Article

An Intelligent Bat Algorithm for Web Service Selection with QoS Uncertainty

Abdelhak Etchiali [1]*, Amina Bekkouche [1] and Fethallah Hadjila [1]

[1] Computer Science Department, University of Tlemcen, Tlemcen 13000, Algeria
* Correspondence: abdelhak.etchiali@univ-tlemcen.dz

Abstract: Nowadays, the selection of web services with uncertain quality of service (QoS) is gaining a lot of attention in the service-oriented computing paradigm (soc). In fact, searching for a service composition that fulfills a complex user's request is known to be NP-Complete. The search time is mainly dependent on the number of the requested tasks, the size the available services, and the size of the QoS realizations (i.e., sample size). To handle this problem, we propose a two-stage approach that reduces the search space using heuristics for ranking the tasks' services and a bat algorithm metaheuristic for selecting the final near optimal compositions. The fitness used by the metaheuristic aims to fulfill all the global constraints of the user. The experimental study shows that the ranking heuristics, termed "fuzzy pareto dominance" and "Zero-order stochastic dominance", are highly effective than the other heuristics and most of the existing state-of-the-art methods.

Keywords: Web service selection; QoS uncertainty; Bat algorithm; Service-oriented computing

1. Introduction

With the advent of cloud computing and specifically the online services (SaaS), it becomes more challenging to discover and select the best services with respect to user's requirements [1], [2]. Broadly speaking, we observe that a given functionality can be fulfilled by numerous SaaS services with a variety of QoS levels. For complex user's requests (in terms of workflow) the task of selecting the best composition of services that satisfies the user's global constraints (e.g., the maximum cost of the composition of services is less than a given budget) is time consuming and far from meeting the user's expectations. It is worth noting that the selection of service compositions is NP-complete and exponentially depends on the number of tasks of the workflow (see the example of Table I). In practice, we observe that the QoS of SaaS applications is inherently uncertain and always changing; for instance, the cost of booking a hotel chamber or an airline ticket is uncertain and depends on the period (such as the season or month), social events, and other contextual Aspects. To compare the services of the same functionality class while considering the different realizations of the QoS criteria, one can use statistical measures such as the mean QoS value or the median value to derive the best alternatives. Unfortunately, these measures may not be effective as it will be explained using the example of Table 1.

In the same line of thought, we point out that the pertinence of service compositions with respect to the user's request is no longer a deterministic score, but it is rather specified as a probability of satisfying the global QoS constraints; this score is termed Global QoS Conformance (GQC) [3]. As a result, the complexity of the selection issue is dependent on the number of tasks and also is impacted by both the size of each task and the size of the QoS sample (i.e., the number of realizations per QoS attribute).

GQC can be also seen as the expected value of the a random variable termed Z, where Z provides an outcome equal to 1 if the aggregated QoS satisfy the global constraint bound; moreover, it is highly desirable to get service compositions that satisfy a maximum number of global constraints in terms of median QoS (this means that 50% of the solution

Table 1. Motivating Example

G.C:$AggregatedQoS(S_x, S_y) \geq 45$	
Task T1	**Task T2**
S_1 : $QoS(S_1) =< 5, 15, 20, 30, 70 >$	S_9 : $QoS(S_9) =< 10, 11, 15, 20, 22 >$
S_2 : $QoS(S_2) =< 6, 18, 20, 40, 90 >$	S_{10} : $QoS(S_{10}) =< 15, 18, 26, 30, 40 >$
S_3 : $QoS(S_3) =< 4, 15, 18, 25, 250 >$	S_{13} : $QoS(S_{13}) =< 3, 8, 10, 20, 30 >$

realizations -of a single QoS attribute- will ensure the end-to-end bounds). This criterion is denoted as the percentage of satisfied global constraints (PSGC). This latter measure can be considered (sometimes) as an alternative to the GQC objective function, since it ensures a high gain of computational cost. For instance, according to Table I, where the global constraint (GC) is specified in the first line, we observe that only (s1,s10) and (S2,s10) will be retained as feasible solutions since PSGC=100% (the example is comprised of a single QoS attribute); however, the remaining compositions are not feasible, and therefore PSGC=0.

To summarize, our selection issue needs effective ranking heuristics for the workflow tasks, as well as time-efficient approaches for exploring the service compositions. To address these difficulties, we propose a two-stage approach that ensures high fitness service compositions and acceptable responsiveness delay.
In the first step, we reduce the search space in each task by only retaining the TopK pertinent services in terms of a given heuristic H_i. Consequently, the total search space is reduced from m^n candidate solutions to k^n candidate solutions, where n stands for the number of tasks and m stands for the number of services per task (see Table 2 for more details).
In the second step, we perform a heuristic global search (this means that our solution will be a vector of n services) and retain the TopK compositions in terms of GQC.
Our contributions can be summarized as follows:

- We downsize our search space from m^n to k^n by retaining the most pertinent elements of each task. To this end, we propose four ranking heuristics of the items of each task. All these heuristics perform pairwise comparisons of services and select TopK elements having the maximum number of wins.
  - H_1 is an efficient implementation of the fuzzy pareto dominance; it is inspired from [33].
  - H_2 (zero order stochastic dominance) is a stochastic dominance relationship that uses the zero order terms of the QoS sample [34]. it directly uses the QoS realizations during the comparisons.
  - H_3 (first order stochastic dominance) is a stochastic dominance relationship that uses the first order terms of the QoS sample [34]; this means that H_3 uses the cumulative distribution of the sample to perform comparisons.
  - H_4 (the majority interval heuristic) is inspired from [19]. In this ranking, we compute the median interval of each service and perform pairwise comparisons of services using equation 27. the services having the highest number of wins are retained in TopK elements.
- In the second step, we perform a global search on the retained TopK services using a swarm intelligence-based algorithm termed "discrete bat algorithm (DBA)". This metaheuristic is chosen because of its ability to leverage both global search operators and local search operators during the exploration of candidate solutions (in contrast to metaheuristics that only use one operator such as particle swarm optimization or ant

colony optimization). The coordinated use of these operators can achieve promising results on NP-complete problems.

- At the end, We evaluate the effectiveness and the efficiency the approach using a consolidated set of experiments.

The rest of this paper is organized as follows: Section 2 presents a literature overview of the existing works. Section 3 specifies the problem statement. In Section 4, we introduce the proposed approach as well as the selection algorithms. In section5, we present a set of experimental evaluations and compare our method with existing works. Section 6 concludes the paper and presents future perspectives.

2. State of the Art

Selecting service compositions using QoS is a major topic in service-oriented computing (SOC), We mainly distinguish two categories: service selection with certain (deterministic) QoS and service selection with uncertain (nondeterministic) QoS, we will review the to parts in what follows.

2.1. Service Selection with Certain QoS

In this category, we assume that the QoS attributes are static and do not change overtime, therefore the evaluation function of compositions is also deterministic. Many works and reviews have been proposed to address this kind of issues ([1], [4], [5],[6]). In what follows, we will discuss the most important ones.

[7] proposed a framework that first takes the skyline services of each task; then, a set of service clusters (within each task) are hierarchically created using K-means to lower the size of the search space. At the end, the solutions are explored using the combinations of cluster-heads. The work by [8] decomposes global QoS constraints into local constraints using culture genetic algorithm, then the top items are selected to aggregate the final compositions.

[4] adopted both functional (the function signature) and nonfunctional attributes (QoS, global constraints) to select the Top-K service compositions. The authors leveraged harmony search to derive the compositions that best meet the complex requirements.

[9] used the multi-criteria decision method termed Topsis to search the most promising services in terms of QoS. The proposal leveraged six QoS criteria with different workflow patterns and constructs.

[10] leveraged both local and global search for tackling the selection of cloud services. The local selection allows for downsizing the search space, while the global selection allows for keeping near-optimal compositions.

[11] tackled the optimal selection of web services by adopting both PetriNet models and skyline search; these authors also used the R- tree structure to accelerate the search for Pareto-optimal solutions.

In [5], the authors used an optimized artificial bee colony (oabc) method for service composition. Mainly, the authors introduced three ideas in the initial bee algorithm: the first one is the diversification of the initial population; the second one is the dynamic adjustment of the neighborhood size of the local search; the third one is the addition of a global movement operator that aims to get closer to the global solution. The work by [12] leverages fuzzy dominated scores to derive the TopK services that have more balanced QoS (and which can be better than some skyline services with undesirable QoS values) in a self-contained task. [13] considered self-organising migrating algorithm (SOMA) and fuzzy dominance relationship to aggregate service workflows. The fuzzy dominance function is used in the SOMA meta heuristic to compute the QoS aware distances between services. [14] proposed a bio-inspired method termed enhanced flying ant colony optimisation (EFACO). This approach constrains the flying activity and handles the execution time problem by a modified local selection. Since this phase may degrade the selection quality, a multi-pheromone approach is adopted to enhance the exploration through the pheromone assignment to each QoS criterion.

[15] clustered the cloud services using a trust-oriented k-means, then they created the composition of cloud services using honey bee mating. It is worth noting that the proposed framework is not scalable for large datasets. The work by [16] tackled the service selection problem by handling multiple users' requirements. The approach is comprised of two steps: firstly, an approximate pareto-optimal set is computed using approximate dominance; secondly, the near optimal compositions are selected using artificial bee colony algorithm.

2.2. Service Selection with Uncertain QoS

The work of [17] is one of the earliest works that addresses the service selection with uncertain QoS. The authors proposed a excellent heuristic termed, P-dominant skyline, to derive the best QoS aware services in a self-contained task. P-dominant skyline is considered to be resilient to QoS inconsistencies and noise. Moreover, this heuristic is accelerated using R-trees. [18] adopted probability distributions to model the QoS uncertainty of service workflows. To select the best compositions, the authors used both integer programming and global constraints penalty cost functions.

[19] proposed a promising heuristic to derive the pertinent services of local tasks using majority intervals. The main idea consists of computing the median interval of each nondeterministic QoS attribute and comparing them using rectified linear unit functions (RelU) [20]. After that, an exhaustive search is applied to get the final compositions. In [21], the authors proposed a set of heuristics for ranking the services of the workflow tasks. These propositions include probabilistic dominance relationships and fuzzy dominance alternatives. Once the TopK elements are retained from each task, a constraint programming approach is applied to retain the TopK optimal compositions of services. [22] addressed the service composition issue by handling the QoS uncertainty and the location awareness. They proposed a sophisticated approach that combines the firefly metaheuristic with a fuzzy logic-based web service aggregation.

The framework proposed in [23] sorted the services of each task using both entropy and variance of the QoS attributes, the services that have larger values in terms of entropy and variance are discarded since they are considered as noisy or inconsistent services. Then, the items having the lowest entropy/variance scores were retained to compose the final solutions.

[3] is one of the first works that handled of QoS uncertainty and composition. Based on ideas defined in [24], the strategy adopted by the authors consists of decomposing the end-to-end constraints into local constraints; the local edges (entrances) are calculated by dividing the end-to-end constraint bounds in proportion to the aggregated median QoS of each class of the workflow. After that, an initial service composition is built using a predefined utility function. If this latter one is not optimal, the method searches for alternative solutions using simulated annealing. In the same line of thought, [25] introduced a proposition for Web service selection with presence of outliers. Contrary to the work of [3], this method leverages a different heuristic to divide the end-to-end constraints into local constraints. The proposed idea ensures a high resilience with respect to outliers (services with noisy or unusual QoS). The work by [26] leverages stochastic dominance relationship to sort the services of each task of the user's workflow; after that, a backtracking search is applied to the filtered tasks to derive optimal service compositions. [27] proposed an interval-based multi-objective bee colony method to address the uncertain QoS-aware service composition problem. The authors proposed an interval-oriented dominance relationship for comparing the services using intervals that represent the variation range of QoS attributes. In addition an interval-valued utility function is introduced to assess the quality of a composition with QoS uncertainty. Finally, an improved version of NSGA-II is used to derive the non-dominated service compositions. The framework proposed in [28] involves two steps: the first one retains the pertinent services of the local tasks using majority grades, and the second step performs a constraint programming search to keep the optimal compositions. In the same line of thought, the work by [29], proposes a heuristic

for filtering the desirable services of each local task using hesitant fuzzy sets and cross entropy, then a metaheuristic termed grey wolves optimization is applied to retain the TopK near optimal service compositions.

3. Problem Specification

In what follows, we introduce the formalism used in handling the selection of service compositions with QoS uncertainty.

3.1. Parameters' Notation

To tackle our problem, we use the notation shown in Table 2. We assume that the user's workflow is composed of n sequential tasks cl_1, cl_2,...,cl_n, each task is achieved by a service s_i that has r QoS attributes. Each QoS criterion is materialized by a sample of l realizations (see Table 2 and Figure 1).

Table 2. Notations

Parameter	Semantic
n	The number of Tasks (classes).
m	The number of services per task.
r	The number of QoS Criteria.
l	The number of QoS realizations (i.e., or the sample size).
cl_1, cl_2,...,cl_n	The set of Tasks, each tasks involves atomic SaaS services with the same functionality and different QoS.
s_1 (resp. s_2,...,s_m)	represents the id of the selected service related to cl_1 (resp. cl_2,...,cl_n)
QoS_{piju}	The value of the p^{th} QoS attribute related to the u^{th} instance of the service $S_i \in cl_j$.
b_1,b_2,..b_r	The user's global constraints (i.e., the bounds that need to be satisfied by the QoS of the composition).
w_1,..w_r	The weight of the QoS attributes, the default value of each w_p is $\frac{1}{r}$.
k	The size of the outcome list (of compositions).

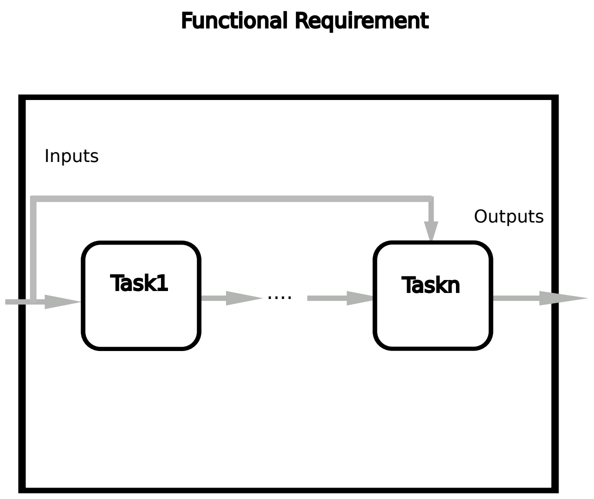


Figure 1. A general sequential workflow.

3.2. QoS model

In this work, we will only consider positive QoS attributes (i.e., those that need to be maximized). For negative attributes, we will simply multiply them by -1 and treat the new versions as positive ones. We note that our workflow is composed of n sequential tasks. The aggregated QoS of a workflow (having different patterns such as sequence, loops, parallelism, and choice) is presented in [7], [30].

3.3. Global QoS Conformance

The measure of Global QoS Conformance (GQC) [3] is leveraged to rank the TOPK compositions. GQC is the probability that the composition of services satisfy all global constraints (see Equation 1). In particular, we say that a composition C is better than another composition $C^{'}$ if the GQC of C is higher than that of $C^{'}$ with respect to Equation 1. If C ties with $C^{'}$, then we sort them according to the utility ($U(.)$) function that is shown in Equation 4, the larger the score of $U(.)$, the better the rank.
Our aim is to search the compositions $C(s_{w1}, ..., s_{wn})$ such that GQC is maximized:

$GQC((S_{w_1}, \cdots, S_{w_n}), (b_1, \cdots, b_r)) =$

$$\prod_{p=1}^{r} CC((S_{w_1}, \cdots, S_{w_n}), b_p) \tag{1}$$

Since we assume that the QoS criteria are independent, the global QoS conformance is defined as the product of constraint conformances (CC for short).

The criterion CC is defined as:

$CC((S_{w_1}, \cdots, S_{w_n}), b_p) =$

$$\frac{1}{l^n} \sum_{u_1=1}^{l} ... \sum_{u_n=1}^{l} step(aggregate(QoS_{pw_1u_1}, \cdots, QoS_{pw_nu_n}), b_p) \tag{2}$$

The function CC computes the satisfaction degree of a single global constraint. Finally, the binary function “Step” is defined as:
$Step(Aggregate(QoS_{pw_1u_1}, \cdots, QoS_{pw_nu_n}), b_p) =$

$$\begin{cases} 1 & \text{if } Aggregate(s_{pw_1u_1}, \cdots, QoS_{pw_nu_n}) \geq b_p \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$U(C) = \sum_{p=1}^{r} w_p * \frac{(MedianQ_p^{'}(C) - Qmin^{'}(p))}{(Qmax^{'}(p) - Qmin^{'}(p))} \tag{4}$$

$$Qmin^{'}(p) = \sum_{j=1}^{n} Qmin(j,p) \tag{5}$$

$Qmin^{'}(p)$ is the minimal aggregated QoS of the p^{th} attribute for all possible compositions.

$$Qmax^{'}(p) = \sum_{j=1}^{n} Qmax(j,p) \tag{6}$$

$Qmax^{'}(p)$ is the maximal aggregated QoS of the $p^{\ th}$ attribute for all possible compositions. The Equations $Qmin(j,p), Qmax(j,p)$ are defined as follows :

$$Qmin(j,p) = Min_{u\in\{1,\dots,l\}, s_i \in cl_j}(QoS_{piju}) \tag{7}$$

$Qmin(j,p)$ is the minimal QoS value of the p^{th} attribute of all services related to the i^{th} task.

$$Qmax(j,p) = Max_{u\in\{1,\dots,l\}, s_i \in cl_j}(QoS_{piju}) \tag{8}$$

$Qmax(j,p)$ is the maximal QoS value of the p^{th} attribute of all services related to the i^{th} task.

$$MedianQ_p^{'}(C) = \sum_{j=1}^{n} Median_{u\in\{1,\dots,l\}} QoS_{ps_jju} \tag{9}$$

By assuming that the criterion p is positive, the global constraint with respect to the median value is specified as:

$$MedianQ'_p(C) \geq b_p; \forall p \in \{1,\dots,l\} \tag{10}$$

By assuming that the $p^{\ th}$ attribute is aggregated with a sum function, $MedianQ_p^{'}(C)$ represents the aggregated QoS of C with respect to the median QoS value of each component of C (of the p^{th} attribute).

Equation 10 is used to determine whether the global constraints are respected or not by the composition C.
To clarify the computation of the previous equations, we continue with the example cited in Table I:

- $MedianQ'_p(C =< s_1, s_{10} >) = 20 + 26 = 46 \geq 45.$

- $GQC(C) = 16/25 = 0.64$
  If $Qmin(1,1) = Qmin(2,1) = 0$ and $Qmax(1,1) = Qmax(2,1) = 300$, then

- $U(C) = \frac{46-0}{(600-0)} = 0.075$
  The composition C is feasible.
  However, if the components of C' are $< s_3, s_{13} >$ then:

- $MedianQ'_p(C' =< s3, s_{13} >) = 18 + 10 = 28 \leq 45.$
- $GQC(C') = 9/25 = 0.36$
- $U(C') = \frac{28-0}{(600-0)} = 0.046.$
  The composition C' is not feasible because it violates the global constraint.

4. Proposed Approach

In what follows, we present the architecture of the proposed solution as well as the different implemented algorithms. (see Figure 2).

4.1. Overall Architecture

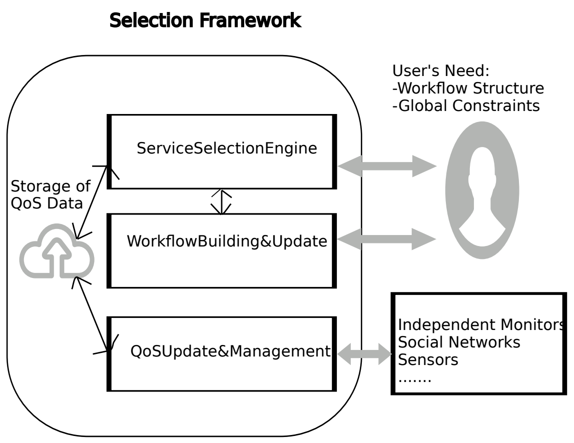


Figure 2. Service selection architecture.

Our proposed framework involves three principal parts:

- The workflow building and update module: Its goal is to assign the new services to their corresponding tasks (a task is a functionality available on the internet, e.g., hotel booking). This component also updates the tasks by changing/removing the services.
- The QoS update and management module: It stores all the QoS realizations of all services in a data-warehouse; The QoS information may stem from different sources such as social networks (e.g., ratings, fidelity), third parties (e.g., throughput, latency), and service providers (e.g., cost).
- The QoS aware service selection engine: Given a user's workflow and the set of global constraints, the selection module allows to search the Top-K pertinent service compositions. As mentioned in the sequel, this engine achieve two steps: a local optimization (or sorting) and a global optimization. The first phase (local optimization) uses a set of heuristics (see Equations 15, 24, 20, and 28) to rank the services of each task. The primordial goal is to downsize the search space by only keeping the first k services in the next phases.
  The second phase of the engine performs a global optimization on the previous results. This step is realized using a discrete bat algorithm.

4.2. Local Optimization

In the following, we introduce four heuristics $(H_1, H_2, H_3, and H_4)$ that retain a subset (of size k) of each task. These services are the most promising items in terms of each H_i. In this work, we assume that the higher the value of a QoS level, the better the service.

4.2.1. Fuzzy Pareto Dominance Heuristic (H1)

Many alternatives are available for implementing the fuzzy version of pareto dominance [31], [21], [32], and [33]. To compare 02 r-dimensional vectors u_d and v_d, we use the implementation specified in [33] since it is slightly effective than the remaining alternatives and has zero hyper-parameters (in contrast to the others). Its definition is given in 12. The elementary fuzzy dominance (EFD) compares two scalar QoS values using Equation 11.

$$EFD(u_d(j), v_d(j)) = \begin{cases} 1 & \text{if } u_d(j) \geq v_d(j) \\ \frac{MIN(u_d(j), v_d(j))}{v_d(j)} & \text{otherwise} \end{cases} \tag{11}$$

$$FD(u_d, v_d) = \prod_{i=1}^{l} EFD(u_d(i), v_d(i)) \tag{12}$$

We assume that u_d and v_d, represent the values of the d^{th} QoS attribute of two existing services S and S' (respectively). To compare S and S' with respect to all QoS attributes, we use Equation 13 (Aggregated fuzzy Dominance or AFD for short).

$$AFD(u, v) = \prod_{d=1}^{r} EFD(u_d, v_d) \tag{13}$$

The fuzzy contest function shown in Equation 14 (FC for short) inspects the fuzzy dominance power of a service w with respect to another service q.

$$FC(S_w, S_q) = \begin{cases} 1 & \text{if } AFD(S_w), S_q) \geq AFD(S_q, S_w) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Equation 15 computes the sorting score of a service S_w by achieving a comparison with the rest of candidate services of the current task (the larger the score, the better the rank).

$$FD\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} FC(S_w, S_q) \tag{15}$$

In the experimental study, we will sort the services of each task according to the decreasing order of Equation 15 and take first k elements.

We illustrate the principle of H1 by comparing the services S_1 and S_2 of Table I:

If we apply Equation 13, we get
$AFD(QoS(S_1), QoS(S_2)) = FD(QoS(S_1), QoS(S_2)) = \frac{5}{6} \times \frac{15}{18} \times \frac{30}{40} \times \frac{70}{90} = 0.40$.
On the other hand: $AFD(QoS(S_2), QoS(S_1)) = 1$ Consequently, $FC(S_1, S_2) = 0, FC(S_2, S_1) = 1, FC(S_2, S_3) = 1,$
$FD\_SCORE(S_2) = 1.$

4.2.2. Zero Order Stochastic Dominance (H2)

This heuristic compares the services using the raw QoS values [34] (see Equation 16).

$$ZSD(u_d, v_d) = \frac{1}{l} \sum_{i=1}^{l} Step(u_d(i), v_d(i)) \tag{16}$$

$$Step(u_d(i), v_d(i)) = \begin{cases} 1 & \text{if } \ u_d(i) \geq v_d(i) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

To compare two S and S' with respect to all QoS attributes, we use Equation 18 (Aggregated zero order stochastic dominance or AZSD for short).

$$AZSD(u, v) = \prod_{d=1}^{r} ZSD(u_d, v_d) \tag{18}$$

To perform the majority vote (within a task), we need to compare each pair of services. To do so, we leverage the contest function shown in Equation 19, it is termed Aggregated zero order stochastic dominance contest (AZSDC) AZSDC returns 1 if S_w dominates S_q (in the sense of AZSD), otherwise, it returns 0.

$$AZSDC(S_w, S_q) = \begin{cases} 1, & \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Equation 20 calculates the sorting score of a service S_w by achieving a comparison with the rest of candidate services of a given task (the larger the score, the better the rank).

$$ZSD\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} AZSDC(S_w, S_q) \tag{20}$$

In the experiments, we will sort the services of each task according to the decreasing order of Equation 20 and take first k elements.

4.2.3. First Order Stochastic Dominance (H3)

Like H2, the first order stochastic dominance (H3) performs the same steps, except that it processes the cumulative distribution (CumulDistr) of the sample instead of the raw QoS. If we assume that u_d is the QoS sample of the d^{th} attribute of a given service S, then the cumulative distribution of u_d is approximated as follows:

$u'_d(i) = CumulDistr_i(u_d) = \sum_{t=1}^{i} \frac{1}{l}$.

In addition, we increase the resolution (size) of u'_d and set it to $2 \times l$, the added entries (i') will have a score equal to $\frac{u_d(i-1)+u_d(i)}{2}, \wedge i' \in [i-1, i]$.

$$\begin{aligned} FSD(u'_d, v'_d) =& ZSD(CumulDistr(u_d), CumulDistr(v_d)) \\ =& \frac{1}{2 \times l} \sum_{i=1}^{2\times l} Step(u'_d(i), v'_d(i)) \end{aligned} \tag{21}$$

We have the same expressions mentioned in H2 for the rest of equations.

$$AFSD(u', v') = \prod_{d=1}^{r} FSD(u'_d, v'_d) \tag{22}$$

$$AFSDC(S_w, S_q) = \begin{cases} 1, & \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

$$FSD\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} AFSDC(S_w, S_q) \tag{24}$$

In the experiments, we will sort the services of each task according to the decreasing order of Equation 24 and take first k elements.

4.2.4. Majority Interval Dominance (H4)

In this heuristic, we first compute the median interval for each QoS attribute of each service (this means that the d^{th} of each service S_x is represented with an interval $[lb_{x,d}, ub_{x,d}]$). then, we rank the services by comparing these representative intervals. To elucidate this idea, we consider the services S1 and S2 of Table I. The median interval of S1 is [5,30], and the corresponding one of S2 is [18,40]. To compare the median intervals, we use the function presented in [19]; this function is defined in Equation 24 and it is termed majority interval dominance (MID). (we assume that the compared services $S_x$and S_y belong to the task j, and the current QoS attribute is d, S_x is represented with $[a_1, a_2]$ and and S_y is represented with $[b_1, b_2]$).

$$MID([a_1, a_2], [b_1, b_2]) = \frac{Relu(a_1 - b_1) + Relu(a_2 - b_2)}{2 \times (Qmax(j, d) - Qmin(j, d))} \tag{25}$$

Where Relu (Rectified linear unit) [20] is the activation function used in deep learning. For instance, if we assume that $Qmax(j, d) = 300, Qmin(j, d) = 0$, then $MID(S_1, S_2) = MID([5, 30], [18, 40]) = 0$, and $MID(S_2, S_1) = MID([18, 40], [5, 30]) = \frac{13+10}{2\times300} = 0.038$.
The aggregated majority interval dominance is shown in Equation 26.

$$AMID(u, v) = \prod_{d=1}^{r} MID(u_d, v_d) \tag{26}$$

u_d, v_d represent the median intervals of the compared QoS attributes (having the d^{th} rank). Like H1, H2, and H3, the contest function is defined in Equation 27

$$AMIDC(S_w, S_q) = \begin{cases} 1, & \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

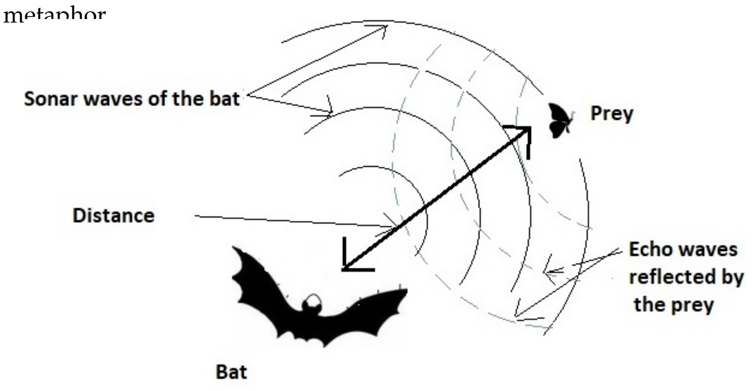
$$MID\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} AMIDC(S_w, S_q) \tag{28}$$

In the experiments, we will sort the services of each task according to the decreasing order of Equation 28 and take first k elements.

4.3. Global Optimization

Once the n lists are given by the first step of the method, it is time now to perform a global search by composing and assessing the service compositions. To do so, we leverage a swarm intelligence metaheuristic that adapt the bat algorithm to our discrete context. This discrete optimization algorithm is chosen because of its ability to combine local search and global search in a harmonious way. Bat algorithm [35] is a promising metaheuristic for continuous optimization. Its metaphor is based on the echolocation behaviour of micro-bats, that can vary frequencies, loudness, and pulse rates of emission to capture the prey (see Figure 3).

Figure 3. Bat metaphor.



Before giving the pseudo-code of the discrete bat algorithm, we explain all its technical parameters.
Pop: it is a matrix of PopSize*n dimensions, it represents all the virtual bats. $Pop = \{Bat_1, .., Bat_{PopSize}\}$.

Bat^*: the position of the best bat.
A: it stands for the loudness of the chirp; it is a vector of Popsize random numbers comprised in [0,1], it controls the neighborhood size of the local search. It is decreased along

the execution of the metaheuristic.
Freq: it stands for the frequencies of the bats. It is a matrix of PopSize*n dimensions, it controls the size of the moving step during the global search phase. It is initialized with random values between 0 and 1.
R: it stands for the pulse emission rate of each bat. Technically, it is a n-dimension vector of random numbers (in [0,1]) that controls the execution of the local search.
Alpha: the factor of decreasing A.
Gamma: it is a factor that controls the increasing rate of the pulse emission rate R.
MaxIt: the maximum number of iterations of DBA.

Algorithm 1: Discrete Bat Algorithm (DBA)

```
    Input:
    $< TopKList_1, ..., TopKList_n >$: the input lists given by the local optimization
    heuristics.
    GC: the global constraints bounds.
    k: the size of the result list
    Output:
    TopKCompositions: the Top-K compositions that best meet tall global constraints
    in terms of GQC (it is initially empty).
1   $A \leftarrow ones(PopSize)$
     $R \leftarrow random(PopSize)$
     $Alpha \leftarrow 0.8$
     $Gamma \leftarrow 0.8$
     for $i \leftarrow 1$ to $PopSize$ do
         $Bat_i \leftarrow RandomPosition(TopKList_1, .., TopKList_n)$ $Freq_i \leftarrow random()$
2   end
3   $Bat^* \leftarrow ArgMax_{i \in \{1,..,PopSize\}}(GQC(Bat_i))$
4   while $(it \leq MaxIt)$ do
5       for $i \leftarrow 1$ to $PopSize$ do
6           $Bat_i \leftarrow GlobalMovment(Bat_i, Freq_i, Bat^*)$ $Freq_i \leftarrow random()$
             if $(random() \geq R_i)$ then
7               $neighborhoodSize \leftarrow round(k * mean_{i \in \{1,..,PopSize\}}(A_i))$
                 $NewPosition \leftarrow neighbor(Bat^*, neighborhoodSize)$
8           end
9           if $(random() \leq A_i$ and $GQC(NewPosition) \geq GQC(Bat_i))$ then
10              $Bat_i \leftarrow NewPosition$ $A_i \leftarrow Alpha \times A_i$ /*decrease the loudness rate*/ ;
11              $R_i \leftarrow 0.01 \times (1 - exp(-Gamma \times it)$ /*increase the pulse emission
                 rate*/ ;
12          end
13          if $GQC(Bat_i) \geq GQC(Bat^*)$ then
14              $Bat^* \leftarrow Bat_i$;
15          end
16      end
17      $it \leftarrow it + 1$
18  end
19  $TopKCompositions \leftarrow update(TopKCompositions, \{Bat^*, Bat_1, Bat_2, ..., Bat_n\})$
     return $TopKCompositions$
```

The pseudo-code of DBA can be explained as follows: Line 1 : for each bat, we initialize its loudness, pulse emission rate, and their updating rates Alpha and Gamma.
Line 2: for each bat, we randomly initialize its position and its frequency that it is used as a step displacement in the GlobalMovment (of line 6). $Freq_i$ is a real value belonging to [0,1].
Line 3: we compute the best bat position of the swarm in terms of GQC. we update the best bat position of the swarm.
Lines 4-18 : this is the principal loop of the metaheuristic; it is constituted of MaxIt itera-

tions.
Lines 5-16 : this is the loop that explores all the bats.
Line 6: this function creates a new composition by moving toward the best solution with a random step. More specifically, for each component (task) j of a given bat i, we replace it with the corresponding value in bat^* with a probability equal to $Freq_i(j)$ ($Bat_i(j) = Bat^*(j)$, with a probability= $Freq_i(j)$). The frequency of each bat is changed after that.
Line7:with a probabiity $1 - R_i$, we create a neighborhood centered on the best bat Bat^*. The width of this neighborhood is equal to k times the average of all the possible loudnesses A_i (this width is termed as spread); then, we create a new composition **NewPosition**= $(component_1, ..., component_n)$ as follows:
for each $j \in \{1, ..., n\}$ $component_j = successor_{Task_j}(Bat^*(j))$ with a probability $= Gaussian_{mean,\sigma}(|Rank(Bat^*(j)) - Rank(successor_{Task_j}(Bat^*(j)))|)$. Knowing that $Mean = Rank(Bat^*(j))$ and $\sigma = spread/2$.
For instance, if a task j is constituted of the following ranked services $< S9, S15, S4, S20, S2 >$, and we assume that $Bat^*(j) = S4, mean = Rank(Bat^*(j)) = 3$ (it is ranked third in the list), and $\sigma = spread/2 = 1$, then the neighborhood of S4, according to line 7, is equal to $\{S15, S4, S20\}$. The probability of getting each of them as a value for $component_j$ is $25\%, 50\%, 25\%$, respectively (since we approximate the Gaussian function for these three observations).
In lines 9-12, we accept the aforementioned solution **NewPosition** (i.e., we update Bat_i), with a probability A_i. In addition, **NewPosition** must have a fitness better than that of Bat_i. We decrease the loudness A_i and increase the pulse emission rate R_i in order to reduce the chances of performing the local search in the future (i.e., line 7).
In lines 13-15, we update the best solution if the actual bat has a better fitness.
Finally, we notice that DBA has a time complexity of $O(PopSize + n \times PopSize + PopSize \times r \times l^n + Maxit \times PopSize(n + n \times k + r \times l^n))$. We notice that the complexity of the fitness function GQC is $O(r \times l^n)$.

5. Experimental Study

Inspired from [3] and [25], we generate the QoS dataset using a random Gaussian distribution. In particular, we use the following setting: mean=0 and standard-deviation=1. The domain of each parameter is given in Table 3.

The experiments were implemented using a Window10 64 bit OS with Intel Core i3-6006U CPU @ 2.0GHz processor and 32 GB RAM. The algorithms where developed with netbeans IDE 12.0.

Before introducing the experimental results, we describe the theoretical complexity of the proposed heuristics. The heuristic H1 (Equation 15) compares each candidate service with the remaining components and each comparison step (Equation 13) is $O(r.l)$; therefore, the time complexity of H1 is $O(m.r.l)$. Like H1, the complexity of H2 (Equation 20) is $O(m.r.l)$. In the same line of thought, the complexity of Equation 24 (H3) is $O(m.r.l)$ and the complexity of Equation 28 (H4) is $O(l.logl + m.r)$.

In the experiments, we will only vary one parameter and keep the remaining set to their default values (see Table 3). As regards the fuzzy dominance implementation of [21], we preserve the same setting chosen by the authors for the parameter ε (which is equal to 0.1). For the sake of concise presentation, we only show the Top2 pertinent compositions (in terms of GQC) for all the remaining experiments.

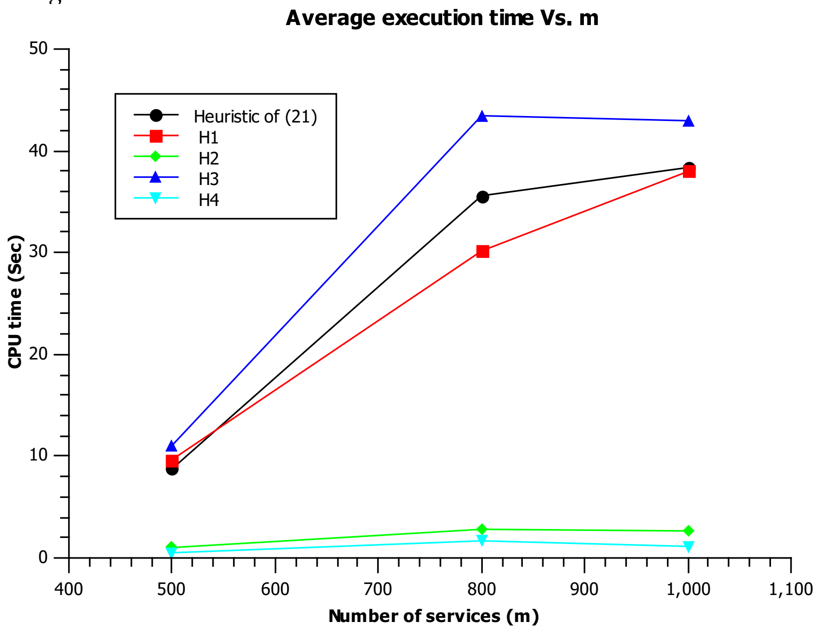
As shown in Figure 4, we observe that the behavior (time) of H1, H3, and the heuristic of (21) is comparable. Additionally, we observe a slight rise of time for H3 since the curve slope is proportional to $2 \times l$ instead of l. At the end, we note that H2 and H4 are the most efficient heuristics since the slope of their curves is lower than the that of first ones.

The Figure 5 shows that the CPU time of H1, H3, and the heuristic of (21) is comparable, but the slopes of their respective curves are different. Additionally, we observe a

Table 3. Parameters' Range

Parameter	Meaning	Domain	Default value
n	The number of tasks	{2,5,8}	2
m	The number of services per class	{500,..,1000}	500
r	The number of QoS attributes	{4,..,10}	4
l	The number of realizations of a given QoS attribute (i.e., the number of instances)	{15,..,100}	21
k	The size of the returned list	{2,5,10}	5
b_i	The i^{th} global constraint bound	Positive real	For attributes aggregated with: an additive function: n*0.6. a multiplicative function: 0.6^n. MAX/MIN functions: 0.6.
w_i	The weight of the i^{th} QoS attribute	[0..1]	1/r

Figure 4. Average CPU time Vs. m



slight rise of the time for the heuristic of (21) since its complexity is quadratic with respect to l. We note that H4 is the most efficient one since the comparison of median intervals does not depend on l (we assume that the sorting of QoS vectors is done in an offline way).

Like the previous experiments, Figure 6 shows that the fuzzy dominance implementation of (21), H1, and H3, have closer CPU times. On the other hand, the majority grade heuristic (28) and H4 have a lower CPU time since their theoretical slope is not dependent on l. We also note that the curve of H2 has an almost flat slope and this is mainly due to the low overhead of Equation 16. It is worth noting that, the majority grade principle is initially presented by [36] for ranking the candidates of an election. After that, it is adapted by [28] to web service selection.

Figure 5. Average CPU time Vs. l

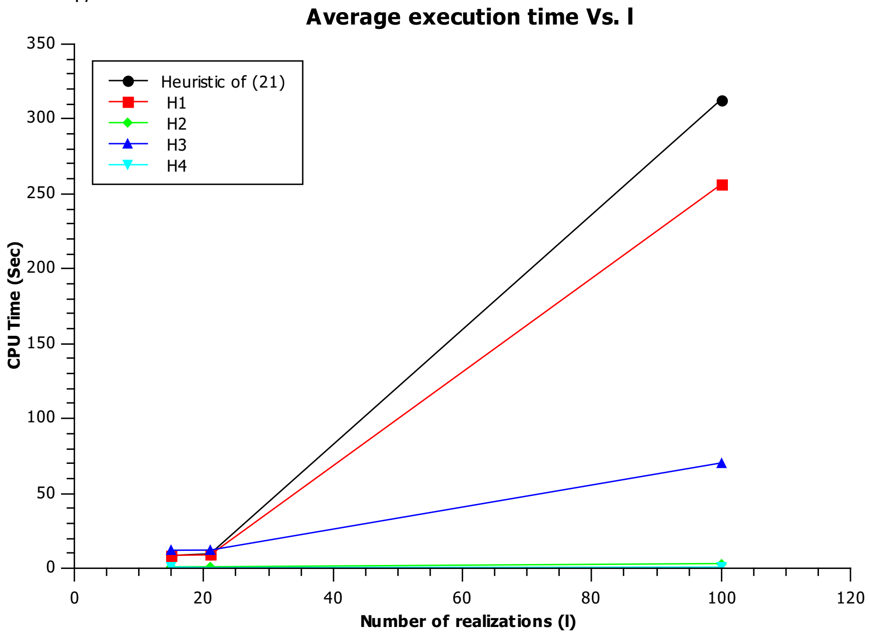
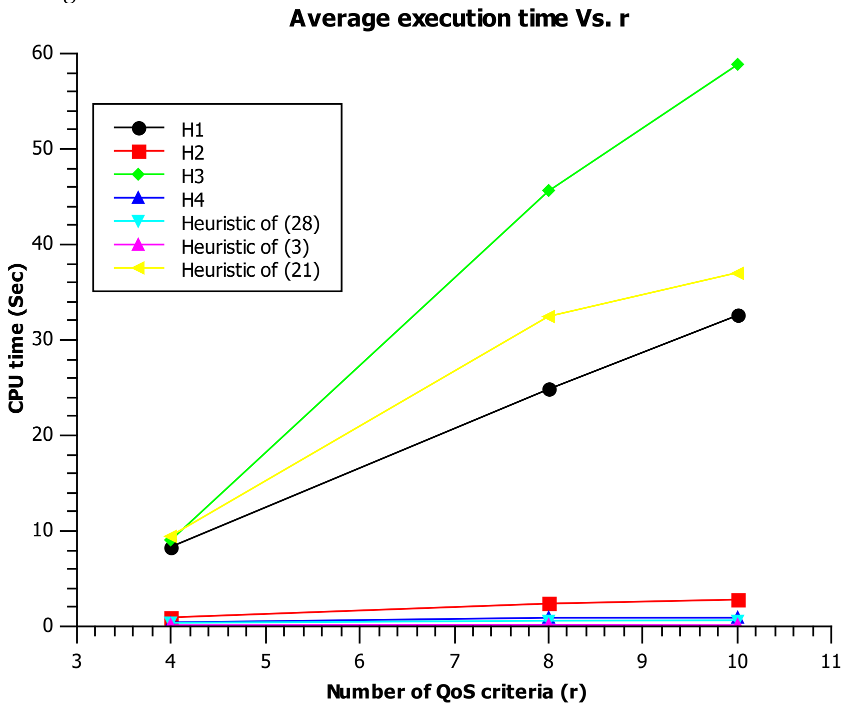


Figure 6. Average CPU time Vs. r



According to Figure 7, we observe that all methods have almost the same CPU time up to n=5. Behind this threshold, the time rises with different scales (according to each alternative). We notice that the exhaustive search is the most prohibitive one since there is an exponential number of candidate solutions; however, DBA (with all configurations) only explores a polynomial number of candidate compositions (but GQC is still exponential). As a result, the increase rate of time is less drastic for the three configurations of DBA. In summary, we can state that a selection problem with less than 8 tasks can be efficiently handled with DBA while using less than 100 bats. It is worth noting that the majority of real-world workflows have less than 10 abstract tasks, and this fact highlights the suitability of DBA to the QoS aware service selection problem.

Figure 7. Average CPU time for DBA and Exhaustive Search

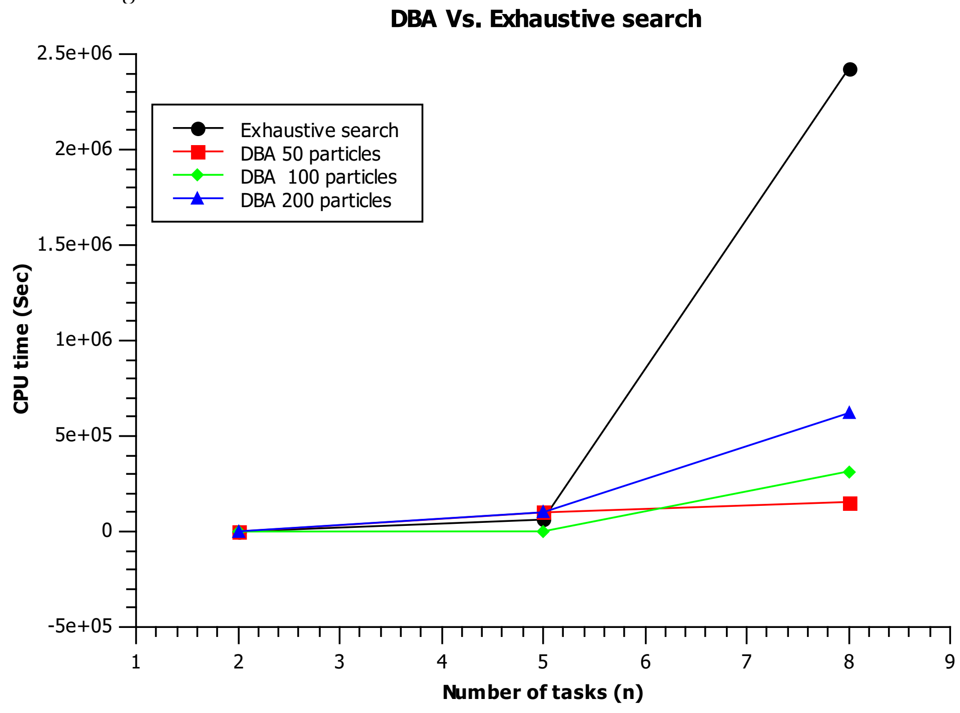


Table 4. GQC and Global Constraints Satisfiability Vs. r

	r=4		r=8		r=10	
Model	GQC	PSGC	GQC	PSGC	GQC	PSGC
H_1	**0.673**	**75%**	**0.484**	**50%**	**0.500**	**40%**
	0.646	**75%**	**0.480**	**62.5%**	**0.480**	**20%**
H_2	**0.721**	**100%**	**0.515**	**37.5%**	**0.570**	**30%**
	0.664	**100%**	**0.515**	**37.5%**	**0.463**	**30%**
H_3	0.302	0%	0.393	0%	0.388	0%
	0.253	0%	0.343	0%	0.356	0%
H_4	0.562	50%	0.6628	12.5%	0.408	10%
	0.486	50%	0.524	25%	0.388	20%
Fuzzy dominance heuristic of (21)	0.673	75%	0.5155	37.5%	0.500	50%
	0.633	50%	0.515	50%	0.441	10%

Table 5 shows the behavior of the heuristics with respect to the QoS sample size l. Broadly speaking, we notice that both GQC and PSGC degrade as the l grows. This degradation is logic since the satisfaction of tight global constraints will be rare as l increases.We observe that H1 and H2 are more effective than the remaining heuristics; more specifically H2 performs better than H1 for low values of l (we can even obtain 100% of PSGC), however H1 performs better for medium and large values of l. In contrast to the heuristics H2, H3, and H4, we observe that H1 has a stable and consistent performance for all values of l.

Table6 presents the performance of the heuristics with respect to m (the cardinal of the task). We observe a slight degradation for both GQC and PSGC when the number of services m increases, (for almost all heuristics). This observation may be due to the fact that the new extended dataset have less promising QoS levels. We also notice that the heuristics H1 and H2 are more effective than the rest of alternatives (for all values of m).

Table 5. GQC and Global Constraints Satisfiability Vs. l

	l=15		l=21		l=100	
Model	GQC	PSGC	GQC	PSGC	GQC	PSGC
H_1	**0.673**	**75%**	**0.655**	**100%**	**0.420**	**0%**
	0.646	**75%**	**0.544**	**50%**	**0.414**	**0%**
H_2	**0.721**	**100%**	**0.704**	**50%**	**0.408**	**0%**
	0.664	**100%**	**0.655**	**100%**	**0.402**	**0%**
H_3	0.302	0%	0.343	0%	0.346	0%
	0.253	0%	0.311	0%	0.324	0%
H_4	0.562	50%	0.538	25%	0.392	0%
	0.486	50%	0.467	25%	0.390	0%
Fuzzy dominance heuristic of (21)	0.673	75%	0.665	75%	0.415	0%
	0.633	50%	0.588	75%	0.411	0%

Table 6. GQC and Global Constraints Satisfiability Vs. m

	m=500		m=800		m=1000	
Model	GQC	PSGC	GQC	PSGC	GQC	PSGC
H_1	**0.673**	**75%**	**0.645**	**75%**	**0.549**	**50%**
	0.646	**75%**	**0.626**	**75%**	**0.491**	**25%**
H_2	**0.721**	**100%**	**0.604**	**50%**	**0.552**	**50%**
	0.664	**100%**	**0.583**	**75%**	**0.486**	**75%**
H_3	0.302	0%	0.358	0%	0.379	0%
	0.253	0%	0.311	0%	0.299	0%
H_4	0.562	50%	0.638	50%	0.506	25%
	0.486	50%	0.620	25%	0.474	25%
Fuzzy dominance heuristic of (21)	0.673	75%	0.590	75%	0.551	50%
	0.633	50%	0.583	75%	0.551	50%

Table7 demonstrates the performance of the heuristics with respect to the number of tasks n. It is clearly shown that the scores given by all heuristics degrade with the increasing of n, since it is more difficult to satisfy a constraint comprised of a larger sum of random variables (according to the central limit theorem, this sum will follow – under some conditions- a Gaussian probability distribution with a narrower standard deviation). Like the precedent experiments, we notice that H1 performs better than the rest of heuristic for all values of n. Additionnaly, we note that H2 has a better GQC and PSGC for low values of n, but these scores drastically degrade when n increases.

Table 8 presents a comparison between our contributions (DBA with H1 and H2) and some existing state-of-the-art approaches. It is clearly shown that GQC and PSGC of H1 and H2 are more effective than the works of the literature. We also observe that the work of [3] gives the lowest values for GQC, and this means that the methods based on local thresholds selection have worse performances on practical datasets. We also observe that the fuzzy implementation of the pareto-dominance using [33] is better than that of [21], since the experiments shown in tables 6, 5, 4,and 7 confirm the slight superiority of our proposed formula.

6. Conclusion

We have presented in this paper a set of ranking heuristics coupled with a bat algorithm metaheuristic for selecting service compositions with uncertain QoS. The main idea of the proposition consists of lowering the size space by first retaining the most pertinent services in each class (task) using well defined heuristics. In the second phase, we perform a

Table 7. GQC and Global Constraints Satisfiability Vs. n

	n=2		n=5		n=8	
Model	GQC	PSGC	GQC	PSGC	GQC	PSGC
H_1	**0.673**	**75%**	**0.665**	**50%**	**0.609**	**50%**
	0.646	**75%**	**0.656**	**50%**	**0.592**	**75%**
H_2	**0.721**	**100%**	**0.647**	**75%**	**0.224**	**0%**
	0.664	**100%**	**0.640**	**50%**	**0.219**	**0%**
H_3	0.302	0%	0.163	0%	0.132	0%
	0.253	0%	0.161	0%	0.126	0%
H_4	0.562	50%	0.557	25%	0.512	25%
	0.486	50%	0.541	25%	0.511	25%
Fuzzy dominance heuristic of (21)	0.673	75%	0.563	50%	0.590	50%
	0.633	50%	0.557	50%	0.580	50%

Table 8. Utility Score, Global Constraints Satisfiability, and GQC for all methods (default configuration)

Heuristic	GQC	US	PSGC
H_1	**0.655**	**0.531**	**75%**
	0.544	**0.482**	**100%**
H_2	**0.704**	**0.511**	**50%**
	0.655	**0.531**	**100%**
H_3	0.342	0.387	0%
	0.311	0.374	0%
H_4	0.538	0.451	25%
	0.467	0.427	25%
Majority grade with constraint programming (28)	0.703	0.519	75%
	0.631	0.527	75%
Fuzzy dominance heuristic of (21)	0.665	0.516	75%
	0.588	0.514	75%
First assignment of (3)	0.302	0.398	0%

global search to get the best compositions in terms of global QOS conformance. The results confirm the ability of both fuzzy pareto dominance relationship and stochastic dominance (of order zero) to outperform the remaining heuristics.
In future works, we plan to test the framework on other types of workflows and compare our bat algorithm method with recent metaheuristics such as Spider Monkey Optimization and whale optimization algorithm.

References

1. Hayyolalam V, Kazem AA. A systematic literature review on QoS-aware service composition and selection in cloud environment. Journal of Network and Computer Applications. 2018 May 15, 110:52-74.
2. Merzoug M, Etchiali A, Hadjila F, Bekkouche A. Effective Service Discovery based on Pertinence Probabilities Learning. International Journal of Advanced Computer Science and Applications. 2021,12(9).
3. Hwang, S. Y., Hsu, C. C., and Lee, C. H. Service selection for web services with probabilistic QoS. IEEE transactions on services computing, 2015,8(3): 467-480.
4. Bekkouche, A., Benslimane, S. M., Huchard, M., Tibermacine, C., Hadjila, F., and Merzoug, M. QoS-aware optimal and automated semantic web service composition with user's constraints. Service Oriented Computing and Applications, 2017. 11(2): 183-201.

5. Zhang S, Shao Y, Zhou L. Optimized artificial bee colony algorithm for web service composition problem. Int. J. Mach. Learn. Comput. 2021 Sep;11(5).
6. Mohammed, M., Chikh, M. A., and Fethallah, H. QoS-aware web service selection based on harmony search. in IEEE ISKO-Maghreb: Concepts and Tools for knowledge Management (ISKO-Maghreb), 4th International Symposium , 2014, pp. 1-6.
7. Alrifai, M., Risse, T., and Nejdl, W. A hybrid approach for efficient Web service composition with end-to-end QoS constraints. ACM Transactions on the Web (TWEB),2012, 6(2): p7.
8. Liu, Z. Z., Jia, Z. P., Xue, X., An, J. Y. Reliable Web service composition based on QoS dynamic prediction. Soft Computing, 2015, 19: 1409-1425.
9. Belouaar, H., Kazar, O., and Rezeg, K. Web service selection based on TOPSIS algorithm. in 2017 IEEE International Conference on Mathematics and Information Technology (ICMIT),2017. pp. 177-182.
10. Shetty, J., D'Mello, D. A. Global and local optimisation-based hybrid approach for cloud service composition. in International Journal of Computational Science and Engineering,2018, 17(1): 1-14.
11. Chen, L., Ha, W. Reliability prediction and QoS selection for web service composition. In International Journal of Computational Science and Engineering,2018, 16(2): 202-211.
12. Halfaoui A, Hadjila F, Didi F. QoS-aware web services selection based on fuzzy dominance. InComputer Science and Its Applications: 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, May 20-21, 2015, Proceedings 5 2015 (pp. 291-300). Springer International Publishing.
13. Halfaoui, A., Hadjila, F., et Didi, F. QoS-aware web service selection based on self-organising migrating algorithm and fuzzy dominance. International Journal of Computational Science and Engineering, 2018, 17(4): p. 377-389.
14. Dahan F, El Hindi K, Ghoneim A, Alsalman H. An enhanced ant colony optimization based algorithm to solve QoS-aware web service composition. Ieee Access. 2021,9(2021):34098-111.
15. Zanbouri K, Jafari Navimipour N. A cloud service composition method using a trust-based clustering algorithm and honeybee mating optimization algorithm. International Journal of Communication Systems, 2020,33(5):e4259.
16. Zhu, W., Yin, B., Gong, S., Cai, K. Y. An Approach to Web Services Selection for Multiple Users, 2017, IEEE Access, 5:15093-15104.
17. Yu, Q., and Bouguettaya, A. Computing service skyline from uncertain qows. IEEE Transactions on Services Computing, 2010, 3(1): 16-29.
18. Schuller, D., Lampe, U., Eckert, J., Steinmetz, R., and Schulte, S. (2012, June). Cost-driven optimization of complex service-based workflows for stochastic QoS parameters. in 2012 IEEE 19th International Conference on Web Services (ICWS) ,2012, pp. 66-73.
19. Abdelhak E, Feth-Allah H, Mohammed M. QoS uncertainty handling for an efficient web service selection. InProceedings of the 9th International Conference on Information Systems and Technologies, 2019, pp. 1-7.
20. Brownlee J. A gentle introduction to the rectified linear unit (ReLU). Machine learning mastery. 2019 Jan 9;6.
21. Hadjila F, Belabed A, Merzoug M. Efficient web service selection with uncertain QoS. International Journal of Computational Science and Engineering. 2020, 21(3):470-82.
22. Rajeswari P, Jayashree K. Hybrid Metaheuristics Web Service Composition Model for QoS Aware Services. Comput. Syst. Sci. Eng.. 2022 Jan 1;41(2):511-24.
23. Sun, L., Wang, S., Li, J., Sun, Q., Yang, F. QoS uncertainty filtering for fast and reliable web service selection. In 2014 IEEE International Conference on Web Services, 2014, pp. 550-557.
24. Sun SX, Zhao J. A decomposition-based approach for service composition with global QoS guarantees. Information Sciences. 2012 Sep 15;199:138-53.
25. Kim, M., Oh, B., Jung, J., and Lee, K. H. Outlier-robust web service selection based on a probabilistic QoS model. International Journal of Web and Grid Services, 2016, 12(2): 162-181.
26. Yasmina RZ, Fethallah H, Fedoua D. Selecting web service compositions under uncertain QoS. InComputational Intelligence and Its Applications: 6th IFIP TC 5 International Conference, CIIA 2018, Oran, Algeria, May 8-10, 2018, Proceedings 6 2018 (pp. 622-634). Springer International Publishing.
27. Seghir F, Khababa A, Semchedine F. An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain QoS. The Journal of Supercomputing. 2019,75(2019):5622-66.

28. Zeyneb Yasmina R, Fethallah H, Fadoua L. Web service selection and composition based on uncertain quality of service. Concurrency and Computation: Practice and Experience. 2022.34(1):e6531.
29. Yasmina RZ, Fethallah H. Uncertain service selection using hesitant fuzzy sets and grey wolf optimisation. International Journal of Web Engineering and Technology. 2022,17(3):250-77.
30. Zheng, H., Zhao,W., Yang,J., Bouguettaya, A. QoS Analysis for Web Service Compositions with Complex Structures. IEEE Trans. Services Computing, 2013, 6(3): 373-386.
31. Benouaret, K., Benslimane, D., Hadjali, A. On the use of fuzzy dominance for computing service skyline based on qos. In 2011 IEEE International Conference on Web Services (ICWS), 2011, pp. 540-547.
32. Wang G, Jiang H. Fuzzy-dominance and its application in evolutionary many objective optimization. In IEEE 2007 International conference on computational intelligence and security workshops (CISW 2007), 2007,pp. 195-198.
33. Köppen M, Vicente-Garcia R, Nickolay B. Fuzzy-pareto-dominance and its application in evolutionary multi-objective optimization. InEvolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings 3, 2005, pp. 399-412. Springer Berlin Heidelberg.
34. Bruni, R., Cesarone, F., Scozzari, A., Tardella, F. On exact and approximate stochastic dominance strategies for portfolio selection. European Journal of Operational Research, 2017, 259(1), 322-329.
35. Yang XS. A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010). 2010:65-74.
36. Balinski M, Laraki R. A theory of measuring, electing, and ranking. Proceedings of the National Academy of Sciences. 2007 May 22,104(21):8720-5.