IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 21, 2019, accepted June 20, 2019, date of publication July 1, 2019, date of current version July 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2926155

Web Service Selection Using Modified Artificial Bee Colony Algorithm

MANIK CHANDRA AND RAJDEEP NIYOGI
Department of Computer Science and Engineering, IIT Roorkee, Roorkee 247667, India
Corresponding author: Manik Chandra (mchandra@cs.iitr.ac.in)

ABSTRACT Web services are a type of application software, which can be remotely accessed through the Internet. Due to the proliferating growth of web services of the same functionality, the user goes into a dilemma to select suitable service for him. In this paper, we study the web service selection (WSS) problem in a sequential composition model. We formulated the WSS as a constrained optimization problem. To solve the problem, we suggest a modified artificial bee colony (mABC) algorithm, which uses a chaotic-based opposition learning method to generate a better initial population. To improve the exploration capability of the mABC, a new search equation for employed bee phase is suggested. On the other hand, to improve the exploitation capacity of the mABC, a new search strategy, inspired by differential evolution (DE), is adopted in the onlooker bee phase. We test the mABC on synthetic web service selection problem taken from QWS dataset. To assess the relative performance of the mABC, we compare it against five other state-of-the-art algorithms. The experimental results show that the mABC is better than other existing approaches in terms of response time, latency, availability, and reliability.

INDEX TERMS Web service selection, artificial bee colony algorithm, chaotic map, *QoS* attributes.

I. INTRODUCTION

Web services are self-describing software applications that can be published, discovered and invoked across the web using standard technologies. These are reusable, inter-operable across different platform and location independent [1]. Every web service possess a core function, referred to as functional property of that web service, for which it is designed. In addition to its functional property, every web service possess some non-functional properties which are also known as *QoS* attributes [2].

Due to regular changes in complex requirement of a user, it is almost impossible to fulfill the user's requirement through single (basic) web service. Therefore, we required a set of basic services integrated in an appropriate order so that the user's request is satisfied. The process of integrating the existing web services to create a new composite web service is called web service composition. In order to design a composite web service (*CWS*) for a user's task, first the task is divided into subtasks; next a basic web service is mapped to a subtask according to its functional property; finally the mapped web services are integrated into a single composite web service. The sub tasks execution order is same as the service composition order. As there exist a number of basic services with same functional property, there may exist a set of *CWS* to perform a single task. So, the selection of an optimal *CWS* from the set of *CWS*s is called web service selection. The optimal selection is defined as: a selection is called optimal selection if the selected *CWS* is not only able to satisfy the user's requirement but also able to satisfy *QoS* aggregated values.

Since the complexity of the WSS problem increases with the increase of number of abstract services and basic services, it is not feasible to find the solution of the problem in polynomial time by applying an exact approach [3], [4]. In this regard, meta-heuristic approaches are considered (see [5], [6]) as the better approximation techniques. However, as suggested by No free lunch theorem [7], there is no specific meta-heuristic algorithm that yields better results for all types of optimization problems. Thus, the new algorithms are always welcomed in the area of web services selection.

In this paper, we propose a new meta-heuristic algorithm modified Artificial Bee Colony algorithm (mABC) for the selection of web services. It uses an opponent based initialization technique to scatter the initial solutions in the search space. Arithmetic recombination based search equation is used in employed bee phase to explore the search space. To carry out guided exploitation of existing solutions,

The associate editor coordinating the review of this manuscript and approving it for publication was Zhangbing Zhou.

the search equation of onlooker bee phase is replaced with crossover operation of DE.

In order to evaluate the performance of our approach mABC, we consider four *QoS* attributes: response time, latency, availability and reliability. We compare the results with some existing approaches: Artificial Bee Colony [8], Differential Evolution [9], Modified Grey Wolf Algorithm [10], Gbest guided ABC [11] and Improved ABC [12]. The experimental results show that the proposed mABC is better than the competitive algorithms in terms of selected performance metrics, i.e., response time, latency, availability and reliability.

The remainder of parts of the paper is organized as follows. The related work is described in Section II. A detailed description of web service composition problem along with problem formulation is given in Section III. The proposed mABC is described in Section IV. The experimental results are presented in Section V. Finally, the conclusion is drawn in Section VI.

II. RELATED WORK

In this section, we describe the existing literature on web service selection problem which is relevant for this work. In recent years web service researchers have proposed various techniques to solve the problem of web service selection. These techniques are broadly classified into three categories (a) exact (b) heuristic and (c) meta-heuristic [3], [4].

A. EXACT ALGORITHMS

In exact techniques, web service selection problem is modeled as a linear optimization using the concept of linear programming [14]. To solve the linear optimization problem, several methods are suggested. In [5], a branch and bound approach is applied to select an optimal composite web service. In [6], [13], a quasi disjunction and conjunction approach is used to determine the best service. In [14], mixed integer programming is used to solve web service selection problem. However these approaches provide an optimal composite, but they are computationally inefficient because their execution time grows exponentially with problem size.

B. HEURISTIC ALGORITHMS

In a general way, the heuristic algorithms are framed by experience for any specific optimization problem. The solution finding approach of heuristic algorithms is based on a ''trial and error'' approach within the limit of specified time. These algorithms give near optimal solution and take less time than that of exact algorithms. The solution given by heuristic algorithms is near to optimum solution but better than the guess. Hill climbing, BFS and A^* algorithms are few examples of heuristic approaches. For the detailed exposition of heuristic approaches, please refer to the [15]–[19].

C. META-HEURISTIC ALGORITHMS

Meta-heuristic algorithms are approximation algorithm and are often termed as population-based algorithms. The core concept of these algorithms is based on their population-based nature. These algorithms work with the population of initially guessed solutions and over the course of time, this population is refined using some high-level heuristics. This high-level heuristic is based on two components, i.e. exploration and exploitation. In meta-heuristic algorithms, the blending of these two components is done in such a manner that the algorithms yield a globally optimal solution without trapping into local optima. some popular meta-heuristic are: Genetic algorithms [20], Differential evolution [22], Ant colony algorithms [21], PSO [23] and ABC [24].

The reason behind their popularity is their intrinsic characteristics which are as follow: (a) simplicity: easy to implement (b) Derivation free: we do not need to check the properties of optimal function that is whether function is differentiable, continue, bounded, separable or non separable etc. (c) Problem independent: for any problem the working principle of algorithm remains the same (d) Local optima avoidance. Due to their potential and prospect, meta-heuristic algorithms are applied in various engineering problems [25], [26].

In [27], the author proposed a new algorithm Gbest-guided artificial bee colony algorithm for discrete data set environment and added time attenuation function into the composition to improve the accuracy of assessment. In [28], Liu *et al.* proposed the solution of *QoS* aware service composition problem using ABC algorithm. They suggested a parameter tuning method to design the optimal parameter setting of the ABC algorithm. This parameter tuning approach is based on the c4.5 classification algorithm. In [12], the authors have proposed a greedy neighborhood based artificial bee colony algorithm called IABC and applied to solve web services selection problem in the dynamic environment. In [29], a modified ABC algorithm is proposed in which neighbor selection is based on the concept of Euclidean distance is proposed. Further, this modified algorithm is applied to select an optimal composite web service. In [9], an opposition based differential evolution algorithm is used to solve the web service selection problem. In [30], the authors proposed swarm optimization *QoS* aware service selection approach in a dynamic environment. In [31], the authors model the fruit fly algorithm for the selection of composite web services and named web service composition based on fruit fly optimization (WS-FOA).

III. PROBLEM DESCRIPTION AND FORMULATION

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.

All the notations and abbreviations used in this paper are summarized in Table 1.

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

TABLE 1. Notations and acronym descriptions used in the paper.

Notations	Description
s	Basic web service
$\mathbf{S}$	Abstract service class $s \in \mathbf{S}$
$\mathbb{S}$	Collection of abstract service classes
m	Number of web services in a abstract service class S
n	Number of services participating in composition
CWS	Composite web service
T	Task to be completed
t	Sub task of T
C	Set of constraints
c_k	k^{th} constraint where $c_k \in C$
QoS	Quality of service
Q	Quality attributes
q	QoS Attribute $q \in Q$
w	Weight of a quality parameter

Acronym	Stands for
WSS	Web service selection
ABC	Artificial Bee Colony
IABC	Improved ABC
GABC	Gbest guided ABC
MGWO	Modified Grey Wolf Optimizer
DE	Differential Evolution
mABC	modified Artificial Bee Colony

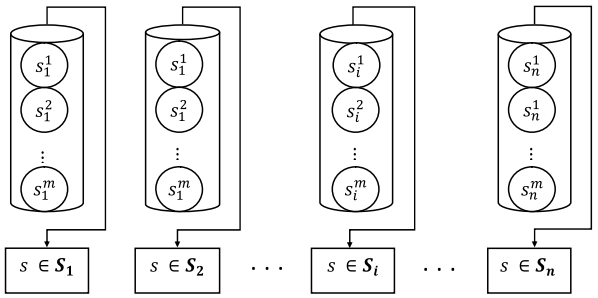


FIGURE 1. Schematic representation of service composition.

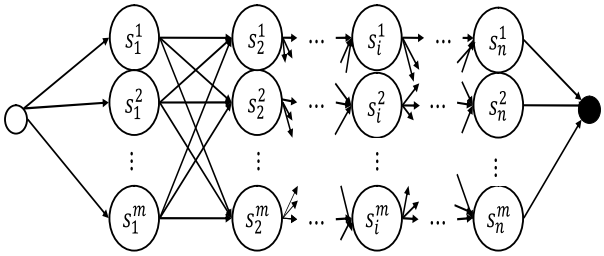


FIGURE 2. Complexity of service composition.

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.

A. WEB SERVICE COMPOSITION

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 an abstract representation of $CWS = \{s_1^{j_1}, s_2^{j_2} \therefore, s_n^{j_n}\}$, where $j_i \in [1, m]$.

The schematic representation of web service composition is given in Fig. 1.

In Fig. 1 the cylindrical shapes represents the class of web services. For example in the first cylinder, there are m web services, i.e., $s_1^1, s_1^2, \cdots, s_1^m$, and each web service is functionally equivalent. In order to compose a web service, only one web service is selected from each abstract class. The selection of web service to compose web services is illustrated in Fig. 2.

In this paper, we are aiming to find a composite service $CWS = \{s_1^{j_1}, s_2^{j_2} \therefore, s_n^{j_n}\}$, where $j_i \in [1, m]$. 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.

In this paper, we consider the sequential composition model. This method maps the aggregated QoS value of a CWS to a real number, referred to as utility value. In this model to compute the utility value, we need to perform the following two steps:

1) *Normalization*: in this step, we normalize QoS attribute values of different web service into a real number between 0 and 1.
2) *Weighting*: in this step, we multiply each normalized QoS value with its corresponding weight and add them to get the utility value of that CWS.

We consider four different types of QoS attributes for a CWS, which are given below:

- Response time: is a time span between the sending a request and receiving the response.
- Latency: is time taken to execute the request.

TABLE 2. *QoS* aggregation formulas for different composition structures.

Composition Models	QoS attributes			
	Response time	Latency	Availability	Reliability
Sequential	$\sum_{j=1}^{n} q(s_i^j)$	$\sum_{j=1}^{n} q(s_i^j)$	$\prod_{j=1}^{n} q(s_i^j)$	$\prod_{j=1}^{n} q(s_i^j)$

$j \in [1, m]$ is the index of the selected web service from the abstract service class i

- Availability: is the ratio between the number of successful invocation to total invocation
- Reliability: is the ratio of error messages to the total messages.

These attributes are categorized into two groups: (i) positive *QoS* and (ii) negative *QoS*. For Positive *QoS*, the larger value of *QoS* gives higher utility, hence, the higher value of positive *QoS* is desired. Availability and reliability are in the category of positive *QoS*. On the other hand, for negative *QoS*, the smaller value of *QoS* gives higher utility, hence, the smaller value of negative *QoS* is desired. Response time and latency belong to this category. The *QoS* computation formulas for sequential composition model is given in Table 2.

B. FORMULATION OF 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

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

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

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

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:

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

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.

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

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

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

IV. MODIFIED ARTIFICIAL BEE COLONY ALGORITHM

In this section, we describe the proposed modified artificial bee colony algorithm (mABC). To set the technical background for our proposed algorithm, we also discuss canonical ABC algorithm and its limitations.

A. AN OVERVIEW OF CANONICAL ARTIFICIAL BEE COLONY ALGORITHM

The ABC algorithm, proposed by Karaboga and Basturk [32], is a swarm based meta-heuristic algorithm, based on the foraging behavior of honey bees. According to [24], group of honey bees living together in a hive is termed as honey bee colony. Here, a colony is made up of three different types of bees: (i) employed bees, (ii) onlooker bees, and (iii) scout bees. The foraging task in bees' colony is initiated by employed bees. Each employed bee targets the location of a food source and collects the information about the nectar amount available in that food source. Thereafter, they share the information regarding nectar amount with their nest mates. After collecting the information about the food sources, an onlooker bee selects better food source to exploit it. When an employed bee finds that a food source became abandoned, the employed bee resumes its role as a scout bee. In this way, the responsibility of exploitation is carried out jointly by employed bees and onlooker bees, while the exploration is solely carried out by scout bees [32].

In mathematical modeling, a food source corresponds to a solution and the nectar amount is analogous to the quality of the solution. Further, three categories of bees indicate three different types of phases of the algorithm. The algorithm is composed of four phases, which are presented below.

1) INITIALIZATION PHASE

ABC algorithm starts with a set of solutions called population. The population size is the number of food sources (*SN*).

It solution is denotes as a D-dimensional, i.e.,

$$X_i = (x_{i1}, , \cdots , x_{ij}, \cdots , x_{iD}) \tag{7}$$

where $i = \{1, \cdots , SN\}$. Initially, the population is randomly generated constrained by variable bounds. The minimum bounds on the solutions is denoted as

$$\underline{X} = (\underline{x}_1, \cdots , \underline{x}_j, \cdots , \underline{x}_D) \tag{8}$$

$\underline{x}_j$ denotes the minimum bound on the j^{th} dimension parameter. Similarly, the maximum bound on the solution is denoted as

$$\overline{X} = (\overline{x}_1, \cdots , \overline{x}_j, \cdots , \overline{x}_D) \tag{9}$$

$\overline{x}_j$ denotes the maximum bound on the j^{th} dimension parameter.

Now, initial solutions are generated randomly by using following equation

$$x_{ij} = \underline{x}_j + \phi_{ij}(\overline{x}_j - \underline{x}_j) \tag{10}$$

where $\phi_{ij} \sim U(0, 1)$ is randomly distributed number between 0 and 1. Moreover, after initializing each solution of the population, *limit* counters for them are set to zero.

2) EMPLOYED BEE PHASE

Each employed bee searches the vicinity of current solution to get a better candidate solution for the solution in hand. The procedure of searching a candidate solution for a solution X_i is described below.

1) Create a mutant vector, say V_i, whose parameter are same as in X_i, i.e.,

$$V_i = (x_{i1}, , \cdots , x_{ij}, \cdots , x_{iD}) \tag{11}$$

2) Randomly choose a variable $j \in (1, D)$ and determine the new value v_{ij} as

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{12}$$

where $k \in (1, SN)$ and $k \neq i$, and $\varphi_{ij} \sim U(-1, 1)$ is a randomly distributed number between -1 and 1.

3) Calculate the fitness of candidate solution V_i by using following formula

$$fit_i(V_i) = \begin{cases} \dfrac{1}{1 + f_i(V_i)} & \text{if } f_i(V_i) \geq 0 \\ 1 + |f_i(V_i)| & \text{if } f_i(V_i) < 0 \end{cases} \tag{13}$$

where $f_i(V_i)$ is the objective function value for V_i.

4) Compare $f_i(V_i)$ and $f_i(X_i)$
   a) if $f_i(V_i) < f_i(X_i)$, then replace X_i with V_i and set its *limit* counter to zero.
   b) otherwise X_i remain in the population and its *limit* counter is increased by 1.

3) ONLOOKER BEE PHASE

In order to intensify the exploitation process, onlooker bees come into the picture. First, each onlooker bee perform a roulette wheel selection to choose a fitter solution from the list solutions exposed by employed bees. The quality of a solution i is determined on the basis of fitness probability, which is determined as

$$fp_i = \frac{fit_i}{\sum_{i=1}^{Sn} fit_i} \tag{14}$$

After the evaluation of the fitness probability, each onlooker bee updates the position of a solution in a similar manner as adopted by employed bees.

4) SCOUT BEE PHASE

If the position of a solution is not updated after searching a predefined number of trials, i.e., *limit*, then it is abandoned and an employed bee associated with this solution changes its role to a scout bee. And, this new scout bee replaces the abandoned solution by a randomly generated solution using Eq.(10).

B. A FEW DRAWBACKS IN CANONICAL ABC

Till date, there is no meta-heuristic algorithm that can achieve the best solution for every type of optimization problems. Thus, ABC is not an exceptional one, it also faces up some limitations. A meta-heuristic algorithm tries two to maintain a proper balance between two conflicting aspects of its performance: exploration and exploitation. The first means to the ability of the algorithm to search various known region of the search space, while the latter one corresponds to the ability of finding a better solution by applying the knowledge of solutions achieved so for and converge to the optimal as quickly as possible.

Recently, to know the advantages and limitations of the ABC, a comprehensive study is carried out in [26]. In this study, it has been found that ABC suffers from some inefficiencies. For example, it shows poor exploitation capability that leads to slow convergence as well as local optima stagnation. Due to poor exploitation, it may not be able to approximate a high-quality solution.

The reason behind this drawback is the search mechanism adopted by bees (employed and onlookers). For discovering the new solution, bees rely on only one solution that is randomly selected from the population. So, it is only 50% chances that selected solution is good in terms of fitness, whilst at the same time, the probability for bad one is also 50%. In this way, there are very fewer chances that a selected solution is better candidate solution. This uncertainty causes a delay in discovering a potential solution and leads to a slow convergence rate. Further, to generate a candidate solution, one-dimensional communication among bees is allowed, i.e., only one randomly selected dimension of a solution is mutated to generate the candidate solution (see, Eq.12). This limited communication is not enough to avoid local optima trap while dealing with multi-model problems.

Apart foregoing issues, this single dimensional information exchange among the bees causes the scalability problem. The reason is obvious as dimensions of the problem increased, information exchange remains limited and the mutation effect on high dimensional cases is diminished.

C. MOTIVATION FOR MODIFICATION

In order to overcome the aforementioned drawbacks of ABC, in the last 5-6 years, a lot of work was done. Many researchers worked on the search equation of ABC and proposed many advancements.

In order to improve the exploitation, Zhu and Kwong [11] have suggested an improved version of ABC, called global best solution guided ABC (GABC). In GABC, a new search equation for bees is suggested. In this search equation, a gbest solution is incorporated as an additional neighbor. However, after experimental study, it has been evinced that the improvement in GABC is limited in multi-model problem [33] and it gives rise to another problem called ''Oscillation phenomenon'' [34]. Further, to enhance the performance of GABC, Gao and Liu [33] have suggested an Improved version of GABC (IABC, here). In IABC, the modification is made in two steps: (i) an initialization method based on chaotic map and opposition based learning is suggested and (ii) two search equations based on crossover method of DE is suggested. Although IABC is able to prevent ''oscillation phenomenon'', but, the information exchange rate in IABC is still single dimensional (means local optima trapping is still possible).

After analyzing the pros and cons of existing algorithms, in this paper, we suggest a modified version of ABC called mABC which ensures a proper trade-off between exploration and exploitation.

D. OUR PROPOSED MODIFIED ABC

In order to design our proposed mABC, we made three changes in the canonical ABC. The changes are:

(i) initialization strategy
(ii) search equation for employed bees
(iii) search equation for onlooker bees

1) INITIALIZATION STRATEGY

In general, meta-heuristic algorithms are considered a sort of black box optimization techniques. Their end success is highly dependent on the point of start, means how the initial population is initialized? Whether the initial population well distributed in the feasible search space or not? Thus, the initialization phase of any meta-heuristic algorithm is very crucial because it directly influences the quality of the final solution and the convergence speed of the algorithm.

In order to select a better initial population, we suggest a chaotic map based opposition learning method. We have selected logistic chaotic map [35] which equation is iterated as follows:

$$ch_{k+1} = \mu * ch_k(1 - ch_k) \tag{15}$$

Algorithm 1: A Chaotic Map and Opposition Learning Based Initialization Scheme

input : SN - population size, D - dimension
$\underline{X}$ - lower bounds for decision parameters
$\overline{X}$ - upper bounds for decision parameters
$k_{max} \leftarrow 300$
output: population of size SN

1 **for** $i \leftarrow 1$ *to* SN **do** /* for each individual */
2 **for** $j \leftarrow 1$ *to* D **do** /* for each dimension */
3 $Ch_0 \leftarrow$ rand(0,1);
4 **for** $k \leftarrow 1$ *to* k_{max} **do**
5 $Ch_{k+1,j} \leftarrow 4 \times Ch_{k,j} \times (1 - Ch_{k,j})$;
6 **end**
7 $x_{ij} \leftarrow \underline{x}_j + Ch_{k,j}(\overline{x}_j - \underline{x}_j)$
8 **end**
9 **end**
10 **for** $i \leftarrow 1$ *to* SN **do** /* for each individual */
11 **for** $j \leftarrow 1$ *to* D **do** /* for each dimension */
12 $x^o_{ij} \leftarrow \underline{x}_j + \overline{x}_j - x_{ij}$
13 **end**
14 **end**
15 Evaluate all X_i and X^o_i from x_i and x^o_i
16 Combine all X_i and X^o_i
17 Select SN solutions using elitism principle
18 **return** (population of size SN)

here, $k = 1, 2, \cdots, k_{max}$ denotes the iteration counter and k_{max} is fixed to 300 [35]. The value of μ is set to 4 and the ch_k is a random number between 0 and 1. We propose an algorithm to initialize the solution. The pseudo code of the algorithm is given in Algorithm 1.

In opposition based learning (OBL), an estimated solution X and its corresponding opposite estimate X^o are considered simultaneously to cover the overall search space. The opposite estimate for every solution is calculated by equation given below

$$X^o_i = \underline{X} + \overline{X} - X_i; \quad i \in SN \tag{16}$$

here, $\underline{X}$ defines the lower bounds of decision variables and $\overline{X}$ defines their upper bounds. Taking this into account, the proposed initialization strategy is efficiently selecting the solutions from a larger range of search space by simultaneously probing original guess solutions along with their OBL counterparts.

The pseudo code of the proposed initialization strategy is given in Algorithm 1. Initially, in the first loop from step 1 to 9, the chaotic mapping based generation is used to get an estimated SN solutions X. The Eq.(15) is employed for this operation. Subsequently, in the second loop from step 10 to 14, the OBL based Eq.(16) is applied to get the another

set of SN solutions termed as opposite estimate X^o. All the solutions are evaluated and combined set is formed. At last, better SN solutions are selected using elitism principle as initialization population.

2) SEARCH EQUATION FOR EMPLOYED BEE

Employed bee is the first one who starts the search for a new potential candidate for an already existing solution. So, to make it exhaustive, it is required to carry out all dimension information exchange. In this view, we proposed a new search equation, as given below, that fulfills our goal

$$V_i = \phi_{ij} X_i + (1 - \phi_{ij}) X_k \tag{17}$$

where $i, k \in (1, SN)$ and $k \neq i$, and $\phi_{ij} \sim U(-1, 1)$ is a randomly distributed number between -1 and 1.

3) SEARCH EQUATION FOR ONLOOKER BEE

The onlooker bees are incorporated to further intensify the exploration process. Mainly, the functioning of onlooker bees is dependent on employed bees as they utilize the information gathered from them. To improve the intensification, we modified the search equation of onlooker bees in canonical ABC algorithm. It has been evident in many studies [36], [37] that involving the best solution and probing the search space around it can improve the exploitation performance. Hence, the modified search equation of onlooker bee is directed toward the best solution in the population. Basically, this equation is inspired by the principle of the DE algorithm [38] and devised as follows:

$$x_{ij} = x_{ij} + \phi_{ij}(x_{best,j} - x_{r_1,j}) + (1 - \phi_{ij})(x_{r_2,j} - x_{r_3,j}) \tag{18}$$

where, r_1, r_2, r_3 are mutually exclusive random numbers, such that; $i, r_1, r_2, r_3 \in (1, SN)$ and $r_1 \neq r_2 \neq r_3$, and $j \in (1, D)$ is randomly chosen index. Similarly, x_{best} is the best solution vector with best fitness in the current population. $\phi_{ij} \sim U(-1, 1)$ is a randomly distributed number between -1 and 1.

E. OUR APPROACH FOR SOLVING THE PROBLEM

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.

1) SOLUTION ENCODING

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].

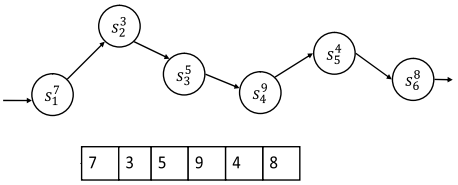


FIGURE 3. Solution encoding.

2) DETERMINATION OF FITNESS VALUE

One of the important factors for constraint based optimization problem is determination of fitness value. The fitness value measures the performance of each composite solution in search space. In constraints based optimization we add penalty to those composite solutions which do not follow the constraints. Outcome after imposing penalty on Utility value of any composite service is fitness value. In this paper the fitness value has been determined as explained in [39]. Utility value of i^{th}composite service $\mathcal{U}(CWS)_i$ is determined as explained in formula 3. Let the constraint given against j^{th} QoS attribute is c_j and the aggregated value of same QoS attribute is (q_j^{agg}). then that particular composite solution is feasible when $(q_j^{agg}) \geq c_j$ if j^{th} QoS attribute is positive or $(q_j^{agg}) \leq c_j$ if j^{th} QoS attribute is negative. then the fitness value determined as Eq.(19).

$$fit(CWS)_i = \begin{cases} 0.5 + \mathcal{U}(CWS)_i \times 0.5; & \text{if ind is feasible} \\ \mathcal{U}(CWS)_i \times 0.5 - Pn_i; & \text{otherwise} \end{cases} \tag{19}$$

where the Pn_i is penalty value for a CWS. The penalty value is determined as Eq. (20)

$$Pn_i = \begin{cases} 0 & \text{satisfied constraints} \\ \sum_{t=1}^{r} (CV_{ij})^{\alpha} \times p_j & \text{unsatisfied constraints} \end{cases} \tag{20}$$

here r is number constraints and α is severity of the penalty (here $\alpha = 2$) and $p_j \in [0, 1]$ is penalty factor with $\sum_{j=1}^{r} p_j = 1$ and CV_{ij} is the constraint violation value for the parameter q_j in Eq. (21).

$$CV_{ij} = \begin{cases} \dfrac{max(0, c_j - agg(q_j))}{cj} & \text{if } q_j \text{ is positive} \\ \dfrac{max(0, agg(q_j) - c_j)}{c_j} & \text{if } q_j \text{ is negative} \end{cases} \tag{21}$$

While $c_j \in C$ and agg_j is the aggregated value of QoS parameter j of i^{th} CWS.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The mABC, for an analysis of the relative performance, is compared against 5 existing meta-heuristic algorithms: Artificial Bee Colony(ABC) [8], Differential Evolution (DE) [9], modified grey wolf optimizer (MGWO) [10], Discrete gbest-guided artificial bee colony (GABC) algorithm [27] and improved artificial bee colony (IABC) [33] on QWS dataset [40].

We used QWS dataset [40] of real web services. The data set contains 2507 rows and 9 columns. A row represents a basic web service, and a column represents a *QoS* attribute. In our experiments, we consider 2500 rows (2500 basic web services) and 4 columns (4 attributes). The four *QoS* attributes are: response time (ms), latency (ms), availability(%) and reliability(%).

All the experiments are implemented on MATLAB 2018 (b) in the window environment on a 64 bit 3.40 GHz Intel(R) Core(TM) i7-3770 PC with 8 GB RAM. We use the following parameter settings for the mABC in the experiments. The population size (m) is set to 250, and the dimension (n) is set to 10, the number of scout bee in an iteration is at most one [8], the limit is set to ($mn/2$) [8]. For each experiment, the mABC is simulated 20 times independently. The stopping criteria for all the algorithms are the maximum number of iteration (maxItr) which is set to 500.

The c_k constraint against *QoS* attribute k is determined as described in [41], which is as Eq. (22) and (23):

- if q_k is negative attribute then

$$c_k = \lambda \times (q_k^{agg(max)} - q_k^{agg(min)}) + q_k^{agg(min)} \tag{22}$$

- if q_k is positive attribute then

$$c_k = q_k^{agg(max)} - \lambda \times (q_k^{agg(max)} - q_k^{agg(min)}) \tag{23}$$

where $q_k^{agg(max)}$ and $q_k^{agg(min)}$ are maximum and minimum possible aggregated values of k^{th} *QoS* attribute of *CWS* respectively. Here λ is strength of *QoS* constraints which is taken $\lambda = 40\%$.

A. DISCUSSION OF THE RESULTS

In order to compare the optimality of our proposed algorithms, we take the following three criteria:

(i) On the basis of fixed number abstract services n and the fixed number of basic services m in each service class **S**.
(ii) On basis of varying basic services m and fixed number of abstract services n.
(iii) On basis of varying abstract services and the fixed number of basic services m.

For all cases, we have given equal preference to each *QoS* attribute.

For the first criteria, we took the number of abstract services $n = 10$ and number of basic services $m = 250$ in service class **S**. The stopping criterion for all algorithms is maxItr $= 500$. Fig. 4(a) shows the fitness comparison of our algorithm with other algorithms. It can be observed from the figure that our algorithm obtains the maximum fitness value. While other algorithms are stagnating on their best values within 100 iterations, our approach tries to improve solution up to 200 iterations and succeed to obtain the maximum fitness value. The Fig.4(b) displays the objective values obtained by all algorithms. Objective value is the utility value as per formula 3 obtained for each composite service. The fitness value is the objective value with the penalty, hence the fitness graph and objective graph 4(b) looks similar.

The Fig.4 (c),(d),(e) and (f) are display the actual values of *QoS* attributes obtained by different algorithms. Here response time and latency are negative *QoS* attributes; ergo values of these attributes should be less. Similarly; the availability and reliability are positive attributes; therefore the values of these attributes required more. Observing the Fig.4 (c),(d),(e) and (f) of these four *QoS* attributes we found that in all cases our algorithm mABC obtains best value of *QoS* attributes too, over the other five algorithms.

If we see the fitness value curve and objective value curve see Fig. 4 (a) and (b), the performance in descending order of algorithms is mABC, IABC, GABC, MGWO, DE and ABC. But this order is getting different in subsequent curves i.e. in Fig.4 (c),(d),(e) and (f). As we see in response time curve, Fig.4(c), the GABC algorithm outperforms IABC in contrast to fitness curve. Similarly, in the latency curve, ABC algorithm succeeded to obtain less latency value over the DE and MGWO, while it obtained less fitness value than DE and MGWO.

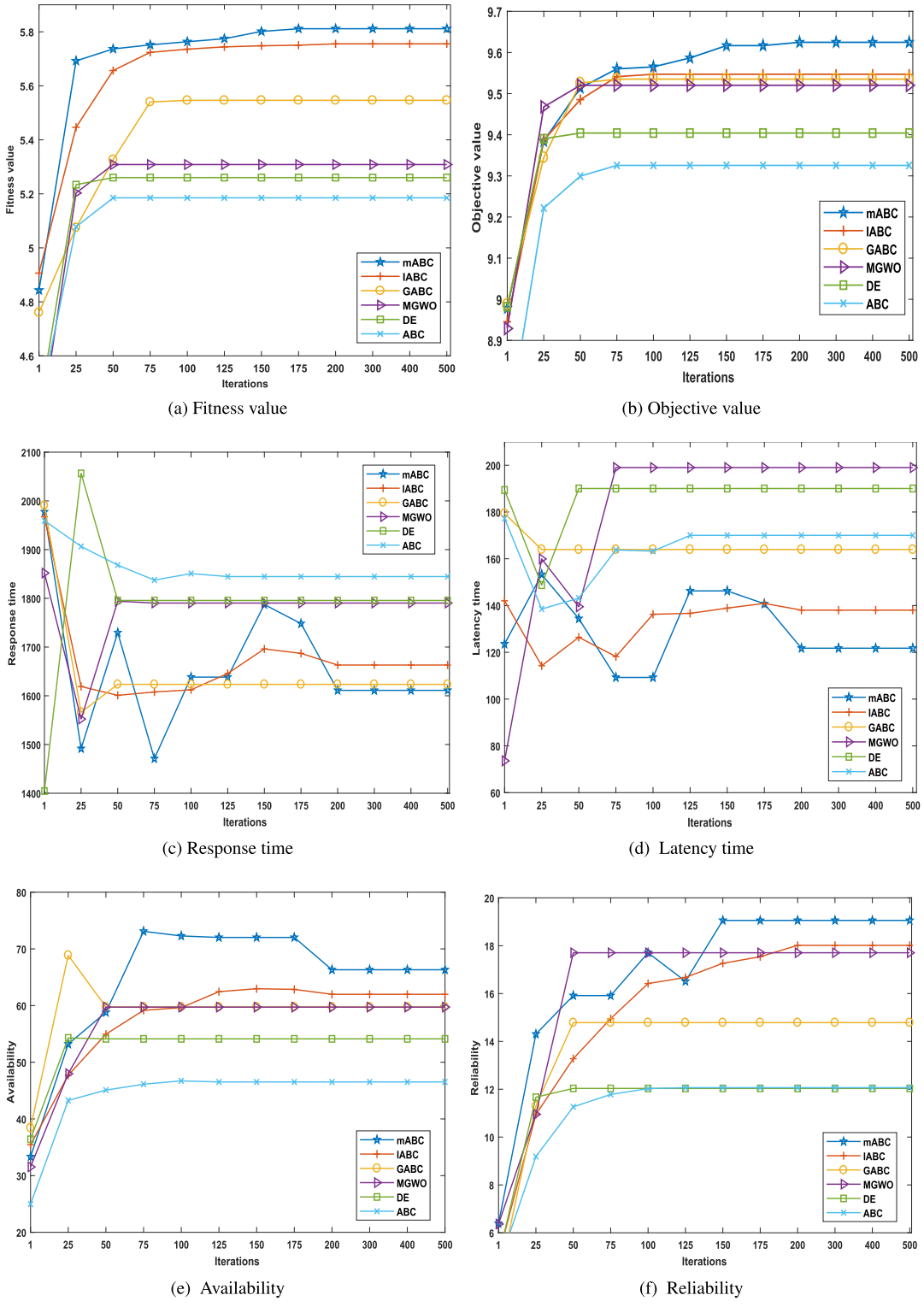
Therefore we can conclude that all algorithms are not able to achieve better value on all *QoS* attributes at their termination point, while our proposed algorithm mABC outperformed in all cases.

For the evaluation by second criteria we take varying basic services m from 50 to 250 with an interval of 50 and a fixed number of $n = 10$ and all other parameters are same as previous experiments. The fitness vs m graph in Fig. 5 which shows the fitness value obtained by all algorithms. From figure, it can be seen that mABC outperforms for $m = 100, 150, 200, 250$. But, in case of $m = 50$ the performance of IABC is slightly better than the mABC because of insufficient input of basic services.

TABLE 3. Results in terms of response time (*ms*) for varying *m* and fixed *n* = 10.

m	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
50	**2283**	1979	2118	1926	1919	2163
100	**1723**	2425	1735	1809	1829	2030
150	**1793**	2006	1794	1724	1308	2233
200	**1717**	1772	2003	1651	1959	2128
250	**1611**	1863	1623	1790	1796	1845

We also obtained the values of response time, latency, availability and reliability from this experiment which are given in Tables 3, 4, 5 and 6. The values reported in all tables are in aggregated form. Response time and latency are in



(a) Fitness value

(b) Objective value

(c) Response time

(d) Latency time

(e) Availability

(f) Reliability

FIGURE 4. Results comparison on fixed number of basic services ($m = 250$) and fix number of abstract services (n=10). (a) Fitness value. (b) Objective value. (c) Response time (ms). (d) Latency (ms). (e) Availability (%). (f) Reliability (%).

millisecond and availability and reliability are in percentage. Here we have shown these values to reveal the efficiency of all competitive algorithms in terms of obtaining aggregated QoS attributes values. For good fitness value, it is necessary that the internal components of fitness values should also be good or at least the majority of internal components should be good within permissible constraints. Here internal components of fitness are response time, latency, availability and reliability.

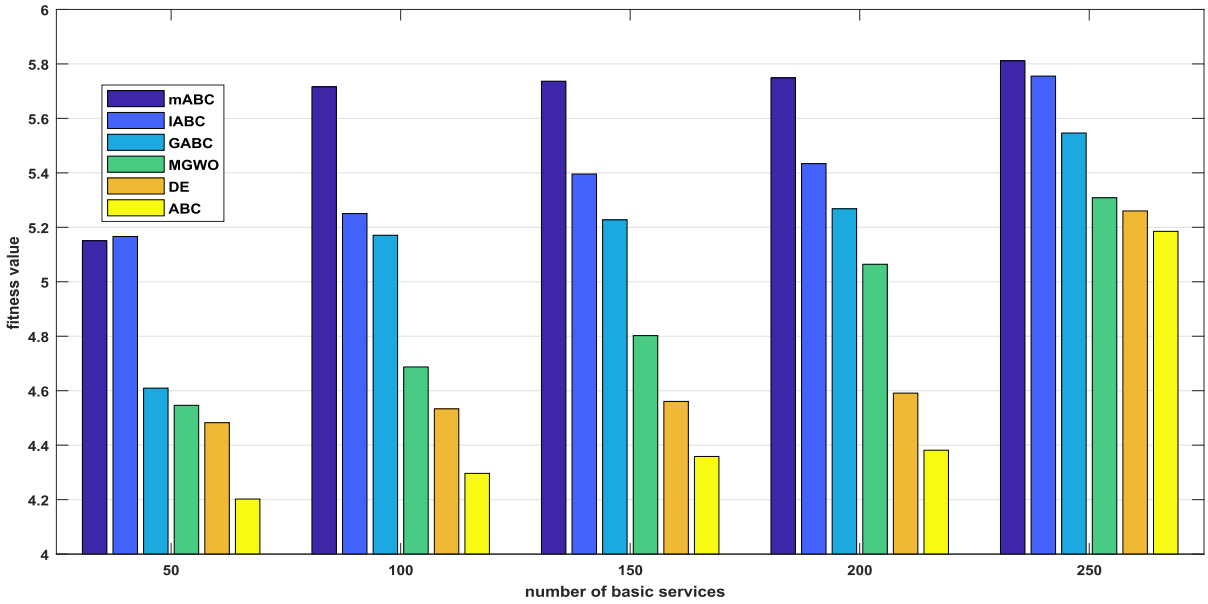


FIGURE 5. Fitness value on varying basic services (*m*).

TABLE 4. Results in terms of latency (*ms*) for varying *m* and fixed *n* = 10.

m	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
50	**090**	178	169	163	132	207
100	**158**	147	096	106	162	182
150	**128**	174	200	122	172	183
200	**119**	159	177	120	187	170
250	**122**	138	164	199	190	170

TABLE 5. Results in terms of availability (%) (*ms*) for varying *m* and fixed *n* = 10.

m	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
50	**48**	43	50	47	46	25
100	**61**	46	48	46	45	28
150	**65**	52	58	50	43	32
200	**71**	49	65	49	35	35
250	**66**	62	60	60	54	47

TABLE 6. Results in terms of reliability (%) (*ms*) for varying *m* and fixed *n* = 10.

m	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
50	**13**	08	10	08	06	04
100	**12**	11	11	10	05	04
150	**13**	11	13	10	07	05
200	**14**	11	14	11	09	05
250	**19**	18	15	18	12	12

when $m = 250$ (please refer Tables 3, 4, 5 and 6) our algorithms mABC have response time = 1611 milliseconds, latency = 122 milliseconds which are minimum among all other algorithms as these are negative attributes. and

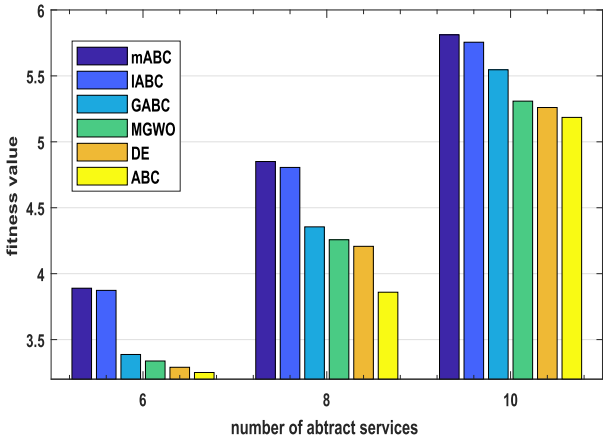


FIGURE 6. Fitness value on varying Abstract services (*n*).

availability = 66% and reliability = 19% is maximum as these are positive attributes. Thus our proposed algorithm mABC have all good values and have the best fitness value among other algorithms. Algorithm GABC, MGWO, DE and ABC have response time = 1623, 1790, 1796 and 1845 which are less than the response time of IABC, but IABC is better with other QoS attributes and have fitness value better than GABC algorithms. Similar cases are with when $m = 200$. But when $m = 150$, our algorithm mABC have the poor value of response time rather than DE algorithm but other attributes are have better values than DE and when m = 150, our algorithm has the best fitness value. Similar cases are with $m = 100$ and hence no need any explanation. when $m = 50$, good values of QoS attributes are scattered among all algorithm, and our algorithm mABC does not have any best attribute value. It maybe due to the insufficient value of m.

TABLE 7. Results in terms of response time (*ms*) for fixed $m = 250$ and varying n.

n	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
6	**1210**	1137	1483	1127	1142	1217
8	**1336**	1540	1597	1360	1466	1474
10	**1611**	1663	1623	1790	1796	1845

TABLE 8. Results in terms of latency (*ms*) for fixed $m = 250$ and varying n.

n	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
6	**61**	78	49	79	69	91
8	**120**	130	160	196	182	158
10	**122**	138	164	199	190	170

TABLE 9. Results in terms of availability (%) for fixed $m = 250$ and varying n.

n	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
6	**85**	85	64	75	65	61
8	**79**	73	53	67	58	67
10	**66**	62	60	60	54	47

TABLE 10. Results in terms of reliability (%) for fixed $m = 250$ and varying n.

n	Algorithms					
	mABC	IABC	GABC	MGWO	DE	ABC
6	**36**	30	35	29	25	18
8	**25**	17	27	17	14	24
10	**19**	18	15	18	12	12

For the evaluation by third criteria, we take a varying number of abstract services $n = 6$, 8 and 10, whereas the number of basic services $m = 250$ is fixed and all other parameters are same as in previous experiments. The comparison on basis of fitness value of this experiment is depicted in Figure 6. From the Figure, we can easily see that our algorithm outperforms other algorithms in all cases.

To reveal the efficiency of all competitive algorithms in term of obtained QoS attributes values, we plot a comparison chart on response time, latency, availability and reliability on varying abstract services (please refer Tables 7, 8, 9 and 10). When the number of abstract services is $n = 6$, fitness value obtained by IABC is approximately equal to mABC. When $n = 8$ and $n = 10$, our algorithm mABC scores best fitness value along with good QoS attributes values. Therefore, we can conclude that mABC has better performance in all the cases for QoS aware web services selection.

VI. CONCLUSION

We have considered the web service selection problem with the aim of selecting an optimal composite service. We modeled web service selection problem a combinational optimization problem which is NP-Hard. Since the problem is NP-Hard, this calls for an efficient optimization algorithm; and with this view, we suggested a modified artificial bee colony (mABC) algorithm. In this algorithm, we introduced a new kind of search procedure in employed bee phase of ABC algorithm to improve its exploration capacity. In order to ameliorate the exploitation capacity, a new search equation, inspired by DE, is incorporated in the onlooker bees phase. Moreover, to provide a better start for the searching process a chaotic opposition learning based technique is used in the initialization phase. The performance of mABC is compared against five other existing approaches. The experimental results revealed that the performance of mABC is better than all existing approaches in terms of response time, latency, availability and reliability.

In this study, we consider sequential work-flow of web services composition. In future, we would like to extend this work for non-sequential work-flow too.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. New York, NY, USA: Springer, 2004.

[2] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proc. 18th Int. Conf. World Wide Web*, Apr. 2009, pp. 881–890.

[3] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Computational intelligence based QoS-aware Web service composition: A systematic literature review," *IEEE Trans. Services Comput.*, vol. 10, no. 3, pp. 475–492, May/Jun. 2017.

[4] E. Pejman, P. R. Yousef, M. Esfahani, and A. Salajegheh, "Web service composition methods: A survey," in *Proc. Int. Multi Conf. Eng. Comput. Sci.*, vol. 1, Mar. 2012, pp. 603–607.

[5] C. Wan, C. Ullrich, L. Chen, R. Huang, J. Luo, and Z. Shi, "On solving QoS-aware service selection problem with service composition," in *Proc. 7th Int. Conf. Grid Cooperat. Comput.*, Shenzhen, China, Oct. 2008, pp. 467–474.

[6] H. Q. Yu and S. Reiff-Marganiec, "Automated context-aware service selection for collaborative systems," in *Proc. 21st Int. Conf. Adv. Inf. Syst.*, 2009, pp:193-200.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[8] A. Kishor, and P. K. Singh, "Comparative study of artificial bee colony algorithm and real coded genetic for analysing their performances and development of a new algorithmic framework," in *Proc. 2nd Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2015, pp. 15–19.

[9] M. A. Remli, S. Deris, M. Jamous, M. S. Mohamad, and A. Abdullah, "Service composition optimization using differential evolution and opposition-based learning," *Res. J. Appl. Sci., Eng. Technol.*, vol. 11, no. 2, pp. 229–234, Sep. 2015.

[10] M. Chandra, A. Agrawal, A. Kishor, and R. Niyogi, "Web service selection with global constraints using modified gray wolf optimizer," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Sep. 2016, pp. 1989–1994.

[11] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Appl. Math. Comput.*, vol. 217, no. 7, pp. 3166–3173, Dec. 2010.

[12] X. Wang, Z. Wang, and X. Xu, "An improved artificial bee colony approach to QoS-aware service selection," in *Proc. IEEE 20th Int. Conf. Web Services*, Jun. 2013, pp. 395–402.

[13] H. Q. Yu and S. Reif-Marganiec, "A backwards composition context based service selection approach for service composition," in *Proc. IEEE Int. Conf. Services Comput.*, Sep. 2009, pp. 419–426.

[14] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient Web service composition with end-to-end QoS constraints," *Trans. Web*, vol. 6, no. 2, May 2012, Art. no. 7.

[15] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for QoS-aware Web service composition," in *Proc. IEEE Int. Conf. Web Services*, Sep. 2006, pp. 72–82.
[16] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for QoS-aware Web service composition," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2010, pp. 34–41.
[17] A. Klein, F. Ishikawa, and S. Honiden, "Efficient heuristic approach with improved time complexity for QoS-aware service composition," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2011, pp. 436–443.
[18] S. A. Ludwig, "Memetic algorithm for Web service selection," in *Proc. 3rd Workshop Biologically Inspired Algorithms Distrib. Syst.*, Jun. 2011, pp. 1–8.
[19] J. Li, X. L. Zheng, S. T. Chen, W. W. Song, and D. R. Chen, "An efficient and reliable approach for quality-of-service-aware service composition," *Inf. Sci.*, vol. 269, pp. 238–254, Jun. 2014.
[20] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wessley, 1989.
[21] M. Dorigo and G. D. Caro, "Ant colony optimization: A new metaheuristic," in *Proc. Congr. Evol. Comput.*, Jul. 1999, pp. 1470–1477.
[22] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 842–844.
[23] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Jun. 2007.
[24] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes Univ., Kayseri, Turkey, Tech. Rep. tr06, Oct. 2005.
[25] A. Kishor, P. K. Singh, and J. Prakash, "NSABC: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering," *Neurocomputing*, vol. 216, pp. 514–533, Dec. 2016.
[26] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Comput.*, vol. 8, no. 2, pp. 239–287, 2019.
[27] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Appl. Intell.*, vol. 42, no. 4, pp. 661–678, Jun. 2015.
[28] R. Liu, Z. Wang, and X. Xu, "Parameter tuning for ABC-based service composition with end-to-end QoS constraints," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2014, pp. 590–597.
[29] F. Dahan, K. El Hindi, and A. Ghoneimm, "Enhanced artificial bee colony algorithm for QoS-aware Web service selection problem," *Computing*, vol. 99, no. 5, pp. 507–517, 2008.
[30] T. Zhang, "QoS-aware Web service selection based on particle swarm optimization," *J. Netw.*, vol. 9, no. 3, pp. 565, 2015.
[31] Y. Zhang, G. Cui, Y. Wang, X. Guo, and S. Zhao, "An optimization algorithm for service composition based on an improved FOA," *Tsinghua Sci. Technol.*, vol. 20, no. 1, pp. 90–99, Feb. 2018.
[32] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Nov. 2007.
[33] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Inf. Process. Lett.*, vol. 111, no. 17, pp. 871–882, 2011.
[34] W. F. Gao, S. Y. Liu, and L. L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1011–1024, Jan. 2013.
[35] W. Gao, S. Liu, and L. Huang, "A global best artificial bee colony algorithm for global optimization," *J. Comput. Appl. Math.*, vol. 236, pp. 2741–2753, May 2012.
[36] H. Faris, I. Aljarah, M. A. Al-Betar, and S. Mirjalili, "Grey wolf optimizer: A review of recent variants and applications," *Neural Comput. Appl.*, vol. 30, no. 2, pp. 413–435, Jul. 2016.
[37] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2013.
[38] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[39] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition," *J. Intell. Manuf.*, vol. 29 no. 8, pp. 1773–1792, Dec. 2018.
[40] E. Al-Masri and Q. H. Mahmoud, "QoS-based discovery and ranking of Web services," in *Proc. IEEE 16th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2007, pp. 529–534.
[41] Q. Wu and Q. Zhu, "Transactional and QoS-aware dynamic service composition based on ant colony optimization," *Future Gener. Comput. Syst.*, vol. 29, no. 5, pp. 1112–1129, Jul. 2013.

MANIK CHANDRA received the B.Tech. degree in computer science and engineering from the Institute of Engineering and Technology, Lucknow, India, in 1998, and the M.Tech. degree in computer science from UPTU, Lucknow, in 2007. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, IIT Roorkee, Roorkee, India. His research interests include web services, distributed computing, and optimization.

RAJDEEP NIYOGI received the Ph.D. degree in computer science and engineering from IIT Kharagpur, India, in 2004. He is currently an Associate Professor with the Department of Computer Science and Engineering, IIT Roorkee, Roorkee, India. His research interests include automated planning, multiagent systems, algorithmic game theory, distributed systems, and applications of logic and automata theory. He is a member of ACM and ERCIM.