover point and break each parent solution into two parts at the crossover point. Exchange one part of each parent with each other. In this exchange process, two new solutions are generated that are offspring (children) C1 and C2. Where C1 = 0 0 0 1 1 1 and C2 = 1 1 1 0 0 0 if the crossover point is 3. In the mutation process, there is only one parent generate a new solution by changing a gene value. Let take P1 = 0 0 0 0 0 0 as a parent and it generates a new solution C3 = 0 0 0 0 1 0 by changing a gene value at position 5th. The fifth step fitness evaluation is the determination fitness value of new population similar to step two. Sixth step elitism preservation is the process to preserve the good solutions of the previous iteration. In this process, we take the best population size solutions of previous iteration population and new population, here good solutions are those solutions that have good fitness value. The last step of this flowchart of GA is checking for termination condition if the termination condition is not satisfied then repeat the step 3 to 6 otherwise terminate the algorithm.

B. Gray Wolf Optimizer

The Gray Wolf Optimizer (GWO) is a population based meta-heuristic algorithm that is proposed by *Mirjalili et al.* in 2005 [23] for optimizing numerical problems. The algorithm is specifically based on the hunting and democratic behavior of gray wolves. Generally, the wolves dwell in the group of 5-10 and all the members are bounded to pursue the task in accordance with their assigned position in the group. In the group, the highest position is assigned to alpha wolf (α), followed by beta (β) wolf which is at second position and works as aide to alpha wolf. The beta wolf forwarded the alpha's directions to the group members and reverts their responses to alpha. The delta wolf (δ) holding third position in the hierarchy, act like a subordinate to alpha and beta. Finally, omega wolves (ω) hold the last position in the hierarchy. Another fascinating characteristic of wolves is group hunting which is accomplished in three stages; chasing, encircling, and attacking.

The algorithm starts with predetermined number of wolves where their initial positions are randomly decided. The best, the second best , and the third best positions are assigned to alpha, beta, and delta , respectively. And the remaining positions are allotted to omega wolves. In order to find the position of the prey, wolves adopt an intelligent strategy. After knowing the position of the pray, they cleverly make an effort to encircle it.

Suppose in any generation t of the algorithm, the position of the prey and the wolf is denoted by $X(t)$ and $X^p(t)$, respectively. The mathematical modeling for the encircling process is as follows [23]

$$D = | C * X^p(t) - X(t) | \tag{7}$$

$$X(t+1) = X^p(t) - A * D \tag{8}$$

Where $C = 2r_2$ and $A = 2a * r_1 - a$, r_1 and r_2 are random numbers selected from interval (0,1), and a varies between 2 to 0 [23].

In hunting process, alpha act as leader, while beta and delta play the role of subordinate. The location of the prey (optimal solution) is not known previously in most of the cases. However, it is presumed that alpha, beta and delta are superior and have some clue about location of prey and the rest wolves are entailed to reposition themselves according to alpha, beta, and delta using equations (3)-(5).

$$D^\alpha = | C_1 * X^\alpha - X |; D^\beta = | C_2 * X^\beta - X |; D^\delta = | C_3 * X^\delta - X | \tag{9}$$

$$X_1 = X^\alpha - A_1 * D^\alpha; X_2 = X^\beta - A_2 * D^\beta; X_3 = X^\delta - A_3 * D^\delta \tag{10}$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{11}$$

VI. Experimental settings and results

A. Parameter Setting

All the experiments are implemented on MATLAB 2018 (b) in the window environment on a 64 bit 3.40 GHz Intel(R) Core(TM) i7 PC with 8 GB RAM. The stopping criteria for all algorithms is set as 500 iterations and we simulated each algorithms 20 times independently .

B. Data set

The data set from repository [24] is taken in this study. Some relevant points about the data set is given as follows: the data set provides values of 9 QoS attributes for 2507 concrete web services. We have used data for 2500 web services, in order to make the problem symmetric. We taken four QoS attributes for our experimental study response time, latency are negative and availability, reliability are positive attributes.

C. Experimental results and Discussion

In order to compare the optimality of given algorithms, On the basis of fixed number abstract services n and the fixed number of basic services m in each service class $\mathbf{S}$, and all QoS have given equal weightage.

The performance of both approaches is displayed in figure4, according to the graph the GWO approach perform better in all times in the graph. The performance of any meta-heuristic algorithms is depends on following three parameter:

- Initial Population Generation: How the initial populations or selections of parents are done.
- Exploration: How well the approach can explore the good (points) parents among all populations.
- Exploitation: How good the approach can find good solutions near explored points.

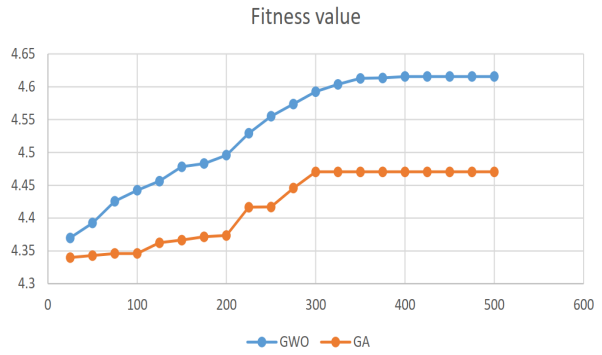

Fig. 4: Flow chart of genetic algorithm

VII. Challenges in Service composition

Important challenges in web service composition are explained as following : [25]

- Technological Challenges: Technical Diversity: Services may be implemented in different programming languages, frameworks, and protocols (e.g., SOAP, REST). Achieving interoperability among these varied technologies can require extensive middleware and communication protocol adaptation.
- Data Format Incompatibility: Different services may use diverse data formats (JSON, XML, etc.), necessitating format transformations and data mapping.
- Input-output matching: In the service composition output parameters of the previous service works as input parameters for the next. The input-output should be matched in terms of number, order, and semantics. The input-output matching in terms of number of the parameters is divided into Exact matching, Plug-In, Intersection, and Disjoint matching. Exact matching is the best matching in this matching the parameters are exactly matched. In the plug-in, the output parameters are less than the required parameters for the input of the next service. In the case of intersection input out parameters are different but some are common. In case of disjoint, there is no matching between input and output parameters.
- Semantics :Even if services perform similar functions, they may interpret inputs and outputs differently, leading to semantic inconsistencies that make integration challenging.
- Since the information in composite web services may flow between geographically distributed networks therefore there is possibility to be compromised the integrity, confidentiality of the information.

VIII. Conclusion

n this study, we presented the advantages, challenges, and problem formulation of web service selection. We used metaheuristic approaches to demonstrate the effect of various existing algorithmic approaches Genetic algorithms and Grey wolf optimizer. We have seen the different approaches deliver different results that depend on Initial population selection, proper exploration, and proper exploitation. In this study, we considered the sequential workflow of composition. Further, we described the challenges of web services composition. In the future, this work can be extended for web service composition in non-sequential workflows and dynamic composition of web services.

Acknowledgment

We are thankful to anonymous reviewers for their valuable suggestions and comments.

References

[1] Zhiying Cao, Xiuguo Zhang, Weishi Zhang, Xiong Xie, Jinyu Shi, and Haotian Xu. A context-aware adaptive web service composition framework. In *2015 IEEE International Conference on Computational Intelligence & Communication Technology*, pages 62–66. IEEE, 2015.

[2] Shankar Kambhampaty. *Service-Oriented Architecture Microservices Architecture for Enterprise, cloud, Big data and Mobile*. WILEY, 2018.

[3] Hongda Wang, Zhichun Jia, Bo Shao, Yiwen Wang, Xiyu Zhang, and Xing Xing. A web service selection model using grey wolf optimization with bwm and topsis. In *2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*, pages 1–4. IEEE, 2024.

[4] W3C Working Group, W3C Working Group, et al. Web services architecture, 2004.

[5] Manik Chandra, Ashutosh Agrawal, Avadh Kishor, and Rajdeep Niyogi. Web service selection with global constraints using modified gray wolf optimizer. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1989–1994. IEEE, 2016.

[6] Zhe Liu, Chunyang Chen, Junjie Wang, Xing Che, Yuekai Huang, Jun Hu, and Qing Wang. Fill in the blank: Context-aware automated text input generation for mobile gui testing. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1355–1367. IEEE, 2023.

[7] Manik Chandra and Rajdeep Niyogi. Web services based path guidance to rescue team alert system during flood. In *2016 Ninth International Conference on Contemporary Computing (IC3)*, pages 1–6. IEEE, 2016.

[8] San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, and Cheng-Hung Chen. Dynamic web service selection for reliable web service composition. *IEEE Transactions on services computing*, 1(2):104–116, 2008.

[9] Yuyao Xu, Lianglun Cheng, Tao Wang, and Mingzhe Ni. An improved harris hawks optimization algorithm for microservice composition and collaborative optimization. In *2024 29th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2024.

[10] Jorge Cardoso, Amit Sheth, John Miller, Jonathan Arnold, and Krys Kochut. Quality of service for workflows and web service processes. *Journal of web semantics*, 1(3):281–308, 2004.

[11] Avadh Kishor, Manik Chandra, and Pramod Kumar Singh. An astute artificial bee colony algorithm. In *Proceedings of Sixth International Conference on Soft Computing for Problem Solving: SocProS 2016, Volume 1*, pages 153–162. Springer, 2017.

[12] MA Remli, S Deris, M Jamous, MS Mohamad, and A Abdullah. Service composition optimization using differential evolution and opposition-based learning. *Research Journal of Applied Sciences, Engineering and Technology*, 11(2):229–234, 2015.

[13] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z Sheng. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421. ACM, 2003.

[14] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1069–1075. ACM, 2005.

[15] Zhi-peng Gao, Chen Jian, Xue-song Qiu, and Luo-ming Meng. Qoe/qos driven simulated annealing-based genetic algorithm for web services selection. *The Journal of China Universities of Posts and Telecommunications*, 16:102–107, 2009.

[16] José Antonio Parejo, Sergio Segura, Pablo Fernandez, and Antonio Ruiz-Cortés. Qos-aware web services composition using grasp with path relinking. *Expert Systems with Applications*, 41(9):4211–4223, 2014.

[17] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[18] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[19] Xianzhi Wang, Zhongjie Wang, and Xiaofei Xu. An improved artificial bee colony approach to qos-aware service selection. In *2013 IEEE 20th International Conference on Web Services*, pages 395–402. IEEE, 2013.

[20] Yi Que, Wei Zhong, Hailin Chen, Xinan Chen, and Xu Ji. Improved adaptive immune genetic algorithm for optimal qos-aware service composition selection in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 96(9-12):4455–4465, 2018.

[21] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

[22] Shifen Han and Li Xiao. An improved adaptive genetic algorithm. In *SHS web of conferences*, volume 140, page 01044. EDP Sciences, 2022.

[23] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, 2014.

[24] Eyhab Al-Masri and Qusay H Mahmoud. Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804, 2008.

[25] Freddy Lécué, Eduardo Silva, and Luís Ferreira Pires. A framework for dynamic web services composition. In *Emerging Web Services Technology, Volume II*, pages 59–75. Springer, 2008.