An extended artificial bee colony with skyline operator for solving the QoS uncertainty-aware web service composition under interval QoS properties

1st Fateh Seghir
Laboratoire des Systèmes Intelligents (LSI), Université Sétif 1
Sétif, Algérie
fateh.seghir@univ-setif.dz

2nd Ghizlane Khababa
Département d'Informatique, Université Sétif 1
Sétif, Algérie
ghizlane.khababa@yahoo.com

***Abstract*—In this paper, we provide an extended artificial bee colony (EABC) algorithm with skyline operator for solving the QoS uncertainty-aware web service composition (IQSC) problem, where the uncertain QoS properties have been expressed as intervals numbers. At first, we formulate the addressed problem as an interval constrained single-objective optimization model. Then we employ the skyline operator to reduce the search space of IQSC. Whereas, the EABC algorithm has been performed to solve IQSC in a reduced search space more effectively and efficiently. To validate the performance and efficiency of the proposed approach, we present the experimental comparisons to an existing skyline-based PSO algorithm on an interval extended version of the public QWS dataset.**

***Index Terms*—Web service composition, Quality of Service (QoS), Interval number, Skyline operator, Artificial bee colony**

I. Introduction

As far as the Service Oriented Architecture (SOA) is concerned, any hardware or software resource can be easily provided as a web service to the end-users. However, with the increasing number of the web services, it offers the same functionalities but with different values in their nonfunctional properties, known by the Quality of Service (QoS) parameters such as response time, price, availability, ..., etc. Therefore, selecting the best web services from their sets of functionally equivalent ones to build the best Composite Service (CS), which should satisfy the end-users' local and global QoS requirements, becomes a challenging problem for both industrial and academic researchers [1], [2]. This problem, known by the QoS-aware web service composition (QSC), is an NP-hard optimization one, where many optimization methods including exact, heuristic and meta-heuristic web service composition approaches have been provided to solve it [3].

From the systematic literature review of [3], most of the existing exact web services selection algorithms modeled the QSC problem as an integer/mixed-integer linear programming model [2], [4], where solvers like LpSolve[1] and CPLEX[2] have been performed to solve the modeled QSC. These solvers can get the optimal solutions of QSC, but they need the linearization of the QSC's objective functions and constraints. Moreover, as the QSC scale increases, the efficiency of the exact web services selection algorithms decreases. Therefore, those based on the evolutionary and bio-inspired algorithms like Genetic Algorithm (GA) [5], Particle Swarm Optimization (PSO) [6], Artificial Bee Colony (ABC) [7], [8], and so on, have drawn the attention of the QSC researchers. These algorithms can get optimal/near-optimal solutions within reasonable processing times and without any objective functions and constraints linearization.

All the above QSC studies consider the advertised QoS values as non-ambiguous, but in real-world environments and due to some unconditional factors of the SOAs such as the network topologies changes and economic policies. Therefore, the QoS values of web services are uncertain in nature [9]. For this reason, some recent QoS-aware web service composition approaches have been proposed to solve the QSC problem under uncertain QoS parameters that have been modeled as intervals numbers [10], [11], probabilistic variables [12] or fuzzy numbers [13]. In the web services selection process of these approaches, all the provided web services are considered as potential candidates to construct the final CS. However, some of them are not possible candidates to build this final solution. Therefore, some existing studies [14]–[16] have used the Skyline operator [17] to reduce the search space of QSC by pruning the web services that cannot be part of the final solutions, since they are dominated by some of their functionally equivalent web services partners. However, in all the aforesaid Skyline-based QSC studies, the QoS parameters are considered with precise and exact values. Hence, our motivation in this work is to provide an efficient approach for solving the QoS uncertainty-aware service composition in a reduced search space.

In this paper, we first propose an interval constrained single-objective optimization model to the QoS uncertainty-aware web service composition (IQSC) problem, where its QoS parameters are expressed as intervals numbers. Then, and to address the formulated IQSC, we propose an approach including two components. **(1)** The first one (Skyline operator) is used to reduce the search space of IQSC, and thus its best CS

[1]http://lpsolve.sourceforge.net/5.5/
[2]https://www.ibm.com/analytics/cplex-optimizer

can be found very quickly with high solution quality. **(2)** The second component is an extended version of the basic ABC algorithm, named EABC, which is performed to solve the formulated IQSC in a reduced search space more effectively and efficiently. Finally, to demonstrate the performance and efficiency of the proposed approach, we have compared it to the one of reference [15], where the comparison experiments are performed on a new interval extended version of the public QWS [18] dataset.

The remainder part of this paper is organized as follows. In Section II, we review and summarize some relevant and notable works to ours. In Section III, we give some preliminaries on intervals numbers to be used throughout the rest of this study. In Section IV, we mathematically formulate the QoS uncertainty-aware web service composition under interval QoS properties, denoted by IQSC, as an interval constrained single-objective optimization model. To address the formulated IQSC, our proposed approach, which includes two components Skyline operator and EABC, is described with details in Section V. Section VI is devoted to discuss the comparison results of our experiments. Finally, our conclusion and future work are given in Section VII.

II. Related work

From the literature, the QSC has been addressed by two main categories of web service selection approaches, including **(1)** Exact and **(2)** Meta-heuristic (i.e., evolutionary and bio-inspired) optimization methods [3]. Moreover, to improve the efficiency of the QSC approaches, the Skyline operator [17] has been used by some researchers to reduce the time of web services selection [14]–[16]. In addition, due to some unconditional factors of the SOA environments, such as the network topologies changes and economic policies [9], some QSC studies have considered the QoS parameters with ambiguous values [11], [13]. Here in this section, we have only reviewed and discussed some relevant and notable works to ours.

The exact optimization methods have modeled the QSC problem as an Integer Linear Programming (ILP) model [1] or a Mixed ILP (MILP) one [2], [4] and have solved it using existing ILP/MILP solvers such as LpSolve and CPLEX. For the QSC problem instances with small search spaces, these methods yield good performance in terms of running time and solution quality of the obtained final CSs solutions. However, the computation time of these methods increases exponentially with the increasing number of the provided functionally identical web services. Moreover, the exact optimization approaches require the linearization of the QSC objective functions and the users' global QoS constraints. Therefore, those based on evolutionary and bio-inspired optimization algorithms like GA [5], PSO [6], ABC [7], [8], and so on, have drawn the attention of the QSC researchers. Compared to the exact optimization methods that can obtain optimal CSs solutions, the meta-heuristic algorithms can get near-optimal ones, but with reasonable processing times and without any linearization of the objective functions and the users' global QoS constraints of the QSC problem.

To improve the efficiency of the QSC optimization approaches, a representative mechanism: Skyline operator [17] has been employed to reduce the search space of the QSC problem. Hence, the computation time cost of these approaches is shortened. For instance, authors of [14] were the firsts who have used the Skyline operator to reduce the QSC search space where a new service dominance based on the QoS attributes of web services has been performed, among web services, to prune the dominated ones. In this study, the QSC problem was formulated as an ILP model and solved more efficiently using the existing LpSolve solver in a reduced search space. In [15], the corresponding authors proposed a fast cloud-based web service composition approach, which prunes the redundant and dominated candidate web services by an adopted Skyline operator and performs the PSO algorithm to find out a more powerful final CS solution. In [16], the authors have used the mathematical programming language (AMPL) [19] to formulate the QSC problem as a nonlinear integer programming model, which has been solved by the existing Bonmin[3] solver. In this study, and similar to the ones of [14] and [15], the Skyline operator has been employed to reduce the search space of the QSC problem.

However, all the above studies consider the advertised QoS values as non-ambiguous ones, which is unreal for dynamic SOA environments since some unconditional factors, like network topologies and economic policies, render the QoS values uncertain in nature. Therefore, some researchers have formulated the QSC problem as a non-deterministic optimization model using interval-numbers [10], [11], fuzzy numbers [13], or probabilistic models [12].

Here, by representing the QoS uncertainty with the interval number model, we formulated the QSC problem as an interval constrained single-objective optimization one. The latter is solved by an extended artificial bee colony algorithm, while an interval-based version of the Skyline operator is employed to reduce the search space of IQSC that will improve the efficiency of the provided algorithm.

III. Preliminaries on intervals numbers

Here in this section, we have introduced the following definitions related to the intervals numbers that have been employed throughout the remainder part of this article.

***Definition** 1:* (*Interval Number* [20]) Given any two real numbers a^l and a^u with $a^l \leq a^u$. The set $A = \{x : x \in \mathbb{R}$ and $a^l \leq x \leq a^u\}$ denoted by $A = [a^l, a^u]$ is named an interval number, where a^l and a^u are called its lower and upper limits, respectively.

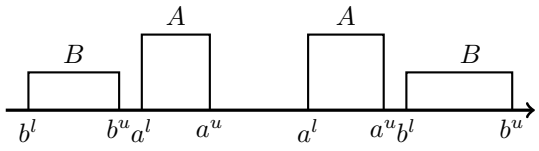
***Definition** 2:* (*Arithmetic operations on intervals numbers* [21]) Let $A = [a^l, a^u]$ and $B = [b^l, b^u]$ be any two intervals numbers, then the arithmetic operations on A and B are defined as follows:

- **Addition $\oplus$:** $A \oplus B = [a^l, a^u] \oplus [b^l, b^u] = [a^l + b^l, a^u + b^u]$

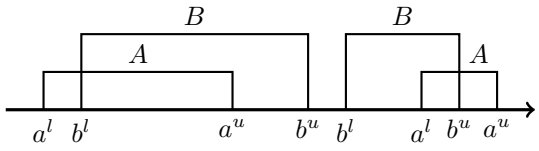
[3]https://github.com/coin-or/Bonmin

- **Subtraction** $\ominus$**:** $A \ominus B = [a^l, a^u] \ominus [b^l, b^u] = [a^l - b^u, a^u - b^l]$
- **Multiplication** $\otimes$**:** $A \otimes B = [a^l, a^u] \otimes [b^l, b^u] = [m^l, m^u]$ with $m^l = \min(a^l b^l, a^l b^u, a^u b^l, a^u b^u)$ and $m^u = \max(a^l b^l, a^l b^u, a^u b^l, a^u b^u)$
- **Scalar multiplication:** $\forall \lambda \in \mathbb{R}$, $\lambda \otimes A = \lambda \otimes [a^l, a^u] = \begin{cases} [\lambda * a^l, \lambda * a^u] \text{ if } \lambda >= 0 \\ [\lambda * a^u, \lambda * a^l] \text{ if } \lambda < 0 \end{cases}$

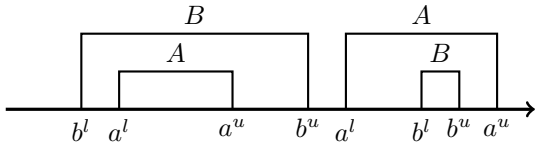
Moreover, let $A = [a^l, a^u]$ and $B = [b^l, b^u]$ be any two intervals numbers, the three possible types of overlapping between A and B as shown in Fig. 1 [22] are used in the following definitions for ranking intervals numbers of the maximization (minimization) optimization problems with objective functions modeled as intervals numbers.



(a) **Type I:** Non-overlapping between intervals numbers



(b) **Type II:** Partial overlapping between intervals numbers



(c) **Type III:** Full overlapping between intervals numbers

Fig. 1. Types of overlapping between intervals numbers

***Definition** 3: (Interval greater ranking operator for maximization optimization problems:* $>_{\max}$ [11]) For any two intervals numbers $A = [a^l, a^u]$ and $B = [b^l, b^u]$, the interval greater ranking operator $>_{\max}$ between A and B is defined as follows:

1) For intervals numbers of Type I or Type II, $A >_{\max} B$ if $(a^l > b^l)$ and $(a^u > b^u)$;
2) For intervals numbers of Type III, $A >_{\max} B$ if:
   a) $(a^l - b^l) > (b^u - a^u)$;
   b) or $\big((a^l - b^l) = (b^u - a^u)\big)$ and $(a^u < b^u)$

***Definition** 4: (Interval smaller ranking operator for minimization optimization problems:* $<_{\min}$) [11]) For any two intervals numbers $A = [a^l, a^u]$ and $B = [b^l, b^u]$, the interval smaller ranking operator $<_{\min}$ between A and B is defined as follows:

1) For intervals numbers of Type I or Type II, $A <_{\min} B$ if $(a^l < b^l)$ and $(a^u < b^u)$;
2) For intervals numbers of Type III, $A <_{\min} B$ if:
   a) $(a^l - b^l) < (b^u - a^u)$;
   b) or $\big((a^l - b^l) = (b^u - a^u)\big)$ and $(a^u < b^u)$

***Definition** 5: (Interval equal ranking operator*: $=$) For any two intervals numbers $A = [a^l, a^u]$ and $B = [b^l, b^u]$, if $a^l = b^l$ and $a^u = b^u$ then A and B are equal intervals numbers represented as $A = B$.

***Definition** 6:* For any two intervals numbers A and B, the interval greater or equal ($\geq_{\max}$) and the interval smaller or equal ($\leq_{\min}$) ranking operators for the maximization and the minimization optimization problems, respectively, are defined as follows:

- $A \geq_{\max} B \Leftrightarrow (A >_{\max})$ or $(A = B)$
- $A \leq_{\min} B \Leftrightarrow (A <_{\min})$ or $(A = B)$

***Theorem** 1:* Given any three intervals numbers A, B and C, the following mathematical order relations are provided:

($\mathcal{O}1$) $A \leq_{\min} (\geq_{\max}) A$, which is named reflexivity.
($\mathcal{O}2$) If $A \leq_{\min} (\geq_{\max}) B$ and $B \leq_{\min} (\geq_{\max}) A$ then $A = B$, which is named anti-symmetry.
($\mathcal{O}3$) If $A \leq_{\min} (\geq_{\max}) B$ and $B \leq_{\min} (\geq_{\max}) C$ then $A \leq_{\min} (\geq_{\max}) C$, which is named transitivity.

***Proof** of **Theorem** 1 :* Since the pages number of this paper is limited; the mathematical order relations proofs of $\mathcal{O}1$, $\mathcal{O}2$ and $\mathcal{O}3$ are omitted. We have easily proofed them using Definitions 3, 4, 5 and 6.

***Definition** 7: (The interval minimum* (Min) *and the interval maximum* (Max) *operators*) let $\{A_j, j = 1, 2, \ldots, m\}$ be a set of m intervals numbers, then the minimum and the maximum intervals numbers of $\{A_j, j = 1, 2, \ldots, m\}$ for the minimization and the maximization optimization problems, respectively, are defined as follows:

- If $A_q \leq_{\min} A_j$ for all $j = 1, 2, \ldots, m$ with $j \neq q$ then $A_q = \text{Min}_{j=1}^{m}\{A_j\}$ is the minimum interval number of $\{A_j, j = 1, 2, \ldots, m\}$.
- If $A_p \geq_{\max} A_j$ for all $j = 1, 2, \ldots, m$ with $j \neq p$ then $A_p = \text{Max}_{j=1}^{m}\{A_j\}$ is the maximum interval number of $\{A_j, j = 1, 2, \ldots, m\}$.

IV. Interval model of the QoS uncertainty-aware web service composition problem

Let given a very large set of atomic web services, where each web service, denoted by ws, is characterized by two types of properties, functional and nonfunctional ones. The functional parameters (i.e., input and output attributes) of a ws represent its supported functionality. Whereas, the non-functional properties (QoS) of a ws, such as response time, availability, reputation, price, ..., etc, represent its parameters quality. Each QoS attribute, denoted by q_t with $t = 1, 2, \ldots, r$ and r is the size of the whole considered QoS parameters, can be either positive or negative parameter, where a ws with larger (lower) values in its positive q_ts, the better (worse) is, whereas a ws with lower (larger) values in its negative q_ts, the better (worse) is. Here in this study, the positive list of q_ts is denoted by QoS^+ and the negative list of q_ts is indicated by QoS^-. For example, the reputation and availability attributes belong to the QoS^+ list, whereas the price and response time parameters belong to the QoS^- list. Furthermore, a web

services class, denoted by $S = \{ws_1, ws_2, \ldots, ws_m\}$, is a set of m atomic web services that have similar functionalities but with different values in their q_ts.

As seen in Fig. 2, which represents the graphic depiction of the QSC problem. For a given abstract composite service, denoted by $ACS = \{S_1, S_2, \ldots, S_n\}$, which represents the n needed web services classes to reply a complex user request, and a list of k user's global QoS requirements, denoted by Cst_{q_k}s with $k \leq r$. A concert composite service, indicated by CCS, represents the user request's response, which is built by selecting from each web services class $S_i = \{ws_i^1, ws_i^2, \ldots, ws_i^{m_i}\}$ a unique web service ws_i^j with $j \leq m_i$, where the global QoS values of CCS that have been evaluated by aggregating the QoS values of their atomic wss should meet the considered Cst_{q_k}s. In this study, as shown in Fig 3, four basic connection structures (i.e., sequential, parallel, branch and loop) have been considered to compose the atomic wss of CCSs. Furthermore, due to some unconditional factors of the SOAs like network architectures changes and economic policies, the wss' QoS values are uncertain in nature [9]. Therefore, motivated by the fact that the interval number is efficient, general and easy representation model to express perfectly the uncertain QoS values [11], and by considering the four mostly used QoS parameters in solving the QSC problem [2] including two positive QoS attributes: *availability* (q_1) and *throughput* (q_2), and two negative ones: *response time* (q_3) and *price* (q_4). Hence, the interval constrained single-objective optimization model of QSC, denoted by IQSC, can be stated as follows:

Determine the best CCS solution among all the possible ones[4], which maximize

$$\text{Maximize } IU(CCS) = \textstyle\sum_{t=1}^{r} w_{q_t} \otimes \overline{CCS_{q_t}} \tag{1}$$

Subject to k user's global QoS constraints

$$\forall t = 1 \ldots k \begin{cases} CCS_{q_t} \leq_{\min} Cst_{q_t}, \text{ if } q_t \in QoS^- \\ CCS_{q_t} \geq_{\max} Cst_{q_t}, \text{ if } q_t \in QoS^+ \end{cases} \tag{2}$$

Where CCS_{q_t} is the global interval QoS value of the CCS solution in the q_t attribute that can be evaluated as seen in Table I using the interval arithmetic operations given in Definition 2 and the four basic connection structures as depicted in Fig. 3 for the interval values $ws_{i,q_t} = \left[ws_{i,q_t}^l, ws_{i,q_t}^u\right]$ of each atomic web service ws_i of CCS. $IU(CCS)$ is the interval utility function of CCS that maps the CCS_{q_t}s values into a single interval value. This function adopts the well-known *Simple Additive Weighting* (SAW) method [23] through scaling the interval values CCS_{q_t}s into their normalized interval ones $\overline{CCS_{q_t}}$s. Then afterwards, these normalized values are aggregated using the weights w_{q_t}s with $w_{q_t} \in [0, 1]$ and $\sum_{t=1}^{r} w_{q_t} = 1$, which represent the importance and priority of each q_t by the user. The normalized interval value $\overline{CCS_{q_t}}$ of its related original one $CCS_{q_t} = \left[CCS_{q_t}^l, CCS_{q_t}^u\right]$ is calculated as given in the following interval positive and negative normalization Equations 3 and 4, respectively.

[4]For an ACS of n web services classes, where each one has m candidate web services, then m^n different CCSs can be obtained

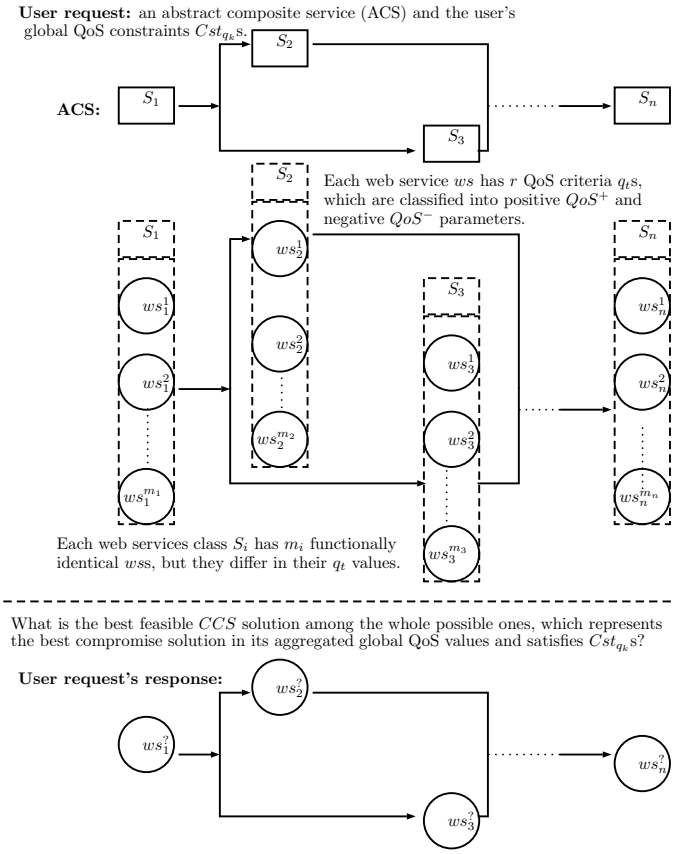


Fig. 2. Graphic depiction of the QoS-aware web service composition problem

- Interval positive normalization

$$\forall q_t \in QoS^+, \overline{CCS_{q_t}} = \begin{cases} \left[\frac{CCS_{q_t}^l - min_{q_t}^l}{max_{q_t}^u - min_{q_t}^l}, \frac{CCS_{q_t}^u - min_{q_t}^l}{max_{q_t}^u - min_{q_t}^l}\right] & \text{if } max_{q_t}^u \neq min_{q_t}^l \\ [1, 1] & \text{if } max_{q_t}^u = min_{q_t}^l \end{cases} \tag{3}$$

- Interval negative normalization

$$\forall q_t \in QoS^-, \overline{CCS_{q_t}} = \begin{cases} \left[\frac{max_{q_t}^u - CCS_{q_t}^u}{max_{q_t}^u - min_{q_t}^l}, \frac{max_{q_t}^u - CCS_{q_t}^l}{max_{q_t}^u - min_{q_t}^l}\right] & \text{if } max_{q_t}^u \neq min_{q_t}^l \\ [1, 1] & \text{if } max_{q_t}^u = min_{q_t}^l \end{cases} \tag{4}$$

Where the real limit values $min_{q_t}^l$ and $max_{q_t}^u$ for an $ACS = (S_1, S_2, \ldots, S_n)$ with $\forall i \in \{1, 2 \ldots, n\}$, $S_i = \{ws_i^1, ws_i^2, \ldots, ws_i^{mi}\}$ and $\forall j \in \{1, 2 \ldots, m_i\}$, $ws_{i,q_t}^j = \left[ws_{i,q_t}^{j,l}, ws_{i,q_t}^{j,u}\right]$ are evaluated as follows

$$max_{q_t}^u = Agg_{q_t}{}_{i=1}^{n} \left(\text{Max}_{j=1}^{m_i} \left\{ws_{i,q_t}^{j,u}\right\}\right) \tag{5}$$

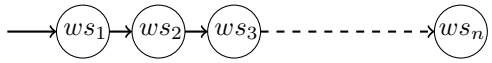
$$min_{q_t}^l = Agg_{q_t}{}_{i=1}^{n} \left(\text{Min}_{j=1}^{m_i} \left\{ws_{i,q_t}^{j,l}\right\}\right) \tag{6}$$

Where $Agg_{q_t}{}_{i=1}^{n}$ denotes the related crisp QoS aggregation formula (i.e., $\sum$, $\prod$, Min and Max) of the q_t attribute as defined in Table I that has been employed to aggregate the n

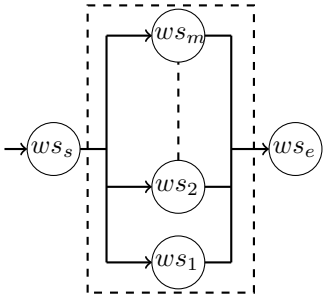
TABLE I
INTERVAL QOS AGGREGATION FORMULAS FOR EVALUATING THE GLOBAL QOS VALUES OF CONCRETE COMPOSITES SERVICES CCSs

QoS parameter	n **sequential web services** ws_i	m **parallel web services** ws_i	**branch**[a] **of** m **web services** ws_i **with their** pr_i**s probabilities**	**call a web service** ws **with** p **times**
Availability (q_1)	$\prod_{i=1}^{n}\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]$	$\prod_{i=1}^{m}\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]$	$\mathrm{Min}_{i=1}^{m}\left\{\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]\right\}$	$\prod_{i=p}^{n}\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]$
Throughput(q_2)	$\mathrm{Min}_{i=1}^{n}\left\{\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]\right\}$	$\mathrm{Min}_{i=1}^{m}\left\{\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]\right\}$	$\mathrm{Min}_{i=1}^{m}\left\{\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]\right\}$	$\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]$
Response time (q_3)	$\sum_{i=1}^{n}\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]$	$\mathrm{Max}_{i=1}^{m}\left\{\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]\right\}$	$\mathrm{Max}_{i=1}^{m}\left\{\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]\right\}$	$p * \left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]$
Price (q_4)	$\sum_{i=1}^{n}\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]$	$\sum_{i=1}^{m}\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]$	$\mathrm{Max}_{i=1}^{m}\left\{\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]\right\}$	$p * \left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]$

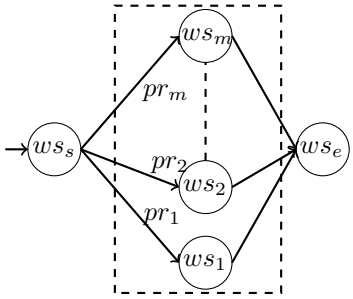
[a]In the branch connection structure, according to the pr_is of ws_i, only one web service among them is executed.



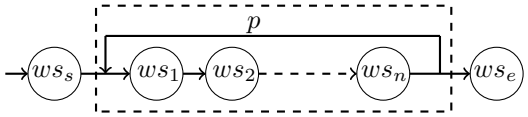
(a) Sequential connection of n wss



(b) Parallel connection of m wss



(c) Branch connection of m wss with their pr_is probabilities, $\forall i = 1, 2, \ldots, m$: $pr_i \in$ [0,1] and $\sum_{i=1}^{m} pr_i = 1$



(d) Loop conncetion structure of n wss with p calls

Fig. 3. Connection structures between atomic web services

obtained real values $\mathrm{Max}\left\{ws_{i,q_t}^{j,u}\right\}/\mathrm{Min}\left\{ws_{i,q_t}^{j,l}\right\}$ from each web services class S_i with $j = 1, 2, \ldots, m_i$.

V. PROPOSED APPROACH

To solve the IQSC problem, we propose an approach including two components. The first one, which uses the Skyline operator [17], is employed to reduce the search space of IQSC by pruning the dominated candidate web services. The second component is to quickly find out the optimal/near-optimal solution of IQSC by performing an interval extended version of the basic ABC algorithm, named EABC.

A. Skyline service

Our main goal in this study is to find an optimal/near-optimal CCS solution for IQSC that maximize the interval utility value as defined in Equation 1 and satisfies the user's overall QoS constraints as given in Equation 2. This optimal/near-optimal solution contains a set of atomic wss, where each one is selected from its related web services class. However, not all wss of each web services class are potential candidates to construct the final optimal/near-optimal solution. Hence, some existing studies [14]–[16] have used the Skyline operator [17] to reduce the search space of QSC by pruning the wss that cannot be part of the final solutions, since they are dominated by some of their functionally identical wss partners. However, in the whole aforesaid skyline-based studies, the QoS parameters are considered with precise and exact values. Therefore, extended definitions of *Service Dominance* and *Skyline Service* [15] that consider QoS properties as intervals numbers have been introduced in this paper.

***Definition** 8:* (*Service Dominance*) Let's consider a web services class S and two web services $ws_1, ws_2 \in S$, where each one has a set of QoS parameters q_ts . We say ws_1 dominates ws_2, denoted as $ws_1 \prec ws_2$, if and only if: **(1)** $\forall q_t \in QoS^+$: $ws_{1,q_t} \geq_{\max} ws_{2,q_t}$ and $\exists q_t \in QoS^+, ws_{1,q_t} >_{\max} ws_{2,q_t}$, and **(2)** $\forall q_t \in QoS^-$: $ws_{1,q_t} \leq_{\min} ws_{2,q_t}$ and $\exists q_t \in QoS^-, ws_{1,q_t} <_{\min} ws_{2,q_t}$.

***Definition** 9:* (*Skyline Service*) For a given web services class $S = \{ws_1, ws_2, \ldots, ws_m\}$ of m functionally identical wss. The skyline service of S, denoted by SkS, contains the candidate wss in S that cannot be dominated by any other ws of S. i.e, $SkS = \{ws_i \in S \mid \nexists ws_j \in S : ws_j \prec ws_i\}$.

To define the skyline service SkS of each web services class S, the skyline computation process performs pair-wise comparisons between the ws_{q_t}s interval values of the compared wss of S. This calculation process can be expensive in terms of evaluation time, especially, if S has numerous wss. However, for the IQSC problem, the skylines services are independent of any online user request [14]. Therefore, the skyline computation can be performed offline by any of the exiting efficient skyline algorithms [17].

B. Extended Artificial Bee Colony: EABC

The ABC algorithm is a well-known bio-inspired optimization algorithm that mimics the smart work of honey bees to find out the optimal/near-optimal solution of an optimization problem [24]. It has been widely applied to solve the QSC problem for several service-based environments like service-oriented applications [8] and cloud computing [7]. Similar to the canonical ABC algorithm, in order to find out the optimal/near-optimal solution of IQSC, EABC performs repetitively and successively three types of bees (Employees, Onlookers and Scouts) on each explored food sources area src_g to find a new better one src_{g+1} with $g = 0, 1, 2, \ldots, MITR$ and $MITR$ is a maximum iteration number of the EABC algorithm. The initial food sources area of Z solutions (food sources positions) $src_0 = \{CCS_1^0, CCS_2^0, \ldots, CCS_Z^0\}$ is generated randomly, where the dimensional values (i.e., web services) of each food source position CCS_z^0 with $z = 1, 2, \ldots, Z$ are defined according to the encoding schema given in the next Section and its nectar amount (i.e. fitness value) is evaluated through using the interval utility function, i.e., $IU(CCS_z^0)$, as defined in Equation 1. Moreover, for each generated food source position CCS_z^0, an integer variable $trial_z$, which is initialized to zero, is assigned to it. In the following subsections, we describe the encoding schema of CCSs, generation of the initial food sources area src_0, the works of the three types of bees for searching good food sources areas, which are *Employees work*, *Onlookers work* and *Scouts work*, as well as the ending criterion of EABC.

1) Encoding schema of food sources positions CCSs and generation of the initial food sources area src_0: Here, in this study, an n-dimensional array of integers is used to represent each food source position CCS_z^g of the g^{th} food sources area src_g, where $z = 1, 2, \ldots, Z$ and Z is the number of food sources positions (i.e., population size of EABC). The CCS_z^g position denoted as $CCS_z^g = (CCS_{z,1}^g, CCS_{z,2}^g, \ldots, CCS_{z,n}^g)$ has n integer elements indicating the selected atomic wss from their skylines services SkS_i, $i = 1, 2, \ldots, n$. For the initial food sources area src_0, each integer element of each food source position $CCS_z^0 = (CCS_{z,1}^0, CCS_{z,2}^0, \ldots, CCS_{z,n}^0)$ was randomly generated as follows

$$\begin{gathered} \forall : i = 1, 2, \ldots, n \\ CCS_{z,i}^0 = 1 + \lceil rand(0,1) * (m_r - 1) \rceil \end{gathered} \tag{7}$$

Where $CCS_{z,i}^0$ is an integer number representing the i^{th} selected ws from the i^{th} skyline service SkS_i that has m_i functionally identical wss, $rand(0,1)$ is a real number, which was randomly generated from the range $[0, 1]$, and $\lceil\ \rceil$ is the rounding up integer function.

2) Employees work: The employees bees explore each current food sources area src_g to find a new better one src_g^{Emp}, where similar to the ABC's conventional updating positions of food sources [24], only one dimension (i.e., a unique web service) of each food source position $CCS_z^g = (CCS_{z,1}^g, CCS_{z,2}^g, \ldots, CCS_{z,j}^g, \ldots, CCS_{z,n}^g)$ with n atomic web services is considered to update the CCS_z^g position. Hence, the new food source position $CCS_z^{Emp} = (CCS_{z,1}^g, CCS_{z,2}^g, \ldots, CCS_{z,j}^{Emp}, \ldots, CCS_{z,n}^g)$ has the same atomic web services as its old one CCS_z^g, except for the j^{th} web service (i.e., the $CCS_{z,j}^{Emp}$ value), which is defined using the following Equation.

$$CCS_{z,j}^{Emp} = \lceil CCS_{z,j}^g + \phi_j * (CCS_{z,j}^g - CCS_{l,j}^g) \rceil \tag{8}$$

Where for each explored CCS_z^g, j is a randomly-selected dimension (i.e., the j^{th} skyline service SkS_j) from the range $[1, n]$, CCS_l^g with $l \neq z$ represents a randomly-selected food source position from src_g, ϕ_j is a random generated real value within the range $[-1, 1]$ for every selected skyline service SkS_j and $\lceil\ \rceil$ is the rounding up integer function.

After defining the new food sources positions CCS_z^{Emp}s with $z = 1, 2, \ldots, Z$, their interval utility values, i.e., $IU(CCS_z^{Emp})$, are calculated using Equation 1. And finally, in order to update each old food source position CCS_z^g by its new defined one CCS_z^{Emp}, the greedy selection mechanism of ABC [24] is adapted through using the following steps of the Deb's selection procedure [25].

- If CCS_z^{Emp} and CCS_z^g are feasible food sources positions and $IU(CCS_z^{Emp}) >_{\max} IU(CCS_z^g)$ then CCS_z^{Emp} is maintained and its $trial_z$ is reset to zero.
- If CCS_z^{Emp} is a feasible food source position and CCS_z^g is an infeasible one then CCS_z^{Emp} is maintained and its $trial_z$ is reset to zero.
- If CCS_z^{Emp} is an infeasible food source position and CCS_z^g is a feasible one then CCS_z^{Emp} is replaced by its old one CCS_z^g and its $trial_z$ is incremented by one.
- If CCS_z^{Emp} and CCS_z^g are infeasible food sources positions then:
  - If the lower global constraint violation of CCS_z^{Emp}, as will be given in the next paragraph, is lower than the one of CCS_z^g then CCS_z^{Emp} is maintained and its $trial_z$ is reset to zero; otherwise, CCS_z^{Emp} is changed by its old one CCS_z^g and its $trial_z$ is augmented by one.

Global constraint violation of an infeasible food source position: Given any infeasible food source position, denoted by $ICCS$, the constraint violation amounts of its violated user's overall QoS requirements, denoted by $ICCS_{q_k}^{cst}$s, are evaluated as follows.

$$\begin{aligned} \forall q_k \in QoS^+ :\ & \text{if } Cst_{q_k} >_{\max} ICCS_{q_k} \text{ then} \\ & ICCS_{q_k}^{cst} = Cst_{q_k} \ominus ICCS_{q_k} \end{aligned} \tag{9}$$

$$\begin{aligned} \forall q_k \in QoS^- :\ & \text{if } Cst_{q_k} <_{\min} ICCS_{q_k} \text{ then} \\ & ICCS_{q_k}^{cst} = ICCS_{q_k} \ominus Cst_{q_k} \end{aligned} \tag{10}$$

By adapting the SAW method [23] to support the interval numbers calculations, the $ICCS_{q_k}^{cst}$s intervals values are aggregated into a single interval value $ICCS_{cst}$ through using the following equation.

$$ICCS_{cst} = \frac{1}{V} \otimes \sum_{t=1}^{k} \overline{ICCS_{q_t}^{cst}} \tag{11}$$

Where V is the total number of the violated overall QoS constraints by $ICCS$, $\overline{ICCS_{q_k}^{cst}}$ is the normalized interval value of its associated original one $ICCS_{q_k}^{cst} = \left[ICCS_{q_k}^{l,cst}, ICCS_{q_k}^{u,cst}\right]$ for the kth user's global QoS constraint $Cst_{q_k} = \left[Cst_{q_k}^{l}, Cst_{q_k}^{u}\right]$. Since the lower $ICCS_{q_k}^{cst}$ is, the lower its global constraint violation amount is (i.e., in other words, the higher its aggregated normalized interval value of its violated constraint $ICCS_{cst}$ is). Therefore, the interval values $\overline{ICCS_{q_k}^{cst}}$s are calculated using the interval negative normalization process as given in the following Equations.

$$\forall q_k \in QoS^{-}, \overline{ICCS_{q_k}^{cst}} = \begin{cases} \left[\frac{max_{q_k}^{u} - Cst_{q_k}^{l} - ICCS_{q_k}^{u,cst}}{\left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) - \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right)}, \frac{max_{q_k}^{u} - Cst_{q_k}^{l} - ICCS_{q_k}^{l,cst}}{\left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) - \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right)}\right] & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) \neq \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \\ [1,1] & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) = \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \end{cases} \quad (12)$$

$$\forall q_k \in QoS^{+}, \overline{ICCS_{q_k}^{cst}} = \begin{cases} \left[\frac{max_{q_k}^{u} - Cst_{q_k}^{l} + ICCS_{q_k}^{l,cst}}{\left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) - \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right)}, \frac{max_{q_k}^{u} - Cst_{q_k}^{l} + ICCS_{q_k}^{u,cst}}{\left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) - \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right)}\right] & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) \neq \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \\ [1,1] & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) = \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \end{cases} \quad (13)$$

Where $max_{q_k}^{u}$ and $min_{q_k}^{l}$ have been previously defined in Equations 5 and 6, respectively.

3) Onlookers work: The onlooker bees select Z food sources positions from the discovered src_g^{Emp} by the employees bees to be explored again. In the basic ABC algorithm, the selection criterion of the Z food sources positions is based on the roulette wheel selection [24]. However, the latter has the local optima stagnation problem [26]. Therefore, in our study, the binary tournament selection method [27] is employed to create the new selected food sources area, denoted by src_g^{Sel}. The same Deb's selection procedure as given in the above subsection is used to create src_g^{Sel} by repeating this selection procedure Z times for each two randomly-selected solutions from the discovered food sources area src_g^{Emp}.

Again, a new food sources area src_g^{Onl} is explored from the selected one src_g^{Sel} through using the same updating food sources positions given in Equation 8, where the aforesaid steps of the Deb's selection procedure given in the *Employees work* subsection are used to update src_g^{Sel} by src_g^{Onl}.

4) Scouts work: Like the scouts work of the original ABC algorithm, here, the scouts bees of EABC are used to update one randomly-selected food source position from the generated ones by the onlookers bees (i.e., src_g^{Onl}) that have not updated theirs positions after $limit$ iterations, where the $limit$ parameter of EABC indicates the criterion to identify an abandoned food source position. The latter is updated using Equation 7 and its $trial$ value is reinitialized to zero. As result, a new food sources area src_{g+1} is obtained.

5) Ending criterion of EABC: The aforesaid works of employees, onlookers and scouts bees are repeated for each new generated food sources area src_{g+1} until the stopping criterion of the EABC algorithm is satisfied (i.e., the maximum iteration number $MITR$ is reached). The best feasible food source position among the ones of the last discovered src_{g+1} that has the highest interval utility value is the final optimal/near-optimal CSS solution in solving IQSC by EABC.

VI. Experiments

In order to validate the effectiveness of our proposed approach, we have used two comparison metrics: **(a) Running time** and **(b) Optimality**. The first one indicates the needed computation time by each performed algorithm to find out its optimal/near-optimal solutions in solving IQSC. Whereas, the second one shows the quality of these obtained solutions in terms of their interval utility values as defined in Equation 1.

A. Interval version of the public QWS dataset

In the comparison experiments, as the public QWS dataset [18] was published with 2507 atomic web services, where each web service has 09 non-ambiguous QoS values for 09 QoS properties including (1) Response Time, (2) Availability, (3) Throughput, (4) Successability, (5) Reliability, (6) Compliance, (7) Best Practices, (8) Latency and (9) Documentation. Therefore, an interval version of QWS, denoted by IQWS, is provided to be used in our experiments, where the QoS intervals numbers of IQWS are generated via multiplying the precise QoS values of each considered QoS parameter of QWS by the random interval number $[r_1, r_2]$ with $r_1 = 0.9 + 0.1 * rand(0,1)$, $r_2 = 1.0 + 0.1 * rand(0,1)$, and $rand(0,1)$ is a uniformly distributed random real number in the range $[0,1]$. Since the QWS dataset dos not contain the price parameter, and as we said previously, only the availability (q_1), throughput (q_2), response time (q_3) and price (q_4) attributes are considered to generate the IQWS dataset. Hence, the interval numbers of the price attribute have been randomly generated by $r3 \otimes [r_1, r_2]$ with $r3$ is a random generated real number from the range $[2,5]$\$. Moreover, the importance and priorities of the four considered QoS attributes were set to the same value (i.e, $\forall t \in \{1,2,3,4\}, w_{q_t} = \frac{1}{4}$) and each considered user's global QoS requirement Cst_{q_t} for the

q_t attribute was set as given in the following Equation.

$$Cst_{q_t} = \left[\max\left(r_1 * SU_{q_t}, min^l_{q_t}\right), \min\left(r_2 * SU_{q_t}, max^u_{q_t}\right)\right]$$

With

$$SU_{q_t} = \begin{cases} \mu_{q_t} * (max^u_{q_t} - min^l_{q_t}) + min^l_{q_t} & \text{if } q_t \in QoS^+ \\ max^u_{q_t} - \mu_{q_t} * (max^u_{q_t} - min^l_{q_t}) & \text{if } q_t \in QoS^- \end{cases} \quad (14)$$

Where $\mu_{q_t} \in [0, 1]$ is a severity factor, which is used to adjust the considered user's global QoS constraint Cst_{q_t}, and $max^u_{q_t}$ and $min^l_{q_t}$ are calculated as given by Equations 5 and 6, respectively.

Here, only two global QoS constraints Cst_{q_3} and Cst_{q_4} of the response time and price attributes are considered in solving IQSC by setting their severity factors μ_{q_3} and μ_{q_4} to 0.3 and 0.2, respectively, whereas, μ_{q_1} and μ_{q_2} of the availability and throughput properties are set to the zero values.

B. Parameters setting of the compared algorithms

To investigate the performance and the efficiency of our proposed EABC, we have compared it to that obtained using the PSO algorithm with skyline operator [15]. The latter was proposed to solve the QSC problem with non-ambiguous QoS parameters. Hence, we have adapted it to support interval utility calculations of solutions as defined in Equation 1. For simplicity, the extended version of the proposed PSO-based approach in [15] is named EPSO. For the parameters setting of the two compared approaches, EABC and EPSO share the same population size ($Z = 40$) and the same stopping criterion, which is the number of solutions evaluations that was set to 50000. However, for their appropriate parameters, the inertia weight (w) and the two accelerating coefficients (c_1 and c_2) of EPSO are set to $w = 0.8$ and $c_1 = c_2 = 2.0$, as they were recommended in their related reference [15]. Whereas, the $limit$ parameter of EABC was set to the value of 80.

EABC and EPSO are performed to solve five abstract composite services ACS^m_ns, with $(n, m) \in \{(5, 501), (10, 250), (15, 167), (20, 125), (25, 100)\}$, where each ACS^m_n consists of solving IQSC with n web services classes per m functionally identical web services that have been randomly selected from the 2507 ones of IQWS. The compared algorithms are implemented with Matlab R2016b and performed on the same personnel computer, which runs Windows 7 and has an Intel(R) Core(TM) i5-4570, CPU 3.20 GHz and 4 Go of memory as a hardware configuration. For each ACS^m_n, the compared EABC and EPSO algorithms are carried out 30 independent times to define the best, worst and average optimality values, and average running times of their obtained optimal/near-optimal solutions.

C. Comparison results discussion

For each solved ACS^m_n, the best, worst and average interval utility values of the obtained optimal/near-optimal solutions by EABC and EPSO have been written down in Table II, and the average running times to get these optimal solutions over 30 independent executions have been plotted in Fig. 4. As we can show from the results of solving the five ACS^m_ns listed in Table II, that our proposed EABC reach very higher interval optimality values compared to the ones of EPSO. Furthermore, as seen in Fig. 4, when the number of web services classes n was set with small values, i.e., $n \leq 10$, the average running times of EPSO are better than the ones of EABC. However, when n was set with large values, $n \geq 15$, our proposed EABC algorithm obtain optimal/near-optimal solutions with less average running times compared to ones of EPSO.

From this discussion, our EABC outperforms the compared EPSO, especially for solving users requests that need the composition of an important number of atomic web services.

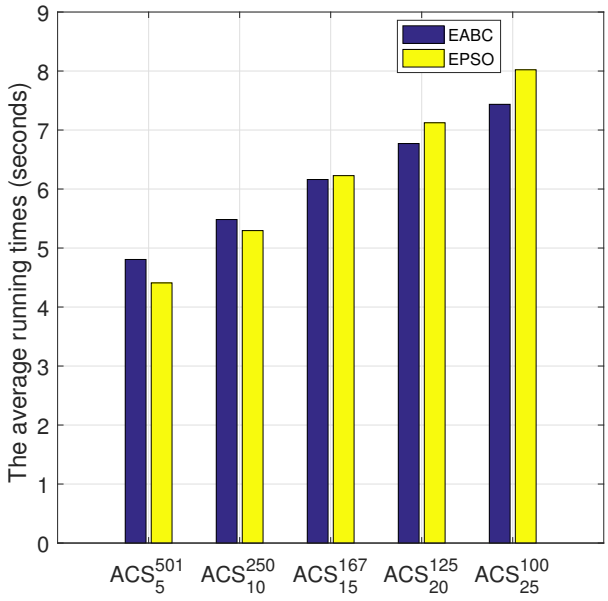


Fig. 4. The average running times of EABC and EPSO in solving ACS^m_ns of n web services classes per m functionally equivalent web services.

VII. Conclusion and Future Work

In this study, the interval number, which is a simple and general uncertain model, is used to represent the ambiguity of the QoS values in solving the QoS uncertainty-aware web service composition problem. The latter is modeled as an interval constrained single-objective optimization (IQSC) model, while a new approach that combines two components: skyline operator and interval extended version of the basic artificial bee colony (EABC) algorithm, is introduced to address the formulated IQSC. The first component (skyline operator) is used to reduce the search space of IQSC by pruning the redundant and dominated web services from their sets of functionally equivalent ones. Whereas, the second component (EABC) is performed to obtain the optimal/near-optimal composite service of IQSC in a reduced search space. The experimental results of comparing our proposed approach to an existing skyline-based PSO method, which have been performed on an interval extended version of the public QWS dataset, demonstrate and validate the performance of the introduced approach. As a future work, we are planning

TABLE II

COMPARISON RESULTS OF THE BEST, WORST AND AVERAGE INTERVAL UTILITY VALUES OF THE OBTAINED OPTIMAL/NEAR-OPTIMAL SOLUTIONS BY THE COMPARED APPROACHES IN SOLVING EACH ACS_n^m OF IQSC WITH n WEB SERVICES CLASSES PER m FUNCTIONALLY EQUIVALENT WEB SERVICES

ACS_n^m	EABC			EPSO		
	Best	***Worst***	***Average***	***Best***	***Worst***	***Average***
ACS_5^{501}	**[0.7075,0.8234]**	**[0.7043,0.7777]**	**[0.7023,0.8048]**	[0.6830,0.7962]	[0.6463,0.7486]	[0.6716,0.7697]
ACS_{10}^{250}	**[0.5891,0.7418]**	**[0.5795,0.7244]**	**[0.5887,0.7279]**	[0.5773,0.6802]	[0.5287,0.6336]	[0.5571,0.6614]
ACS_{15}^{167}	**[0.5529,0.7144]**	**[0.5437,0.6987]**	**[0.5515,0.7016]**	[0.5055,0.5985]	[0.4708,0.5555]	[0.4915,0.5701]
ACS_{20}^{125}	**[0.5274,0.6761]**	**[0.5212,0.6342]**	**[0.5195,0.6547]**	[0.4874,0.5421]	[0.4557,0.4932]	[0.4571,0.5243]
ACS_{25}^{100}	**[0.5161,0.6059]**	**[0.4993,0.5703]**	**[0.5169,0.5773]**	[0.4792,0.5260]	[0.4387,0.4767]	[0.4640,0.4949]

The best interval utility values are in boldface.

to solve the QoS uncertainty-aware web service composition problem for the Internet of Things (IoT) and Cloud computing environments that consider more specialized QoS parameters.

REFERENCES

[1] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on software engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[2] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient web service composition with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, p. 7, 2012.

[3] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Computational intelligence based qos-aware web service composition: A systematic literature review," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 475–492, 2017.

[4] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on software engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[5] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1069–1075.

[6] J. Liao, Y. Liu, X. Zhu, and J. Wang, "Accurate sub-swarms particle swarm optimization algorithm for service composition," *Journal of Systems and Software*, vol. 90, pp. 191–203, 2014.

[7] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Applied Intelligence*, vol. 42, no. 4, pp. 661–678, 2015.

[8] x. wang, X. Xu, Q. Z. Sheng, Z. Wang, and L. Yao, "Novel artificial bee colony algorithms for qos-aware service selection," *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 247–261, 2019.

[9] M. Razian, M. Fathian, and R. Buyya, "Arc: Anomaly-aware robust cloud-integrated iot service composition based on uncertainty in advertised quality of service values," *Journal of Systems and Software*, vol. 164, p. 110557, 2020.

[10] X. Jian, Q. Zhu, and Y. Xia, "An interval-based fuzzy ranking approach for qos uncertainty-aware service composition," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 4, pp. 2102–2110, 2016.

[11] F. Seghir, A. Khababa, and F. Semchedine, "An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain qos," *The Journal of Supercomputing*, 2019.

[12] H. Zheng, J. Yang, and W. Zhao, "Probabilistic qos aggregations for service composition," *ACM Transactions on the Web (TWEB)*, vol. 10, no. 2, p. 12, 2016.

[13] J. Xu, L. Guo, R. Zhang, H. Hu, F. Wang, and Z. Pei, "Qos-aware service composition using fuzzy set theory and genetic algorithm," *Wireless Personal Communications*, vol. 102, no. 2, pp. 1009–1028, 2018.

[14] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 11–20.

[15] S. Wang, Q. Sun, H. Zou, and F. Yang, "Particle swarm optimization with skyline operator for fast cloud-based web service composition," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 116–121, 2013.

[16] H. Ying and Z. Jiande, "A nonlinear service composition method based on the skyline operator," *Journal of Systems Engineering and Electronics*, vol. 31, no. 4, pp. 743–750, 2020.

[17] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings 17th international conference on data engineering*. IEEE, 2001, pp. 421–430.

[18] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 795–804.

[19] D. M. Gay, "The ampl modeling language: An aid to formulating and solving optimization problems," in *Numerical analysis and optimization*. Springer, 2015, pp. 95–116.

[20] A. K. Bhunia and S. S. Samanta, "A study of interval metric and its application in multi-objective optimization with interval objectives," *Computers & Industrial Engineering*, vol. 74, pp. 169–178, 2014.

[21] S. Mahato and A. Bhunia, "Interval-arithmetic-oriented interval computing technique for global optimization," *Applied Mathematics Research Express*, vol. 2006, 2006.

[22] S. Karmakar and A. K. Bhunia, "A comparative study of different order relations of intervals." *Reliable Computing*, vol. 16, no. 2, pp. 38–72, 2012.

[23] R. V. Rao, "Introduction to multiple attribute decision-making (madm) methods," *Decision Making in the Manufacturing Environment: Using Graph Theory and Fuzzy Multiple Attribute Decision Making Methods*, pp. 27–41, 2007.

[24] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[25] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[26] W.-L. Xiang and M.-Q. An, "An efficient and robust artificial bee colony algorithm for numerical optimization," *Computers & Operations Research*, vol. 40, no. 5, pp. 1256–1265, 2013.

[27] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.