QoS-aware web service composition using discrete meta-heuristic algorithms

Sunil Kumar Mahendrakara, Chandrashekar Jatothb,c, G R Gangadharanb,*, Rajkumar Buyyad

aNational Institute of Technology Rourkela, Odisha, India
bInstitute for Development and Research in Banking Technology, Hyderabad, India
cKoneru Lakshmaiah Education Fundation, Hyderabad, India
dUniversity of Melbourne, Australia

Abstract

Quality of Service (QoS)-aware web service composition is one of the challenging problems in service-oriented computing. The seamless proliferation of web services makes difficult to find an optimal web service during composition that satisfies the requirements of a user. QoS-aware web service composition problems are solved using continuous optimization algorithms with a suitable encoding scheme to adopt continuous optimization to discrete optimization. In this paper, we propose a fuzzy encoding scheme where a solution is represented as a fuzzy matrix and the composition of web services is obtained using the fuzzy matrix and the global QoS attributes are computed. Discrete Fuzzy Bat Algorithm (DFBA), Discrete Fuzzy Cuckoo Search (DFCS) and Discrete Fuzzy Particle Swarm Optimization (DFPSO) are developed by using our proposed fuzzy solution encoding. We present an empirical comparison of performance of DFBA, DFCS, and DFPSO on a set of real world web services, to select a best web service composition that optimizes QoS attributes.

Keywords: Web service composition, Quality of service (QoS), Discrete optimization, Meta-heuristics, Bat algorithm, Cuckoo search, Particle swarm optimization, Fuzzy encoding.

*Corresponding author

Email addresses: sunil.mahendrakar19@gmail.com (Sunil Kumar Mahendrakar), chandrashekar.jatoth@gmail.com (Chandrashekar Jatoth), GRgangadharan@idrbt.ac.in (G R Gangadharan), rbuyya@unimelb.edu.au (Rajkumar Buyya)
1. Introduction

Web Services are defined as software applications identified by a uniform resource identifier (URI), whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts [1]. There are several web services available on the Internet [2]. They gained further popularity with the emergence of Cloud computing that provides an easy-to-use and setup environment for hosting applications as (web) services on a pay-as-you-go basis [3]. However, these services as individual entities often cannot satisfy a user’s request to perform a complex task. In order to satisfy a requested task by a user, multiple web services may need/require to be composed [4, 5]. Let us consider a travel plan where a flight ticket needs to be booked, a hotel has to be reserved, tour guides have to be hired and many other services need to be composed in order to satisfy the user requirements. The choice of the service to use depends on the non-functional properties or quality of service (QoS) attributes. For composite services, there are different service providers with similar functionalities, which have different QoS attributes. The QoS attributes determine the values of non-functional attributes such as cost, response time, throughput, availability, and reliability of a service. These QoS attributes apply to both service provider and service composition with global QoS constraints.

Web service composition involves selecting a service from the pool of available web services for each subtask and combining them together to build a desired composite service. The seamless proliferation of web services makes service selection and service composition with diverse QoS attributes under multiple user constraints as a multi-objective optimization problem, characterized to a NP-hard problem [6]. The objective of QoS-aware web service composition is to select a best composition which optimizes the QoS attributes of the composite service satisfying the user’s constraints [7]. Due to the discrete nature of web service composition problem, an appropriate solution encoding scheme is used in association with continuous optimization algorithms to solve
QoS-aware web service composition. Most of the existing approaches to solve QoS-aware web service composition which use nature-inspired continuous optimization techniques [8, 9, 10] encode solution as a vector of integers having a dimension equal to the number of subtasks where each entry of the vector contains an index of the corresponding candidate web service selected to perform that subtask. The inherent drawbacks of encoding solution as a vector of integers involve performing different operations for updating the vector to reflect the new and better solution [11, 12]. We consider the influence of neighborhood of solutions for updating the subtask, whereas vector encoding scheme considers only the best solutions. Hence, the probability of being trapped into local optima gets increased [13]. The influence on the other subtask solutions is calculated using fuzzy set. With this concept, more than one subtask solutions can affect the others to some extent. Apart from this, the fuzzy encoding scheme considers shifting and mutation to improve the convergence speed and avoids the trap in local minima.

This paper proposes to solve QoS-aware web service composition using a fuzzy solution encoding scheme applied to Bat Algorithm (BA) [14], Cuckoo Search (CS) [15], and Particle Swarm Optimization (PSO) [16]. The proposed fuzzy encoding scheme transforms the discrete optimization problem into a continuous optimization problem and converges quickly compared to the encoding scheme with a vector of integers [17] for continuous optimization algorithms. To the best of our knowledge, no such fuzzy encoding scheme has been used along with BA, CS, and PSO to solve QoS-aware web service composition.

The salient contributions of this paper are: (i) A formal model describing QoS-aware web service composition (ii) A proposal of discrete fuzzy bat algorithm (DFBA), discrete fuzzy cuckoo search (DFCS), and discrete fuzzy particle swarm optimization (DFPSO) and (iii) A comparison of vector of integers encoding scheme of PSO, CS, and BA with fuzzy encoding scheme of PSO, CS, and BA. The results illustrate the improved convergence and efficiency of fuzzy scheme over integer encoding scheme.

The rest of the paper is organized as follows: Related work is discussed
in Section 2. Section 3 introduces BA, CS and PSO along with rudimentary fuzzy theory. Further, Section 3 describes the multi-objective algorithms such as NSGA-II, $E^3$-MOGA and GDE3. A formal model for QoS-aware web service composition is presented in Section 4. Our proposed DFBA, DFCS, and DFPSO are presented in Section 5. Performance evaluation and results are illustrated in Section 6 followed by concluding remarks in Section 7.

2. Related Work

Since QoS-aware web service composition is a NP-hard problem, different techniques have been extensively researched. Different approaches for web service composition were proposed in literature [5]. For optimal solution of web service composition, integer linear programming was employed by Zeng et al. [18]. Branch and bound method was used for modeling web service composition as a multi-dimension multi-choice knapsack problem by Yu et al. [19]. Since the computation time for these methods grows exponentially with the problem size, they are not feasible if we have a large number of services and an optimal/near-optimal solution is needed in a reasonable amount of time for real world applications. Hence to reduce the computation time, web service selection problem is transformed into resource constrained project scheduling problem which can be solved via different techniques such as greedy selection, discarding subsets, bottom-up approximation, and pattern-wise selection as proposed in [20]. In the said approaches, as the number of subtasks and available candidate services increase, it becomes impractical to use them since either the computation time required is more or the solution is not optimal / near-optimal. To avoid these issues, Berbner et al. [21] proposed a mixed integer programming formulation solved through IP relaxation, random replacement of the assignments of subtasks, and simulated annealing. Another approach to decrease the computation time is by pruning the search space using different methods. Chen et al. [22] developed a partial selection technique taking the advantage of pruning candidates by QoS-based comparison operator and dominance relationships that are
defined among candidates and work-flows. Since QoS attributes can be affected by the unpredictability of networks, it is important to monitor and re-optimize if QoS values are unacceptable or do not satisfy the constraints of the user. Hence we need efficient optimization methods for not only finding the near-optimal value in such cases but also satisfying the constraints.

Canfora et al. [7] solved web service composition using GA where each genome is encoded as a vector of integers having the dimension equal to the number of subtasks. Further, each index corresponds to a subtask and has a value equal to the index of the selected candidate service. Similar solution encoding schemes are used by [23, 24, 25]. However, GAs have some inherent shortcomings including slow rate of convergence and premature convergence in local optimum due to fixed and predetermined crossover and mutation rates. Yu et al. [26] proposed an improved tree-based genetic programming (GP) approach to solve the problem of QoS-aware web services composition. To avoid premature convergence and speed up the convergence process, it applies an adaptive strategy to the crossover and mutation parameters of the standard genetic programming approach. However, in GP, the parameters are tuned by trail and error.

Yin et al. [27] proposed a hybrid multi-objective discrete PSO algorithm which combines genetic operators and PSO. In [27], the particle updating strategy is designed by introducing a crossover operator which is based on the exchange of candidate services. Zhang et al. [28] proposed to solve QoS-aware web service selection using PSO by encoding the solution as an array of integers where each dimension corresponds to the candidate service selected. Ludwig et al. [8] proposed a new hybrid-PSO (a combination of PSO and Kuhn-Munkres algorithm) that handles several workflow request simultaneously. Zhao et al. [9] developed an improved discrete immune optimization algorithm based on PSO with an improved local best first strategy for candidate service selection. However, the existing PSO approaches consider only one best solution. So, this consideration increases the probability of being trapped into the local optima [13] and it is not good at diversification.
Generally, the solutions for QoS-aware web service composition use the encoding scheme as a vector of integers. In meta-heuristic algorithms, different operations are performed on the solution for updating the solution after each iteration and have been redefined to make such encoding schemes. In contrast to the said methods, our proposed fuzzy encoding scheme does not require the operators to be redefined. The proposed fuzzy encoding scheme converges quickly and obtains better composition when compared to the scheme of vector of integers \cite{17} for continuous optimization problems. The fuzzy scheme transforms the problem into a continuous problem allowing better exploration of the search space.

3. Background Concepts

This section introduces the concepts of standard BA, CS, and PSO, together with the basics of fuzzy theory. Further, this section describes the multi-objective algorithms such as NSGA-II, E3-MOGA and GDE3. These continuous optimization algorithms obtain near-optimal solutions for QoS-aware web service composition using an encoding scheme, that makes these algorithms discrete.

3.1. Bat Algorithm (BA)

BA is a metaheuristic algorithm \cite{14} based on the behavior of Microbats that track prey/foods using the capability of echolocation. Bats utilize echolocation, in order to detect prey/food, avoid obstacles, and locate their roosting crevices in the dark. These bats release a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Some of the bats utilize the constant-frequency signals for echolocation, while others use the frequency signals to sweep through about an octave. Their signal bandwidth depends on the species, and often increased by using more harmonics.

The BA uses the following idealized rules \cite{14}:
Each bat flies randomly with velocity $v_i^t$ and position $x_i^t$ with a fixed frequency $f_{min}$, a varying wavelength $\lambda$ (or frequency) and a loudness $A_0$ to search for prey/food. The bats automatically adjust the $\lambda$ of their emitted pulses and adjust the pulse emission rate ($r$) in the range of $[0, 1]$, relying on the proximity of their target.

Although the loudness can vary in various ways, we assume that the loudness varies from a maximum (positive) $A_0$ to a minimum constant value $A_{min}$.

In this algorithm, each bat is associated with a velocity $v_i^t$ and a location $x_i^t$ at an iteration $t$ which represents the current iteration in a D-dimensional solution space. The current best solution among all bats is $x_*$. The position of the remaining bats are updated in each iteration by using equations [1][2] and [3].

1. $f_i = f_{min} + (f_{max} - f_{min})\beta$  
2. $v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i$  
3. $x_i^t = x_i^{t-1} + v_i^t$

where $\beta$ in the range $[0, 1]$ is a random vector drawn from a uniform distribution. Initially each bat is randomly assigned a frequency range which is drawn from $[f_{min}, f_{max}]$ and $f_i$ is varying frequency. A new solution for each bat is generated locally using a random walk.

$$x_{new} = x_{old} + \epsilon A^t$$
where $\epsilon$ is a random number drawn from $[-1, 1]$ and $A^t = \langle A^t_i \rangle$ is the average loudness of all the bats at this step. Once a bat has found its prey, the loudness ($A_i$) usually decreases and the rate ($r_i$) of pulse emission increases.

$$\begin{align*}
A^t_{i+1} = \alpha A^t_i, \\
r^t_i = r^0_i [1 - \exp(-\gamma t)]
\end{align*}$$  \hspace{1cm} (5)

where ($\alpha$) and ($\gamma$) are constants. The range of the initial loudness ($A_i$) and the rate ($r_i$) of pulse emission is chosen close to $[1, 2]$ and $[0, 1]$ respectively. ($\alpha$) is alike to the cooling factor of a cooling schedule in the simulated annealing [29].

For any $0 < \alpha, \gamma < 1$:

$$A^t_i \rightarrow 0, r^t_i \rightarrow r^0_i \text{ as } t \rightarrow \infty$$  \hspace{1cm} (6)

The loudness ($A_i$) and the emission rate ($r$) are updated for each iteration and the loudness can be chosen as any value of convenience.

### 3.2. Cuckoo Search (CS)

Generally, CS is based on the brood parasitism of some cuckoo species such as Ani and Guira cuckoos. Female cuckoos use the nests of other birds to lay their eggs. Cuckoos have a remarkable ability to match their eggs in color, texture, and size of the host’s eggs. If the eggs of cuckoo are discovered, they are either thrown by the host bird or the nest is abandoned. Based on the reproductive strategy of cuckoos, CS algorithm has been developed [15].

The idealized rules for CS are as follows:

(i) for each iteration, each cuckoo lays one egg, and randomly dumps the egg in a selected nest;

(ii) The best nests with high quality of eggs will carry over to the next generations;

(iii) The number of available host nests is fixed, and the egg laid by a cuckoo is found by the host bird with a probability $p_a \in [0, 1]$ [15]. While generating a
new solution for the cuckoo $i$ say $x_i^{(t+1)}$, a Lévy flight is performed

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Lévy}(\lambda) \quad (7)$$

where $\alpha > 0$ and the product $\oplus$ means entry-wise multiplication. The Lévy flight is essentially a random walk with a step size drawn from a Lévy distribution, which is a distribution of the sum of $N$ independently and identically distributed random variables whose Fourier transform takes the following form:

$$F_N(k) = \exp[-N|k|^{\beta}] \quad (8)$$

The inverse gives us the actual distribution which does not have analytical forms, except for few cases. Hence we can express the inverse as an asymptotic series, and it’s leading-order approximation for the flight length results in a power-law distribution which is heavy-tailed.

### 3.3. Particle Swarm Optimization (PSO)

PSO [16] is a meta-heuristic technique inspired by the behavior of birds flocking and fish schooling. In PSO, the individual is represented as a particle and the population is represented as a swarm. Each particle is randomly positioned in the global search space. Each particle represents a candidate solution and is associated with position, velocity, and memory vector. Each particle has a position and moves with a velocity and stores the best solution it finds. For each iteration, it evaluates fitness, position, and velocity of each particle and flying in the search space towards the local and global solutions. Each individual particle comprises with the current position ($x_{i+1}$), the previous best position ($x_i$) and the velocity ($v_{i+1}$). The position of each particle is updated using the updated particle velocity, evaluated by the following equations:

$$v_{i+1} = \omega v_i + \phi_1 \beta_1 (p_i - x_i) + \phi_2 \beta_2 (p_g - x_i) \quad (9)$$

$$x_{i+1} = x_i + v_{i+1} \quad (10)$$
where $x_i$ is the position of the particle and $v_i$ is the velocity of the particle. $\omega$ is the inertial weight. $\phi_1$ determines the influence of knowledge of individual particle and $\phi_2$ determines the knowledge of the group. Thus, $\phi_1$ and $\phi_2$ balance between individual knowledge and the group’s knowledge. $\beta_1$ and $\beta_2$ are uniformly generated random numbers bound between some $[\beta_{\text{min}}, \beta_{\text{max}}]$. $p_g$ is the group’s best position so far and $p_i$ is the particle’s best position so far.

3.4. Fuzzy Theory

Fuzzy systems use possibility theory to handle uncertainty in their reasoning process [30]. A fuzzy set is a fundamental part of every fuzzy system. Mathematically, a set is defined as a finite, infinite or countably infinite collection of elements. Each element is either a member of a set or not. But in fuzzy systems, the element can be a partial member of the set. Each fuzzy set $A$ has several pairs (countable or uncountable) as $(x, \mu_A(x))$, where $x$ is a set and $\mu_A$ is called the membership function which represents the degree of truth or compatibility with the set. Fuzzy systems involve of fuzzification (the process of making crisp values fuzzy) and defuzzification (the process of finding crisp values from the fuzzy system’s output) [31].

During fuzzification, crisp variables are mapped to fuzzy sets using membership functions. Each fuzzy set is associated with a membership function which can be constructed using any method: exact, heuristic and meta-heuristic, such as triangular, Gaussian, Sigmoid or any other function. The values of the membership function must be restricted between $[0, 1]$ where 0 means that it is not a member and 1 means that it is fully a member of the fuzzy set and the membership function should be unique. During defuzzification, the fuzzy system’s output is mapped to the crisp values using defuzzification methods [32].

3.5. Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II is one of the most popular multi-objective optimization methods, based on fast non-dominated sorting approach, fast crowded distance estimation and simple crowded comparison operator [33].
The general procedure of NSGA-II is as follows:

Initial population: The population is initialized based on the range of the problem and constraints if any.

Non-dominated sort: Once the population is initialized, it is sorted based on non-domination in each Pareto front. Each individual in each Pareto front is assigned a rank called fitness value. Along with the rank, the crowding distance is calculated for each individual to measure how close an individual is to its neighbours.

Selection: The individuals are selected using a binary tournament selection with the crowded-comparison operator.

Genetic Operators: It performs simulated binary crossover and polynomial mutation to generate real coded GA.

Recombination and selection: In recombination, an offspring population and the current generation population are combined and the individuals of the next generation are set by the selection process. The new generation is filled by each front subsequently until the population size exceeds the current population size.

3.6. $E^3$-MOGA

$E^3$-MOGA is a multi-objective genetic algorithm, proposed by [34] that provides equal quality solution sets, assessing the trade-off among different solutions.

In $E^3$-MOGA, a fitness value of an individual is calculated based on its density and domination rank. The individual density specifies how many individuals provide the similar objective values as the other individuals. The domination rank specifies how the individual outperforms the other individuals. The domination rank encourages a fitness value, while the density discourages a fitness value. $E^3$-MOGA maintains a set of individuals with higher fitness values, which is called elite population, and they evolve their genes across
generations through the use of genetic operations such as crossover and mutation.

3.7. GDE3

Generalized Differential Evolution (GDE3), is an extension of Differential Evolution (DE) to provide global optimization with an arbitrary number of objectives functions and constraints [35]. In case of a problem having a single objective and without constraints GDE3 resembles like the original DE. GDE3 improves the multi objective problems by giving a better distributed solution. The third version of GDE extends the DE/rand/1/bin method with M objectives and K constraint functions with Crowding Distance which approximates the crowdedness of a vector in its non-dominated set [33].

It handles any number of M objectives and K constraints, including the cases where M = 0 (constraint satisfaction problem) and K = 0 (unconstrained problem). The original DE is a special case of GDE3.

In GDE3, the selection is based on the following rules [35]:

1. In the case of infeasibility of both vectors with constraint violation space, the trial vector is selected if it is weakly dominating the old vector, else the old vector is selected.

2. In the case of infeasible and feasible vectors, the feasible vector is selected.

3. In the case of feasibility of both vectors, the trial vector and the old vector are compared. If the trial vector is weakly dominated by the old vector in the objective function space, the trial vector is selected otherwise the old vector is selected. If neither of vectors dominate each other, then both the vectors are selected for the next generation.
4. Modelling of QoS-aware web service composition

Let the user requested task be performed by the composition of \( \{t_1, t_2, \ldots, t_n\} \) subtasks, and for simplicity, each subtask \( t_i \) can be performed by \( \{s_{i1}, s_{i2}, \ldots, s_{im_i}\} \) which is the set of available web services, where \( m_i \) is the number of services available for each subtask \( t_i \). The binding of web service \( s_{ij} \) for performing subtask \( t_i \) can be represented as a matrix \( X \) having dimension \( n \times m \).

\[
X_{ij} = \begin{cases} 
1 & \text{if } s_{ij} \text{ is selected for subtask } t_i \\
0 & \text{otherwise} 
\end{cases} 
\]  

(11)

where \( 1 \leq i \leq n \) and \( 1 \leq j \leq m_i \).

Each subtask \( t_i \) can be performed by only one service \( s_{ij} \) selected from \( \{s_{i1}, s_{i2}, \ldots, s_{im_i}\} \). Each service \( s_{ij} \) has four QoS attributes: Throughput (T), Availability (A), Reliability (R), and Response Time (RT). Let this set of attributes be denoted by \( Q \).

QoS attributes can be categorized as positive attributes and negative attributes. Positive attributes denote the higher user utility with higher values. Negative attributes denote the lower user utility with higher values [36, 37]. For example, throughput, availability, and reliability are positive attributes and cost and response time are negative attributes. The non-functional properties of web services are described by its QoS attributes. The list of QoS attributes and their description is described in Table 1. The normalizing process for QoS attributes is as follows:

\[
U_q(x) = \begin{cases} 
1 & \text{if } q^{max} = q^{min} \\
\frac{q^{max} - x}{q^{max} - q^{min}} & \text{if } q \text{ is negative} \\
\frac{x - q^{min}}{q^{max} - q^{min}} & \text{if } q \text{ is positive} 
\end{cases} 
\]  

(12)
where $q^{\text{max}}$ and $q^{\text{min}}$ are the maximum and the minimum values of the QoS attribute $q$ for all candidate services respectively.

The global QoS attributes are the aggregate of QoS attributes of the selected services for each subtask $t_i$, $1 \leq i \leq n$. The best composition can be obtained by maximizing the fitness value of the global QoS according to the user’s preference.

The user’s preference is expressed as weights $W_i$ where $i$ is the QoS attribute. For instance $W_{\text{throughput}} = 0.2$ and $W_{\text{Availability}} = 0.1$ means that throughput is twice as important as availability.

Let $Agg_q$ be the aggregate functions of QoS attributes which computes the global QoS of the composition. The normalizing function $U_q$ defines the values of the attribute that are more useful to the user [37, 38]. Thus the goal of the multi-objective optimization is to find a set of compromise solutions $X$ that minimizes the global fitness function, given as follows (similar to the scheme used by [46]):

$$
\text{Min} \quad F(X) = \begin{cases} 
  f_1(x), f_2(x), f_3, \ldots, f_n(x) \\
  s.t \quad x \in \omega \end{cases}
$$

(13)

Therein, $f(n) : \omega \leftarrow \mathbb{R}^n$ is vector values function, $n > 2$ objectives, $\omega \in \mathbb{R}^n$ is the feasible solutions set. The computation of global QoS is determined by

<table>
<thead>
<tr>
<th>S.No</th>
<th>QoS attributes Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Price (Pr)</td>
<td>The amount needed to pay by user for requesting a service.</td>
</tr>
<tr>
<td>2</td>
<td>Throughput (Th)</td>
<td>The possible number of service invocations in a given time (measured in invokes/sec).</td>
</tr>
<tr>
<td>3</td>
<td>Availability (Av)</td>
<td>The frequency of a service is available during the user request (measured in percentage).</td>
</tr>
<tr>
<td>4</td>
<td>Reliability (Re)</td>
<td>The correctly responded request probability within the maximum expected time (measured in percentage).</td>
</tr>
<tr>
<td>5</td>
<td>Response time (Rt)</td>
<td>The time taken to serve a request once the request is received. (measured in milliseconds).</td>
</tr>
</tbody>
</table>
recursively applying the aggregate functions to the building blocks that form the structure of the composition \[7\]. For example, if services are executed in sequence, the global response time would be the sum of all individual candidate services response times. If the services are executed in parallel, then the global response time would be the highest response time of individual services. We have considered the tasks to be sequential in the composition plan. The aggregation functions for each of the QoS attribute are represented as follows:

\[
\begin{align*}
\text{AggPr} &= \sum_{i=1}^{n} Pr(s_{ij}) \\
\text{AggTh} &= \min_{i=1}^{n} Th(s_{ij}) \\
\text{AggAv} &= \prod_{i=1}^{n} Av(s_{ij}) \\
\text{AggRe} &= \prod_{i=1}^{n} Re(s_{ij}) \\
\text{AggRt} &= \sum_{i=1}^{n} Rt(s_{ij})
\end{align*}
\]

where \(s_{ij}\) is the service selected for subtask \(t_i\). The aggregation functions for each QoS attribute are described in Table 2, where \(P_i\) represents the probability of branch \(i\) and \(k\) is the number of loops.

<table>
<thead>
<tr>
<th>QoS attribute</th>
<th>Sequence</th>
<th>Parallel</th>
<th>Loop</th>
<th>Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price (Pr)</td>
<td>(\text{AggPr} = \sum_{i=1}^{n} Pr(s_{ij}))</td>
<td>(\text{AggPr} = \sum_{i=1}^{n} Pr(s_{ij}))</td>
<td>(\text{AggPr} = \sum_{i=1}^{n} Pr(s_{ij}) \times P_i)</td>
<td>(\text{AggPr} = k \times \sum_{i=1}^{n} Pr(s_{ij}))</td>
</tr>
<tr>
<td>Throughput (Th)</td>
<td>(\text{AggTh} = \min_{i=1}^{n} Th(s_{ij}))</td>
<td>(\text{AggTh} = \min_{i=1}^{n} Th(s_{ij}))</td>
<td>(\text{AggTh} = P_i \times k \times \prod_{i=1}^{n} Th(s_{ij}))</td>
<td>(\text{AggTh} = P_i \times \prod_{i=1}^{n} Th(s_{ij}))</td>
</tr>
<tr>
<td>Availability (Av)</td>
<td>(\text{AggAv} = \prod_{i=1}^{n} Av(s_{ij}))</td>
<td>(\text{AggAv} = \prod_{i=1}^{n} Av(s_{ij}))</td>
<td>(\text{AggAv} = P_i \times \prod_{i=1}^{n} Av(s_{ij}))</td>
<td>(\text{AggAv} = P_i \times \prod_{i=1}^{n} Av(s_{ij}))</td>
</tr>
<tr>
<td>Reliability (Re)</td>
<td>(\text{AggRe} = \prod_{i=1}^{n} Re(s_{ij}))</td>
<td>(\text{AggRe} = \prod_{i=1}^{n} Re(s_{ij}))</td>
<td>(\text{AggRe} = P_i \times \prod_{i=1}^{n} Re(s_{ij}))</td>
<td>(\text{AggRe} = P_i \times \prod_{i=1}^{n} Re(s_{ij}))</td>
</tr>
<tr>
<td>Response time (Rt)</td>
<td>(\text{AggRt} = \sum_{i=1}^{n} Rt(s_{ij}))</td>
<td>(\text{AggRt} = \sum_{i=1}^{n} Rt(s_{ij}))</td>
<td>(\text{AggRt} = P_i \times \sum_{i=1}^{n} Rt(s_{ij}))</td>
<td>(\text{AggRt} = P_i \times \sum_{i=1}^{n} Rt(s_{ij}))</td>
</tr>
</tbody>
</table>

The main objective of single objective optimization is a scalarization based approach, which can yield the convex optimization of the objective functions,
i.e. the relation between two potential solutions. On the contrary, the multi-objective optimization algorithms can find the multiple solutions that approximate the Pareto Front (PF) set, maintaining a good solution diversity. Hence, the solutions of multi-objective optimization are determined by a PF set. Let the objectives be $\min (f_1(x), f_2(x), f_3, \ldots, f_n(x))$, $p = [1, 2, \ldots, n]$ and $f_p(x) \in F(X)$. The PF set is defined as follows:

The decision vector $V \in \mathbb{R}$ is said to dominate another vector $w \in \mathbb{R}(v < w)$ if and only if:

$$\forall p \in P : f_p(v) \leq f_p(w) \land \exists p \in P : f_p(v) < f_p(w)$$

A solution $x \in \mathbb{R}$ is a pareto optimal if and only if:

$$\forall x \in \beta, \forall p \in P, f_p(x^*) = f_p(x) \exists p \in P, f_p(x^*) < f_k(x)$$

where $\beta$ represents the feasible Pareto Optimal Set (POS) that comprises of all pareto optimal solutions. The PF is defined as the set of all objective function values corresponding to the solutions in POS.

$$PF = f_1(x), f_2(x), \ldots, f_n(x)/x \in POS$$

5. Proposed Methods

This section presents discrete fuzzy bat algorithm (DFBA), discrete fuzzy cuckoo search (DFCS), discrete fuzzy particle swarm optimization (DFPSO) algorithms.

5.1. Discrete Fuzzy Bat Algorithm

Let $T = \{t_1, t_2, \ldots, t_i\}$, $i \in 1, 2, \ldots, n$ be a collection of subtasks $t_i$. Each subtask is resolved by available services $t_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\}$. Then the fuzzy relation between the tasks and the available services are expressed as follows:

$$S = \begin{pmatrix}
    s_{11} & s_{12} & \cdots & s_{1n} \\
    s_{21} & s_{22} & \cdots & s_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_{m1} & s_{m2} & \cdots & s_{mn}
\end{pmatrix}$$

(14)

Here $s_{ij}$ represents the degree of membership of the $i^{th}$ element $t_i$ in the task set and $j^{th}$ element $s_{ij}$ in the available services $t_i$ to the relation $S$. 

16
\[ s_{ij} = \mu_R(t_i, s_j), \quad i \in 1, 2, \ldots, m, \quad j \in 1, 2, \ldots, n \]  \hspace{1cm} (15)

where \( \mu_R \) is membership function. The value of \( s_{ij} \) represents the degree of membership so that the task \( t_i \) is resolved by the service \( s_j \) in the feasible solution. Here each service \( s_{ij} \) is a fuzzy set, of which a subtask \( t_i \) is a partial member with some membership value \( u_{ij} \) where \( 1 \leq i \leq n \) and \( 1 \leq j \leq m \).

Hence, for each subtask \( t_i \), the location (solution) of each bat can be represented as \([u_{i1}, u_{i2}, \ldots u_{im}]\). Here, for all subtasks, the solution can be represented as:

\[
X = \begin{pmatrix}
  u_{11} & u_{21} & \cdots & u_{n1} \\
  u_{12} & u_{22} & \cdots & u_{n2} \\
  \vdots & \vdots & \ddots & \vdots \\
  u_{1m} & u_{2m} & \cdots & u_{nm}
\end{pmatrix}
\hspace{1cm} (16)
\]

Accordingly, the element of the matrix \( X \) must satisfy the following conditions:

\[ u_{ij} \in [0, 1], i \in 1, 2, \ldots, m, \quad j \in 1, 2, \ldots, n \]  \hspace{1cm} (17)

and

\[ \sum_{j=1}^{m} u_{ij} = 1, \quad i \in 1, 2, \ldots, m, \quad j \in 1, 2, \ldots, n \]  \hspace{1cm} (18)

Each bat is associated with a location and a velocity which is encoded as a set of real variables, that represents the location of bats. A fuzzy encoding scheme is proposed in which the location (solution) of each bat is represented as a fuzzy matrix. The velocity of each bat is represented as:

\[
V = \begin{pmatrix}
  v_{11} & v_{21} & \cdots & v_{n1} \\
  v_{12} & v_{22} & \cdots & v_{n2} \\
  \vdots & \vdots & \ddots & \vdots \\
  v_{1m} & v_{2m} & \cdots & v_{nm}
\end{pmatrix}
\hspace{1cm} (19)
\]

where \( v_{ij} \in \mathbb{R} \).

The position matrix value of \( u_{ij} \) may violate the constraints in the equation 17.
and equation 18 after a location update (or some iterations). In order to satisfy the constraints, the following operations are performed:

$$\text{if } u_{ij} < 0, u_{ij} = 0$$ (20)

$$X_{\text{normalize}} = \begin{pmatrix}
\frac{u_{11}}{\sum_{j=1}^{m} u_{1j}} & \frac{u_{21}}{\sum_{j=1}^{m} u_{2j}} & \cdots & \frac{u_{n1}}{\sum_{j=1}^{m} u_{nj}} \\
\frac{u_{12}}{\sum_{j=1}^{m} u_{1j}} & \frac{u_{22}}{\sum_{j=1}^{m} u_{2j}} & \cdots & \frac{u_{n2}}{\sum_{j=1}^{m} u_{nj}} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{u_{1m}}{\sum_{j=1}^{m} u_{1j}} & \frac{u_{2m}}{\sum_{j=1}^{m} u_{2j}} & \cdots & \frac{u_{nm}}{\sum_{j=1}^{m} u_{nj}}
\end{pmatrix}$$ (21)

In order to obtain the crisp values from the said fuzzy representation (defuzzification), we take the maximum of \([u_{i1}, u_{i2}, \ldots, u_{im}]\) for each subtask \(t_i\) and assign the subtask \(t_i\) to \(s_{ij}\) if \(u_{ij}\) is maximum. The binding matrix in equation 11 can be obtained by setting \(u_{ij}\) with the maximum value to “1” and all others to “0”. The global fitness function is computed using equation 13. This process is applied for all proposed approaches.

In DFBA, initially, a population of the bat having a location as in equation 10 is randomly initialized and normalized using equation 21. Each bat has a randomly generated frequency between \([f_{\text{min}}, f_{\text{max}}]\), pulse rate \(r_i\), and loudness \(A_i\). The binding matrix as equation 11 is obtained by defuzzification, and the global fitness is computed using equation 13. In each iteration, the bat with the maximum fitness is selected as the global best. Frequency, velocity and location of each bat is updated by using equations 1 to 3 and the location is bound using equation 20 and normalized using equation 21. DFBA is described in Algorithm 1.

5.2. Discrete Fuzzy Cuckoo Search

We represent each nest/solution as a fuzzy matrix as in equation 16. Initially, a population of \(n\) host nests as in equation 10 is randomly initialized
Algorithm 1: Discrete Fuzzy Bat Algorithm for QoS-aware web service composition

Randomly initialize the bat population with fuzzy location matrix $x_i$ ($i = 1, 2, ..., n$) as in equation 16 and fuzzy velocity matrix $v_i$ as in equation 19.

Bound $x_i$ using equation 20.

Normalize $x_i$ for each bat using equation 21.

Define pulse frequency $f_i$ at $x_i$.

Initialize pulse rates $r_i$ and the loudness $A_i$.

while $t < \text{Max number of iterations or stopping criterion}$ do

for each bat $B_i$ do

Generate new solution by adjusting the frequency, and updating velocity and location [equation 1 to 3];

Bound using equation 20 and normalize using equation 21;

if rand $> r_i$ then

Select a solution among the best solutions;

Generate a local solution around the selected best solution;

Bound using equation 20 and normalize using 21;

end

Generate a new solution by flying randomly;

Obtain the binding matrix by defuzzifying $x_i$;

Calculate fitness by using equation 13;

if rand $< A_i$ & $Util(x_i) > Util(x_\star)$ then

Accept the new solution;

Increase $r_i$ and reduce $A_i$;

end

end

Rank the bats and find the current best $x_\star$;

end

and normalized using equation 21. The best solution among initial solutions is found. In each iteration, new solutions are generated by a random walk using step size drawn from a Lévy distribution keeping the current best. Some nests are replaced by constructing new solutions/nests. The binding matrix as in equation 11 is obtained by defuzzification, and the global fitness is computed using equation 13. The nests are ranked, and the current best is found. DFCS is described in Algorithm 2.

5.3. Discrete Fuzzy Particle Swarm Algorithm

The proposed fuzzy encoding scheme can be used for PSO by representing the position (solution) of each particle as a fuzzy matrix as in equation 16. Initially,
Algorithm 2: Discrete Fuzzy Cuckoo Search for QoS-aware web service composition

Randomly generate initial population of n host nests \(x_i\) as in equation 16, bound using equation 20 and normalize \(x_i\) using equation 21;

while \((t < \text{Max-generations})\) or (stop criterion) do

Get a cuckoo randomly/generate a solution by lévy flight;

Bound using equation 20;

Normalize the solution using equation 21;

Obtain the binding matrix by defuzzifying \(x_i\);

Evaluate its fitness \(F_i\);

Choose a nest among n (say, \(j\)) randomly;

if \(F_i > F_j\) then

Replace \(j\) by the new solution;

end

A fraction \((p_a)\) of worse nests are abandoned and new ones / solutions are built/generated and normalized using (13);

Keep best solutions (or the nests with quality solutions);

Rank the solutions and find the current best;

end

the population of particles location as in equation 16 and the fuzzy velocity matrix \(v_i\) as in equation 19 are randomly initialized. Then, we normalize \(x_i\) for each particle using equation 21. The binding matrix as in equation 11 is obtained by defuzzification, and the global fitness is computed using equation 13. In each iteration, the particle with the maximum fitness is selected followed by velocity updation as in equation 9 and location updation as in equation 10. DFPSO is described in Algorithm 3.
Algorithm 3: Discrete Fuzzy PSO for QoS-aware web service composition

Randomly initialize the population with fuzzy location matrix \( x_i \) \((i = 1, 2, ..., n)\) as in equation 16 and fuzzy velocity matrix \( v_i \) as in equation 19;

Bound \( x_i \) using equation 20;

Normalize \( x_i \) for each particle using equation 21;

while \( t < \) Max number of iterations or stopping criterion do

for each particle \( P \) do

Calculate fitness by using equation 13;

Update \( p_i \) and \( p_g \);

Adapt velocity of the particle using equation 9;

Update the location of the particle using equation 10;

Bound \( x_i \) using equation 20;

Normalize \( x_i \) for each particle using equation 21;

end

end

6. Performance Evaluation

In this paper, we proposed DFBA, DFCS, and DFPSO algorithms to handle QoS-aware web service composition. The paper applies the concepts of a fuzzy encoding scheme for QoS-aware web service composition on a dataset of real world web services [39]. In this section, we verify the performance of our proposed method for web service composition and compare the performance of the vector of integers encoding scheme of BA, CS, and PSO against our proposed fuzzy encoding scheme.

6.1. Experimental Setup

All algorithms are implemented in Matlab R2013a. The QWS Dataset v2 [39] is used, and the algorithms are evaluated on a computer system with a processor of Intel (R) Core (TM) i5 2.60GHz and 8 GB of memory. The parameters tuned are as follows: the population size was 50 for all algorithms, \( p_a \) of the DFCS and CS was 0.25, and \( \alpha \) was set to 1. For DFBA and BA, \( \alpha, \gamma, r_i \), and loudness \( A_i \) were set to 0.9, 0.1, 0.9 and 0.9 respectively. The \( f_{min} \) and \( f_{max} \) were set to 0 and 1. For DFPSO and PSO with \( \omega = 0.72984, \phi_1 = 1.4962, \) and \( \phi_2 = 1.4962 \) were used. For NSGA-II, crossover probability and
mutation probability values set to 0.9 and 0.1. The distribution indexes for both crossover and mutation operators were 20. Simulated polynomial mutation and binary crossover were applied (based on \[33, 40, 41\]). For GDE3 algorithm, we set \(CR = 0.75\) and \(F = 0.25\) (obtained from \[42, 43\]). For \(E^3 – MOGA\), crossover probability is 0.7, mutation probability is 0.01. All these values are adopted empirically after several experiments. In order to check the distribution of QWS data set, we performed Shapiro-Wilk test. The results revealed normal distribution for the given data set. Table 3 presents the list of algorithms (with name, abbreviations, and references) that we compared with our proposed methods. The number of iterations is set to 1000 for all experiments. To validate the fuzzy encoding scheme, a vector of integers representation is used where each dimension corresponds to a subtask and its entry as the indices of candidate services, similar to the scheme used by \[17, 28\]. If solutions are encoded as a vector of integers, then the operators have their usual meaning for BA, CS, and PSO. If fuzzy solution encoding is used, then the operators are extended to work on matrices. Matrices are added and subtracted element-wise. If a constant is multiplied with the matrix, all elements of the matrix are multiplied by the constant. Additionally, in the fuzzy encoding scheme, the solution matrix is bound using equation 20 and normalized using equation 21.

As visual observations of the Pareto Front (PF) are not sufficient to decide the effectiveness of the approaches, and to compare the approaches exactly, an objective evaluation is required \[46\]. We have evaluated the performance of our proposed approaches using the following metrics:

<table>
<thead>
<tr>
<th>Algorithm Name</th>
<th>Abbreviations</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bat Algorithm</td>
<td>BA</td>
<td>[14]</td>
</tr>
<tr>
<td>Particle swarm optimization</td>
<td>PSO</td>
<td>[28]</td>
</tr>
<tr>
<td>Cuckoo Search</td>
<td>CS</td>
<td>[15]</td>
</tr>
<tr>
<td>Generalized Differential Evolution</td>
<td>GDE3</td>
<td>[44]</td>
</tr>
<tr>
<td>Evolutionary multi-objective algorithm</td>
<td>(E^3 – MOGA)</td>
<td>[45]</td>
</tr>
<tr>
<td>Nondominated Sorting Genetic Algorithm II</td>
<td>NSGA-II</td>
<td>[40]</td>
</tr>
</tbody>
</table>
• Hyperarea/ Hypervolume based comparisons
• Spreading-based comparisons and
• Set Coverage-based comparisons

We have tested and compared our proposed approaches with PSO, BA, CS, NSGA-II, DGE3, and MOGA for 2-objectives: maximization of availability and minimization of response time. Similarly, we have tested and compared our proposed approaches for 3-objectives: maximization of availability, maximization throughput, and minimization of response time.

6.2. Performance Results

We have considered the following QoS attributes in our experiments: price, throughput, availability, reliability, and response time. The list of QoS attributes and their description are mentioned in Table 1. All the attribute values of candidate services are normalized using equation 12. All the attributes are considered equally important, and hence the weights were set to 0.20. Since all the algorithms are non-deterministic, 30 independent runs were performed for each algorithm.

In Figure 1, we have illustrated the balanced solution obtained by our proposed algorithms and other compared algorithms (PSO, BA, CS, MOGA, GDE3, NSGA-II) with the objective functions $f_1$ and $f_2$ (where $f_1$ represents the positive QoS attribute (availability) and $f_2$ represents the negative QoS attribute (response time). Figures 1 and 2 represent 2-objective optimization problem (response time and availability) and 3-objective optimization problem (response time, availability, and throughput) respectively. The problem scenario comprises of 30 abstract services and each abstract service has 100 candidate services. As observed in Figure 1, each algorithm produced slightly different and not identical solutions due to the stochastic nature of the algorithms and their characteristics. From Figure
we observed that most of the solutions obtained by our proposed approaches are in PF. A good solution should have a balance between positive and negative QoS attributes, i.e., the solutions with high availability, high throughput, high reliability, low price, and low response time. For instance, PSO, BA, and CS provide numerous optimal solutions with high availability and high throughput. Similarly, MOGA, GDE3, and NSGA-II yield optimal solutions with high availability, and high throughput. But some of the solutions are dominated by NSGA-II. DFCS, DFPSO, and DFBA yield numerous good solutions with high availability, high throughput, low response time, and low price.

![Figure 1: Solutions obtained by DFCS, DFBA, DFPSO, MOGA, NSGA-II, GDE3, CS, BA, and PSO. The problem scenario comprises of 30 abstract services and each abstract service has 100 candidate services and 2-objectives (response time and availability) are considered.](image)

From Figure 2, we observed that MOGA, GDE3, and NSGA-II yield optimal solutions with high availability, low response time, and high throughput. But some of the solutions are dominated by NSGA-II and GDE3 algorithms. PSO, BA, and CS provide numerous op-
timal solutions with high availability, high throughput and low re-
response time. GDE3, DFPSO, DFBA, DFCS algorithms are the best
candidates for solving the QoS optimization problem.

Figure 2: Solutions obtained by DFCS, DFBA, DFPSO, MOGA, NSGA-II,
GDE3, CS, BA, and PSO. The problem scenario comprises of 30 abstract ser-
vices and each abstract service has 100 candidate services and 3-objectives (re-
sponse time, availability, and throughput) are considered.

6.2.1. Hyperarea/hypervolume-based comparisons

The Hyperarea (HA) [47] metric considers the dimension of the
volume dominated in the objective space. In a two-dimensional, this
metric is represented as follows:

\[ H = \{ \sum_{i} S_i \mid x_i \in A \} \]

where \( A \) is the non-dominated front under evaluation and \( S_i \) represents the area dominated by the solution \( x_i \). The computation of area \( S_i \) with regard to a reference point (RF) is composed of the maximum value for each objective function. A maximum composition value for hyperarea/ hypervolume represents the higher portion of the PF covered. Hence, a PF with a higher HV value is considered.

The statistical box-plots (mean, dispersion, and outliers) about the hyper-volume procured for the given approaches over 30 independent runs are illustrated in Figs. 3–11 (two-objective problem) and Figures 12–20 (three-objective problem).

The algorithm with a high median value finds a good approximation set, whereas the algorithm with a low dispersion value and a low number of outliers represent a stable/robust set (the solutions are same on different runs). Based on the experiment results, the considered approaches provide a similar HV mean value of 2.14 for all the 2-objective problem. Thus, it is difficult to make a clear difference among the algorithms used in the experiment. Figures 3–11 illustrate a scenario where a composite service comprises of 25, 50, 75, and 100 abstract services, while for each abstract service, the number of candidate services ranges as 25, 50, 75, and 100. From Figures 3–11 and Figures 12–20 we observe some visible differences in HV dispersion. PSO, BA, CS, and NSGA-II algorithms are slightly stable (yields the lowest dispersion) in some scenarios for 2-objective problem and are not stable in some scenarios for 2 and 3-objective problems. MOGA and GDE3 algorithms are not stable for the both objective problems. NSGA-II, DFPSO, DFBA algorithms are slightly stable for 2 and 3-objective problems. DFCS algorithm yields the good approximate
set solutions for both 2 and 3-objective problems.

Figure 3: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for PSO algorithm. For each algorithm, the blue line inside the box represents median and the box height represent dispersion. Blue circles are the outliers. The algorithm with high median value finds a good approximation set. A low dispersion value, small box and a low number of outliers represent a stable/robustness algorithm (For interpretation of the references to color in these Figures, the reader is referred to the web version of the article.)
6.2.2. Spreading-based comparisons

The Spread ($\Delta$) [47], also known as diversity metric, measures the extent of spread achieved among the obtained solutions. The main goal is finding a set of solutions which span the entire Pareto-optimal region.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{p-1} |d_i - \bar{d}|}{d_f + d_l + (p - 1)d}$$  \hspace{1cm} (23)

where $d_i$ is the Euclidean distance between consecutive solutions in the non-dominated set of obtained solutions, $\bar{d}$ is the average of all distances $d_i, d_f, d_l$, assuming that there are $P$ solutions on the best
Figure 5: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for CS algorithm.

\[ \text{PF. If } d_i \text{ has a large variance, then } \Delta \text{ can be greater than one. The uniform solutions of all distances } d_i \text{ are equal to } \bar{d} \text{ and therefore } \Delta \text{ is zero.} \]

Figures 21, 22, 23, and 24 illustrate the average spreading values (30 independent runs are considered) for the considered approaches applied to two-objective problem. An algorithm with lower values are considered as efficient. This method cannot be used for three and above objective problems. Figures 21, 22, 23, and 24 present a scenario where a composite service comprises of 25, 50, 75, and
Figure 6: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for NSGA-II algorithm.

100 abstract services, while for each abstract service, the number of candidate services ranges as 25, 50, 75, and 100. From Figures 21, 22, 23, and 24, we observed that our proposed approaches DFBA and DFCS yield better solutions spreading (lower △) values compared to other approaches.

6.2.3. Set Coverage-based comparisons

The Set Coverage (C) problem [47] is a binary coverage problem. It compares two solutions of two different algorithms. The computation is done by by considering two PFs to be compared each other as
Figure 7: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for MOGA algorithm.

follows:

\[ C(A, B) = \frac{| \{ y \in B : \exists x \in A : x < y \} |}{|B|} \]  \hspace{1cm} (24) 

where A and B are the two PFs of two algorithms to be compared and \( \leq \) represents the weak dominance relation of one algorithm over the other algorithm. The function (.,.) associates the ordered pair \( C(A, B) \) to the [0,1] interval. If the value \( C(A, B) \) equal to 1 indicates that all the solutions in B are dominated by the front A. As opposite, \( C(A, B) \) equal to 0 indicates that none of the solutions in B is dominated by the PF A. The best algorithm has a higher value for \( C(A, B) \).
and a lower value for $C(B, A)$.

Figures 25, 26, 27, 28 illustrate the Set Coverage box-plots for 2-objective problem with 25 candidate services having 25, 50, 75, and 100 abstract services. The low standard deviation and higher average values of $C(A, B)$ show the better performance for algorithm B. In Figure 25, each rectangular box represents a set of Set Coverage values for PSO, BA, CS, MOGA, NSGA-II, GDE3, DFPSO, DFBA, and DFCS algorithms. In Figure 25, the first row first rectangular box represents the set coverage values of compared algorithms such
Figure 9: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for DFPSO algorithm.

as 1-(PSO-BA), 2-(PSO-CS), 3-(PSO-MOGA), 4-(PSO-NSGA-II), 5-(PSO-GDE3), 6-(PSO-DFPSO), 7-(PSO-DFBA), 8-(PSO-DFCS), the first row second box: 1-(BA-PSO), 2-(BA-CS), 3-(BA-MOGA), 4-(BA-NSGA-II), 5-(BA-GDE3), 6-(BA-DFPSO), 7-(BA-DFBA), 8-(BA-DFCS), the first row third box: 1-(CS-PSO), 2-(CS-BA), 3-(CS-MOGA), 4-(CS-NSGA-II), 5-(CS-GDE3), 6-(CS-DFPSO), 7-(CS-DFBA), 8-(CS-DFCS), the second row first box: 1-(MOGA-PSO), 2-(MOGA-BA), 3-(MOGA-CS), 4-(MOGA-NSGA-II), 5-(MOGA-GDE3), 6-(MOGA-DFPSO), 7-(MOGA-DFBA), 8-(MOGA-DFCS), the second row sec-
Figure 10: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for DFBA algorithm.

ond box: 1-(NSGA-II-PSO), 2-(NSGA-II-BA), 3-(NSGA-II-CS), 4-(NSGA-II-MOGA), 5-(NSGA-II-GDE3), 6-(NSGA-II-DFPSO), 7-(NSGA-II-DFBA), 8-(NSGA-II-DFCS), the second row Third box: 1-(GDE3-PSO), 2-(GDE3-BA), 3-(GDE3-CS), 4-(GDE3-MOGA), 5-(GDE3-NSGA-II), 6-(GDE3-DFPSO), 7-(GDE3-DFBA), 8-(GDE3-DFCS), the third row first box: 1-(DFPSO-PSO), 2-(DFPSO-BA), 3-(DFPSO-CS), 4-(DFPSO-MOGA), 5-(DFPSO-NSGA-II), 6-(DFPSO-DGE3), 7-(DFPSO-DFBA), 8-(DFPSO-DFCS), the third row second box: 1-(DFBA-PSO), 2-(DFBA-BA), 3-(DFBA-CS), 4-(DFBA-MOGA), 5-(DFBA-
Figure 11: The Hypervolume box-plots for a 2-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for DFCS algorithm.

From Figures 25, 26, 27, and 28, we observe that the solutions of
Figure 12: The average Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for PSO algorithm. For each algorithm, the blue line inside the box represents median and the box height represent dispersion. Blue circles are the outliers. The algorithm with high median value finds a good approximation set. A low dispersion value, small box and a low number of outliers represent a stable/ robustness algorithm. (For interpretation of the references to color in these Figures, the reader is referred to the web version of the article.)

DFPSO, DFBA, and DFCS algorithms dominate the solutions of the other compared algorithms (2-objective problem).

Figures 29, 30, 31, 32 illustrate the Set Coverage box-plots for 2-
objective problem with 50 candidate services having 25, 50, 75, and 100 abstract services. In Figure 29, each rectangular box represents a set of Set Coverage values for PSO, BA, CS, MOGA, NSGA-II, GDE3, DFPSO, DFBA, and DFCS algorithms. In Figure 29, the first row first rectangular box represents the set coverage values of compared algorithms such as 1-(PSO-BA), 2-(PSO-CS), 3-(PSO-MOGA), 4-(PSO-NSGA-II), 5-(PSO-GDE3), 6-(PSO-DFPSO), 7-(PSO-DFBA), 8-(PSO-DFCS), the first row second box: 1-(BA-PSO), 2-(BA-CS), 3-(BA-MOGA), 4-(BA-NSGA-II), 5-(BA-GDE3), 6-(BA-DFPSO), 7-
Figure 14: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for CS algorithm.

(BA-DFBA), 8-(BA-DFCS), the first row third box: 1-(CS-PSO), 2-(CS-BA), 3-(CS-MOGA), 4-(CS-NSGA-II), 5-(CS-GDE3), 6-(CS-DFPSO), 7-(CS-DFBA), 8-(CS-DFCS), the second row first box: 1-(MOGA-PSO), 2-(MOGA-BA), 3-(MOGA-CS), 4-(MOGA-NSGA-II), 5-(MOGA-GDE3), 6-(MOGA-DFPSO), 7-(MOGA-DFBA), 8-(MOGA-DFCS), the second row second box: 1-(NSGA-II-PSO), 2-(NSGA-II-BA), 3-(NSGA-II-CS), 4-(NSGA-II-MOGA), 5-(NSGA-II-GDE3), 6-(NSGA-II-DFPSO), 7-(NSGA-II-DFBA), 8-(NSGA-II-DFCS), the second row Third box: 1-(GDE3-PSO), 2-(GDE3-BA), 3-(GDE3-DFBA), 4-(GDE3-DFCS), the second row Fourth box: 1-(DFPSO-PSO), 2-(DFPSO-BA), 3-(DFPSO-CS), 4-(DFPSO-NSGA-II), 5-(DFPSO-GDE3), 6-(DFPSO-DFBA), 7-(DFPSO-DFCS), the second row Fifth box: 1-(DFBA-PSO), 2-(DFBA-BA), 3-(DFBA-CS), 4-(DFBA-NSGA-II), 5-(DFBA-GDE3), 6-(DFBA-DFPSO), 7-(DFBA-DFBA), 8-(DFBA-DFCS), the second row Sixth box: 1-(DFCS-PSO), 2-(DFCS-BA), 3-(DFCS-CS), 4-(DFCS-NSGA-II), 5-(DFCS-GDE3), 6-(DFCS-DFPSO), 7-(DFCS-DFBA), 8-(DFCS-DFCS)
Figure 15: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for MOGA algorithm.

CS), 4-(GDE3-MOGA), 5-(GDE3-NSGA-II), 6-(GDE3-DFPSO), 7-(GDE3-DFBA), 8-(GDE3-DFCS), the third row first box: 1-(DFPSO-PSO), 2-(DFPSO-BA), 3-(DFPSO-CS), 4-(DFPSO-MOGA), 5-(DFPSO-NSGA-II), 6-(DFPSO-DGE3), 7-(DFPSO-DFBA), 8-(DFPSO-DFCS), the third row second box: 1-(DFBA-PSO), 2-(DFBA-BA), 3-(DFBA-CS), 4-(DFBA-MOGA), 5-(DFBA-NSGA-II), 6-(DFBA-DGE3), 7-(DFBA-DFPSO), 8-(DFBA-DFCS), and the third row third box: 1-(DFCS-PSO), 2-(DFCS-BA), 3-(DFCS-CS), 4-(DFCS-MOGA), 5-(DFCS-NSGA-II), 6-(DFCS-DGE3), 7-(DFCS-DFPSO), 8-(DFCS-DFBA).
Similarly, in Figures 30, 31, 32, each rectangular box represents a set of Set Coverage values for PSO, BA, CS, MOGA, NSGA-II, GDE3, DFPSO, DFBA, and DFCS algorithms. In these figures, each algorithm is compared with the other compared algorithms to find the Set Coverage values.

From Figure 29, 30, 31, and 32, we observe that the solution of DFPSO, DFBA, and DFCS algorithms dominate the solutions of the other compared algorithms (2-objective problem). Based on the experimental results (2 and 3-objective problems), we observe that

Figure 16: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for NSGA-II algorithm.
Figure 17: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for GDE3 algorithm.

The details of remaining experiments (for varying combinations of abstract services and candidate services) are omitted, due to the similar nature of experiments.

The average execution time in milliseconds has been evaluated for each algorithm. The execution time are in ascending order: DFCS:10768
Figure 18: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for DFPSO algorithm.
Figure 19: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for DFBA algorithm.
Figure 20: The Hypervolume box-plots for a 3-objective problem (30 independent runs). A scenario with 25, 50, 75, and 100 abstract services versus 25, 50, 75, and 100 candidate services is considered for DFCS algorithm.
Figure 21: Spreading values for 25 abstract services by varying number of candidate services for 2-objective problem.

Figure 22: Spreading values for 50 abstract services by varying number of candidate services for 2-objective problem.

ms, DFBA: 16932 ms, DFPSO: 20112 ms, GDE3: 22568 ms, CS: 24098.12 ms, MOGA: 250156 ms, NSGA-II: 26358 ms, BA: 27013.62
Figure 23: Spreading values for 75 abstract services by varying number of candidate services for 2-objective problem.

Figure 24: Spreading values for 100 abstract services by varying number of candidate services for 2-objective problem.

ms, and PSO: 29689.89 ms. Based on the results, DFCS has the fastest convergence rate compared to other algorithms. In conven-
tional PSO, for each iteration, a particle moves towards a direction estimated from the best-visited position and the best-visited position of all particle in its neighborhood. In this method, the only best fitness of the particle is selected, and the remaining particles are ignored. So, the probability of becoming trapped in the local optima is increased. This change decreases the convergence rate and increases the search space [36]. In BA, for each iteration, the initial population is generated based on the location and the velocity of each bat. In this algorithm, the only best fitness bat is considered, and remaining are eliminated. So, the probability of being trapped into local optima gets increased. In CS, the vector encoding scheme considers only the best solution and the remaining solutions are eliminated. In MOGA and NSGA-II, for each iteration, the initial population is generated randomly (based on each individual crowding distance, crossover, and unguided mutation). The updated population of individuals with the lower rank and the highest distance is survived, and the remaining are eliminated. This whole process takes more computational time and restricts convergence. In GDE3, for each iteration, the new population is generated, and the candidate fitness is evaluated (based on crossover and random mutation of each candidate) that gets changed in each iteration. This change decreases the premature convergence rate and increases the search space, which in turn, consumes excess time. Our proposed algorithms consider the influence of neighborhood of solutions for updating the populations which lead the convergence speed and avoid the trap in local optima. Hence, our proposed methods converge very rapidly and give the satisfactory results in a lesser time than the other algorithms. The experiments show better results for our proposed algorithms efficiently by transforming the discrete problem into a continuous problem.

We have performed statistical analysis tests to ensure that the results of our proposed algorithms are statistically significant (by using
wilcoxon rank-sum test). We have performed Kruskal-Wallis analysis over the HA/HV values, because the procured data does not respect a normal distribution. The wilcoxon test results for 75 and 100 abstract services with 75 and 100 candidate services for 2-objective problem and 3-objective problem are presented in Tables 4 and 5. In Table 4 and Table 5, all the algorithms are compared, where (←, ↑, →) represents the algorithm that procured the best results. The (□) represents that there is no statistical difference between any two algorithms.

Table 4: Wilcoxon sum rank results for a 2-objective problem.

<table>
<thead>
<tr>
<th></th>
<th>PSO</th>
<th>BA</th>
<th>CS</th>
<th>MOGA</th>
<th>GDE3</th>
<th>NSGA-II</th>
<th>DFPSO</th>
<th>DFBA</th>
<th>DFCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>75/100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSO</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA</td>
<td></td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOGA</td>
<td></td>
<td></td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDE3</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSGA-II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFPSO</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFBA</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFCS</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PSO</th>
<th>BA</th>
<th>CS</th>
<th>MOGA</th>
<th>GDE3</th>
<th>NSGA-II</th>
<th>DFPSO</th>
<th>DFBA</th>
<th>DFCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSO</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA</td>
<td></td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOGA</td>
<td></td>
<td></td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDE3</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSGA-II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFPSO</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFBA</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFCS</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each algorithm is assigned a rank by counting the total number of arrows pointing towards that particular algorithm presented in Table 4. From Table 4, the ranking of each algorithm is presented in the descending order as follows: DFCS- rank 13, DFBA- rank 10, DFPSO- rank 10, GDE3- rank 7, CS- rank 5, BA- rank 4, MOGA- rank 3, NSGA-II- rank 2, PSO- rank 1. We also evaluated the ranking of three objective problem as shown in Table 5. From Table 5, the ranking of each algorithm is presented in the descending order as follows: DFCS- rank 14, DFBA- rank 10, DFPSO- rank 10, GDE3- rank 6, CS- rank 6, BA- rank 4, MOGA- rank 4, NSGA-II- rank 4,
PSO- rank 2. Therefore, based on statistical tests, we conclude that our proposed algorithms are statistically more significant than the other algorithms for two objective and three objective problems.

Table 5: Wilcoxon sum rank results for a 3-objective problem.

<table>
<thead>
<tr>
<th>75/100</th>
<th>PSO</th>
<th>BA</th>
<th>CS</th>
<th>MOGA</th>
<th>GDE3</th>
<th>NSGA-II</th>
<th>DFPSO</th>
<th>DFBA</th>
<th>DFCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA</td>
<td>←</td>
<td>→</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
</tr>
<tr>
<td>CS</td>
<td>←</td>
<td>^</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
</tr>
<tr>
<td>MOGA</td>
<td>□</td>
<td>□</td>
<td>↑</td>
<td>-</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>GDE3</td>
<td>←</td>
<td>□</td>
<td>←</td>
<td>↓</td>
<td>←</td>
<td>□</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>↑</td>
<td>←</td>
<td>□</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>DFPSO</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>←</td>
<td>□</td>
<td>←</td>
</tr>
<tr>
<td>DFBA</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>←</td>
</tr>
<tr>
<td>DFCS</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>←</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>100/100</th>
<th>PSO</th>
<th>BA</th>
<th>CS</th>
<th>MOGA</th>
<th>GDE3</th>
<th>NSGA-II</th>
<th>DFPSO</th>
<th>DFBA</th>
<th>DFCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA</td>
<td>←</td>
<td>→</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
</tr>
<tr>
<td>CS</td>
<td>←</td>
<td>^</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>←</td>
</tr>
<tr>
<td>MOGA</td>
<td>□</td>
<td>□</td>
<td>↑</td>
<td>-</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>GDE3</td>
<td>←</td>
<td>□</td>
<td>←</td>
<td>↓</td>
<td>←</td>
<td>□</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>↑</td>
<td>←</td>
<td>□</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>DFPSO</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>←</td>
<td>□</td>
<td>←</td>
</tr>
<tr>
<td>DFBA</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>←</td>
</tr>
<tr>
<td>DFCS</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>□</td>
<td>←</td>
</tr>
</tbody>
</table>
Figure 25: Set Coverage box-plots for 2-objective problem with 25 candidate services for 25 abstract services. Each rectangular sub-figure box-plot represents a set of Set Coverage values from 1 to 8 having the following: 1-(PSO-BA), 2-(PSO-CS), 3-(PSO-MOGA), 4-(PSO-NSGA-II), 5-(PSO-GDE3), 6-(PSO-DFPSO), 7-(PSO-DFBA), 8-(PSO-DFCS), 1-(BA-PSO), 2-(BA-CS), 3-(BA-MOGA), 4-(BA-NSGA-II), 5-(BA-GDE3), 6-(BA-DFPSO), 7-(BA-DFBA), 8-(BA-DFCS). Similarly MOGA, NSGA-II, GDE3, DFPSO, DFBA, DFCA algorithms are compared with the other algorithms.
Figure 26: Set Coverage box-plots for 2-objective problem with 50 candidate services for 25 abstract services.
Figure 27: Set Coverage box-plots for 2-objective problem with 75 candidate services for 25 abstract services.
Figure 28: Set Coverage box-plots for 2-objective problem with 100 candidate services for 25 abstract services.
Figure 29: Set Coverage box-plots for two-objective problem with 25 candidate services for 50 abstract services. Each rectangular sub-figure box-plot represents a set of Set Coverage values from 1 to 8 having the following: 1-(PSO-BA), 2-(PSO-CS), 3-(PSO-MOGA), 4-(PSO-NSGA-II), 5-(PSO-GDE3), 6-(PSO-DFPSO), 7-(PSO-DFBA), 8-(PSO-DFCS), 1-(BA-PSO), 2-(BA-CS), 3-(BA-MOGA), 4-(BA-NSGA-II), 5-(BA-GDE3), 6-(BA-DFPSO), 7-(BA-DFBA), 8-(BA-DFCS). Similarly MOGA, NSGA-II, GDE3, DFPSO, DFBA, DFCA algorithms are compared with the other algorithms.
Figure 30: Set Coverage box-plots for 2-objective problem with 50 candidate services for 50 abstract services.
Figure 31: Set Coverage box-plots for 2-objective problem with 75 candidate services for 50 abstract services.
Figure 32: Set Coverage box-plots for 2-objective problem with 100 candidate services for 50 abstract services.
7. Conclusions and Future Work

In this paper, we presented QoS-aware web service composition solved by PSO, CS, and BA using fuzzy encoding scheme as opposed to encoding as a vector of integers. Fuzzy encoding scheme was used because continuous algorithms cannot be directly used to solve discrete problems like QoS-aware web service composition. DFBA, DFCS, and DFPSO algorithms are developed and implemented using the proposed fuzzy encoding scheme. These proposed algorithms help in selecting a best composition among a set of real world web services that optimizes the QoS attributes of a composite service satisfying the user’s requirements. Based on the experiments, we infer that DFCS algorithm outperformed against all the algorithms according to the Wilcoxon signed rank significant test carried out at 1% level of significance.

In future, we plan to work on finding more effective optimization algorithms (Ageist Spider Monkey Optimization [48], Binary Grey Wolf Optimizer [49]) and solution encoding schemes to cater to the composition of growing number of web and cloud services. Further, we plan to propose a new robust algorithm to handle the unavailable services in dynamic environment.

References


