RAIRO-Oper. Res. 53 (2019) 445–459
https://doi.org/10.1051/ro/2017049

**RAIRO Operations Research**
www.rairo-ro.org

# MULTI-OBJECTIVE AND DISCRETE ELEPHANTS HERDING OPTIMIZATION ALGORITHM FOR QOS AWARE WEB SERVICE COMPOSITION

Samia Chibani Sadouki[1,*] and Abdelkamel Tari[1]

**Abstract.** The goal of QoS aware web service composition (QoS-WSC) is to provide new functionalities and find a best combination of services to meet complex needs of users. QoS of the resulting composite service should be optimized. QoS-WSC is a global multi-objective optimization problem belonging to NP-hard class given the number of available services. Most of existing approaches reduce this problem to a single-objective problem by aggregating different objectives, which leads to a loss of information. An alternative issue is to use Pareto-based approaches. The Pareto-optimal set contains solutions that ensure the best trade-off between conflicting objectives. In this paper, a new multi-objective meta-heuristic bio-inspired Pareto-based approach is presented to address the QoS-WSC, it is based on Elephants Herding Optimization (EHO) algorithm. EHO is characterised by a strategy of dividing and combining the population to sub population (clan) which allows exchange of information between local searches to get a global optimum. However, the application of others evolutionary algorithms to this problem cannot avoids the early stagnancy in a local optimum. In this paper a discrete and multi-objective version of EHO will be presented based on a crossover operator. Compared with SPEA2 (Strength Pareto Evolutionary Algorithm 2) and MOPSO (Multi-Objective Particle Swarm Optimization algorithm), the results of experimental evaluation show that our improvements significantly outperform the existing algorithms in term of Hypervolume, Set Coverage and Spacing metrics.

**Mathematics Subject Classification.** 68T20.

Received October 13, 2016. Accepted June 10, 2017.

## 1. Introduction

With the emergence of cloud and Software as a Service (SaaS), more and more web services became available. Thus, a large number of web services with the same functionalities and different Quality of Service (such as Price, Response Time, Availability, Reliability, Reputation, Security, Throughput, *etc.*) can be found. New functionalities can be provided by combining available services to satisfy complex user's requirements that are not satisfied by a single web service. Hence we notice the appearance of a new emerging challenge as QoS aware web service composition (QoS-WSC) [26,28]. QoS-WSC is a multi-objective optimization problem (MOP), since, different QoS attributes have to be optimized simultaneously [26]. Existing methods mainly use a fitness function to reduce this problem to a single-objective problem. These approaches suffer from conflicting

*Keywords.* QoS, Multi-Objective optimization, Pareto Set, Bio-inspired Algorithms, Elephants Herding optimization, Web service composition.

[1] LIMED Laboratory, University of Bejaia, Faculty of Exact Sciences, Computing Department, 06000, Bejaia, Algeria.
*Corresponding author: samia_chibani@yahoo.fr

objectives as improving one objective may causes the degradation of others. In these approaches the end user is also supposed to have a complete knowledge of its preferences about the desired solutions. In addition, only one solution can be obtained in each execution. Multi-objective optimization is performed to find a set of solutions. Even though Pareto-based approaches offer a higher accuracy to identify the subset of solutions that matches the user's preferences, very few of them addresses QoS-WSC. In this paper, a novel Pareto-based approach for multi-objective optimization called MO-D-EHO is proposed using EHO (Elephants Herding Optimization). EHO is a swarm-based metaheuristic search method, inspired by the herding behaviour of elephants group [19, 20]. In EHO, each elephant implements clan updating operator to change its position based on its current position and the matriarch's (female leader) position in the corresponding clan where the worst elephant is replaced by a separating operator.

The contribution of this paper is summarized in two points: first, we propose improvement of EHO algorithm by a new clan updating operator (crossover operator) which gives a discrete version of EHO (D-EHO), then we give a multi-objective version of D-EHO for addressing the QoS-WSC problem.

The rest of the paper is organized as follows: Section 1 defines the problem and overview of recent research efforts. Section 2 discusses the QoS-WSC problem, the multi-objective optimization problem and gives a description of EHO algorithm. The proposed approach is presented in Section 3. Finally, experimental studies are described in Section 4, conclusions and future work are outlined in Section 5.

## 2. Related work

Traditional optimization techniques such as integer linear programming, multidimensional multiple-choice knapsack and graph theory, have been reported in [25] to address the QoS-WSC problem. Most studies adopt single-objective optimization to handle multiple objectives. These approaches are very efficient when the size of the problem is small. However, these methods suffer from poor scalability due to the exponential time complexity.

Recently, Evolutionary Computation methods have been used to overcome the drawbacks of traditional optimization methods. Genetic Algorithm (GA) [4, 5, 9], Ant Colony Optimization (ACO) [7, 8, 22] and Particle Swarm Optimization (PSO) [18, 21] were used to find optimal service composition in a reasonably short time frame, when a large number of web services and multiple QoS attributes are concerned. However, with Applying these algorithms the problem of early stagnancy in a local optimum cannot be avoided. Moreover, these methods reduce only this problem to a single-objective problem by aggregating all objectives using a fitness function (*e.g.*: the weighted sum approach and fraction-based approaches).

As mentioned before, single-objective optimization approaches have some inherent limitations in solving QoS-WSC. Therefore, few multi-objective heuristics [15, 16, 23] have been proposed. A Non dominated Sorting Genetic Algorithm-II (NSGA-II) is proposed in [23], authors in [15] use the Strength Pareto Evolutionary Algorithm 2 (SPEA2), while a Multi Objective Particle Swarm Optimization (MOPSO) is proposed in [16]. A comparative analyse of multi-objective algorithms for QoS-WSC is given in [6].

To address the multi-objective QoS-WSC problem, we use a swarm based meta-heuristic search method, called EHO [19, 20] which is inspired by the herding behaviour of elephants group. The strength of EHO algorithm is its strategy of dividing and combining the population of clans to allow the exchange of information between local searches to get global optimum. EHO is a mono-objective optimization technique for continuous problem. In order to adapt EHO to QoS-WSC, we propose a discrete version of EHO using a multi-objective optimization strategy.

## 3. Background

In this section, we will give some concepts and definitions related to problem under study such as QoS-aware web service composition, multi-objective optimization problem and the elephants herding optimization algorithm.

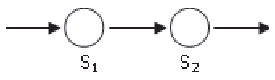

FIGURE 1. Sequential composite relation.

## 3.1. QoS-Aware Web service composition

### *3.1.1. Web service*

A web service is an abstract resource representing a capability of performing tasks that form a coherent functionality from the point of view of providers and requesters entities. To be used, a service must be realized by a provider to be a concrete web service [12]. A Web Service $S_i$ is a triple $S_i =< I, O, VQS_i >$ where:

- $I = \{I_1, I_2, \ldots\}$ is a set of inputs of the service $S_i$.
- $O = \{O_1, O_2, \ldots\}$ is a set of outputs of the service $S_i$.
- $VQS_i$ a set of parameters about quality of the web service $S_i$.

### *3.1.2. Quality of service (QoS)*

Quality of Service (QoS) is an indicator to measure and describe some performance characteristics of a service such as Response Time, Availability, Price, Reputation *etc.* [13]. Let $VQS_i$ a set of quality attributes of the service $S_i$, $VQS_i = \{VQS_i(q), q = 1 : nbq\}$, where $VQS_i(q)$ determines the value of the $q$th quality attribute of $S_i$ and $nbq$ is the number of quality attributes.

### *3.1.3. Web service composition*

Web service composition comes from software reuse. Its basic idea is to combine existing web services according to a certain relation to construct a new or better web service to satisfy complex user's requirements [14].

A composite service $CS$ is a triple $CS =< S, R, VQC >$ where:

- $S = \{S_i, i = 1 : nbc\}$ is a subset of registered abstract services, which can be composed together and satisfy the user's needs. $S_i$ may be an atomic service or a composite service. Each web service $S_i$ has a set of concrete web services, offering the same functionalities with different quality attributes' values, designed by a class of services, $nbc$ is the number of abstract services.
- $R = \{., !, \bigoplus, \bigotimes\}$ is a set of operators. A composite service can be constructed from several component services $S_i$ using four basic composition operators, where (.) is a sequential operator, (!) is a loop operator, ($\bigotimes$) is a parallel operator and ($\bigoplus$) is a conditional operator.
  - $S_1.S_2$ indicates that the services $S_1$ and $S_2$ are composed in sequence, it means that service $S_2$ is processed after $S_1$ has performed completely and $I(S_2) \in O(S_1)$ *i.e.* the $S_1$ outputs include the inputs of $S_2$, Figure 1.
  - $!S_i$ means that a service $S_i$ is executed $k$ times before the next task is executed, Figure 2. A loop operator can be regarded as a special sequential composition.

$$!S_i = \underbrace{S_i.S_i \ldots S_i}_{k\,times}$$

  - $S_2 \bigoplus S_3$ is the selective (conditional) composition of services. For instance, Figure 3 illustrates a composition of four services, where $S_2$ is selected with probability $p_1$ and $S_3$ is selected with probability $p_2$; $p_2 = (1 - p_1)$. $I(S_2) \in O(S_1); I(S_3) \in O(S_1); I(S_4) \in O(S_2); I(S_4) \in O(S_3)$.
  - $S_2 \bigotimes S_3$ is the parallel composition of services. In Figure 4, $S_2$ and $S_3$ process at the same time, $(I(S_2) \cup I(S_3)) \in O(S_1); I(S_4) \in (O(S_2) \cup O(S_3))$.

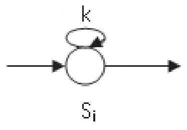

FIGURE 2. Loop composite relation.

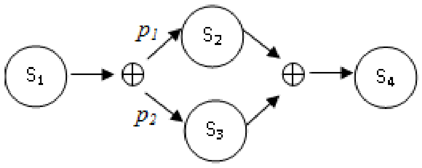

FIGURE 3. Conditional composite relation.

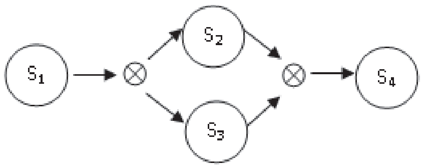

FIGURE 4. Parallel composite relation.

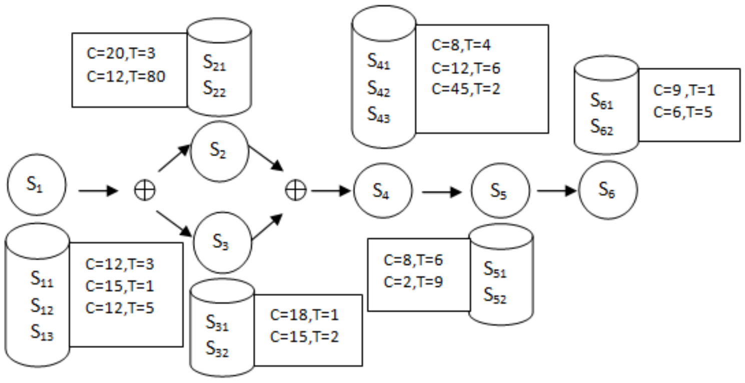

FIGURE 5. Example of workflow with bindings between abstract and concrete services.

Web service composition model can be specified as a workflow of a set of abstract services. At run time, a concrete web service is selected, and invoked for each abstract service.
Figure 5 illustrates an example of workflow with 6 abstract services, and for each one, a concrete web service is selected with appropriates quality values of C: Cost and T: response Time ($nbq = 2$).

- $VQC = \{VQC(q), q = 1 : nbq\}$ is the QoS of the composite service $CS$, $VQC(q)$ is the value of the $q$th quality attribute of $CS$ and $nbq$ is the number of quality attributes. $VQC(q)$ for different composition operators is given in Table 1.
  $VQC(q)$ for a composite service must be derived from aggregating the corresponding attributes of all the component services involved. As used in different aggregation approaches, we divide the QoS attributes into three different categories: Additive, Multiplicative and Max-operator attributes.

Table 1. The aggregation of attributes for different composition structures

| Attributes | Sequential | Parallel | Conditional | Loop |
|---|---|---|---|---|
| Additive | $\sum_{i=1}^{n} VQS_i(q)$ | $\sum_{i=1}^{n} VQS_i(q)$ | $\sum_{i=1}^{n} VQS_i(q) * p_i$ | $n * VQS_i(q)$ |
| Multiplicative | $\prod_{i=1}^{n} VQS_i(q)$ | $\prod_{i=1}^{n} VQS_i(q)$ | $\sum_{i=1}^{n} VQS_i(q) * p_i$ | $(VQS_i(q))^n$ |
| Max-operator | $\sum_{i=1}^{n} VQS_i(q)$ | $Max(VQS_i(q))$ | $\sum_{i=1}^{n} VQS_i(q) * p_i$ | $n * VQS_i(q)$ |

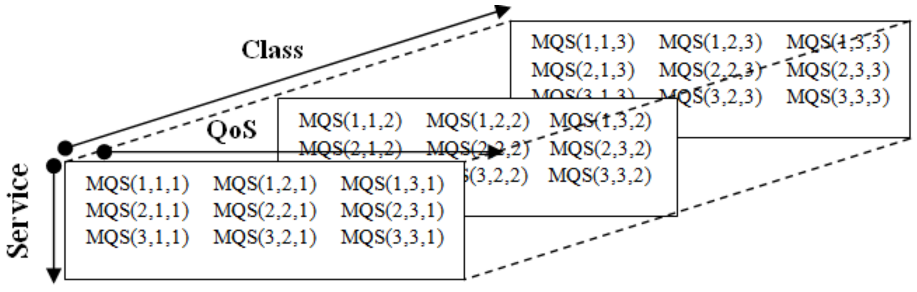
Class
QoS
Service
MQS(1,1,1) MQS(1,2,1) MQS(1,3,1)
MQS(2,1,1) MQS(2,2,1) MQS(2,3,1)
MQS(3,1,1) MQS(3,2,1) MQS(3,3,1)
MQS(1,1,2) MQS(1,2,2) MQS(1,3,2)
MQS(2,1,2) MQS(2,2,2) MQS(2,3,2)
MQS(3,2,2) MQS(3,3,2)
MQS(1,1,3) MQS(1,2,3) MQS(1,3,3)
MQS(2,1,3) MQS(2,2,3) MQS(2,3,3)
MQS(3,1,3) MQS(3,2,3) MQS(3,3,3)

Figure 6. Multidimensional matrix for the QoS.

$VQS_i(q)$ is the value of the $q_{th}$ attribute for the component service $S_i$ and $p_i$ is the probability for selecting the service $S_i$ in a conditional composition.

- Additive attributes: For an attribute such as price, we can determine its value for a composite service through summation.
- Multiplicative attributes: For attributes such as reliability and availability, the QoS value of composite service $CS$ can be determined by multiplying the values of all attributes of the involved component services.
- Max-operator attributes: The attributes, such as response time and service execution time, differ from those discussed above in that a maximum aggregation function is used for the parallel composition structure.

The implementation of the QoS for all concrete services is given as multidimensional (three-dimensional) matrix, where the concrete services representing the row index, the different attributes of QoS representing the column index, and the service class representing the page index.

Figure 6 implements a three-dimensional matrix $MQS$ with three concrete services, three QoS attributes and three abstract services, $MQS(s, q, c)$ is the $q$th QoS attribute value and $s$ the index of the concrete service in a service class $c$.

A normalization of the QoS attributes' values to the same scale is performed in order to avoid inaccurate evaluation due to different measurement metrics used for different QoS attributes. The QoS attributes can be classified into two groups: Positive and Negative attributes. The values of positive attributes need to be maximized (Availability, Reliability, Throughput), while the values of negative attributes must to be minimized (Price, Response time). In the normalization phase, positive and negative QoS attributes are scaled as follow:

Positive attributes:

$$MQS'(s, q, c) = \frac{MQS(s, q, c) - Q\min(q)}{Q\max(q) - Q\min(q)} \tag{3.1}$$

Negative attributes:

$$MQS'(s, q, c) = \frac{Q\max(q) - MQS(s, q, c)}{Q\max(q) - Q\min(q)} \tag{3.2}$$

Where $Qmax(q)$, $Qmin(q)$ are the maximum and minimum values of the $q$th attribute and $MQS(s, q, c)$ is the value of the $q$th attribute for a selected candidate service s in the class c.

Positive attributes can be transformed into negative attributes by multiplying their values by $-1$. So we transform the positive attributes into negative in order to minimise all the attributes.

## 3.2. Multi objective optimization problem

In multi-objective QoS-WSC problem, several QoS attributes have to be optimized simultaneously. Conflicting objectives can occur as a service may have a lower cost but a higher response time whereas another one may be more expensive but faster. Pareto approaches give a solution as a Pareto-optimal set ensuring the best trade-offs between the conflicting objectives.
In this section we formally define a multi-objective optimization problem (MOP) and we introduce its specific terminology. An MOP with $m$ decision variables and $n$ objectives can be formally defined as [24]:

$$\text{Min}(y = f(x) = [f_1(x), \dots, f_n(x)]) \tag{3.3}$$

where $x = (x_1, \dots, x_m) \in X$ is an $m$-dimensional decision vector, $X$ is the search space, $y = (y_1, \dots, y_n) \in Y$ is the objective vector and Y the objective space. We provide some definitions of Pareto concepts used in MOP for a minimisation problem as follows:

(i) Pareto dominance. For two decision vectors $x_1$ and $x_2$, dominance (denoted by $\prec$) is defined as follows:

$$x_1 \prec x_2 \Leftrightarrow \forall i \in [1:n], f_i(x_1) \leq f_i(x_2) \wedge \exists j \in [1:n], f_j(x_1) < f_j(x_2) \tag{3.4}$$

The decision vector $x_1$ is said to dominate $x_2$ if and only if, $x_1$ is as good as $x_2$ considering all objectives and $x_1$ is strictly better than $x_2$ in at least one objective.

(ii) Pareto optimally. A decision vector $x_1$ is said to be Pareto-optimal if and only if:

$$\nexists x_2 \in X : x_2 \prec x_1 \tag{3.5}$$

(iii) Pareto-optimal set. The Pareto-optimal set $PS$ is the set of all Pareto-optimal decision vectors.

$$PS = \{x_1 \in X, |\ \nexists x_2 \in X : x_2 \prec x_1\}. \tag{3.6}$$

(iv) Pareto-optimal front. The Pareto-optimal front $PF$ is the image of the Pareto-optimal set in the objective space.

$$PF = \{f(x) = (f_1(x), \dots, f_n(x)) | x \in PS\}. \tag{3.7}$$

$x_1$ is said to be non-dominated regarding a given set if $x_1$ is not dominated by any decision vector in the set.

The Pareto-optimal decision vector cannot be enhanced in any objective without causing degradation in at least another objective. A decision vector is said to be Pareto-optimal when it is not dominated in the whole search space.

## 3.3. Elephants herding optimization (EHO)

Elephants Herding Optimization (EHO) [19, 20] is a swarm-based meta-heuristic search method proposed to solve optimization problem which is inspired by the herding behaviour of elephants. Elephants are social in nature and in an elephants' group we have several clans. Elephants belonging to different clans live together under the leadership of a matriarch. Male elephants remain solitary and leave their family group while growing up. The behaviour of elephants herding is modelled by clan updating and separating operator. In EHO, each elephant implements clan updating operator to change its position based on its current position and the matriarch's position in the corresponding clan. Subsequently, the worst elephant is replaced by a separating operator.

The herding behaviour of elephants can solve global optimization problem according to the following rules:

(1) The elephants' population is composed of some clans, each clan has a fixed number of elephants.
(2) A fixed number of male elephants will leave their family group and live solitarily far away from the main elephants' group at each generation (separating operator).
(3) The elephants in each clan live together under the leadership of a matriarch.

The notations used are defined as follows:

$C_i$: the $i$th clan.
$X_{i,j}$: the position for elephant $j$ in clan $C_i$.
$X_{i,j,d}$: the $d$th element of the position $X_{i,j}$.
$X_{\text{new,i,j}}$: the newly updated position for elephant $j$ in clan $C_i$.
$X_{\text{best,i}}$: the best elephant individual according to the fitness value in clan $C_i$.
$X_{\text{center,i}}$: the centre of clan $C_i$.
$X_{\text{center,i,d}}$: the $d$th element of the centre of clan $C_i$.
$X_{\text{worst,i}}$: the worst elephant individual according to the fitness value in clan $C_i$.
$X_{\max}$: the upper bound of the position of elephant individual.
$X_{\min}$: the lower bound of the position of elephant individual.
rand, $\alpha$, $\beta$ and $r$ : random numbers between 0 and 1.
$nClan$: the number of clan in the population.
$N_i$: the number of elephants in clan $C_i$.
The different steps of EHO algorithm are described as follows:

**Step 1.** Initialization. Set generation counter $t = 1$; initialize the population; the maximum generation $MaxGen$.
**Step 2.** Sort all the elephants according to their fitness.
**Step 3.** Divide the whole of elephants' population into some clan according to their fitness.
**Step 4.** Implement a clan updating operator using equation (3.8) for all elephants in each clan, equations (3.10) and (3.9) for the matriarch:

$$X_{\text{new},i,j} = X_{i,j} + \alpha * (X_{\text{best},i} - X_{i,j}) * r \tag{3.8}$$

$$X_{\text{new},i,j} = \beta * X_{\text{center},i} \tag{3.9}$$

$$X_{\text{center},i,d} = \frac{1}{N_i} * \sum_{i=1}^{N_i} X_{i,j,d} \tag{3.10}$$

**Step 5.** Implement a separating operator to the worst elephant in each clan using equation (3.11):

$$X_{\text{worst},i} = X_{\min} + (X_{\max} - X_{\min} + 1) * \text{rand} \tag{3.11}$$

**Step 6.** Evaluate the population by the newly updated positions, $t = t + 1$.
**Step 7.** Combine all clans into one population.
**Step 8.** If $t = MaxGen$ return $X_{1,1}$ the best solution, else go to Step 2.

## 4. Approach description

### 4.1. Discrete EHO (D-EHO)

The QoS-WSC problem addressed in this paper is a discrete problem in nature. So, in this section, we describe the proposed discrete EHO algorithm called D-EHO to solve discrete optimization problem. Despite that the original EHO is designed to solve continues optimization problem, we adapt this approach to discrete

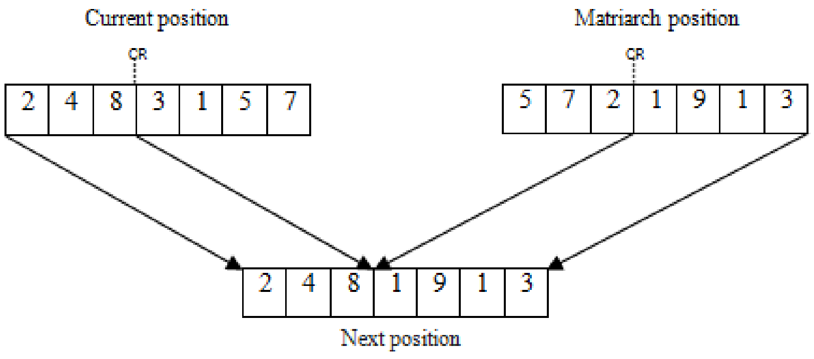

FIGURE 7. Crossover operator.

problems. To achieve this adaptation, we propose to encode a solution using each elephant's position in the herd (population individuals) as a candidate solution for the QoS-WSC problem.

For the composition example of Figure 5, a candidate solution is given as:

| | | | | |
|---|---|---|---|---|
| $S_{13}$ | $S_{22}$ | $S_{43}$ | $S_{51}$ | $S_{62}$ |

Where the solution is encoded as a vector, the $i$th element represents the index of the concrete service selected for the $i$th abstract service:

| | | | | |
|---|---|---|---|---|
| 3 | 2 | 3 | 1 | 2 |

For the discritization of the algorithm, we define a new clan updating operator (which replaces the Eq. (3.8)) using a crossover operation illustrated in Figure 7, one crossover point $CR$ is applied between the current position of the elephant and the position of matriarch in the corresponding clan.

The next position = Crossover(Current position, CR, Matriarch position);

while $CR \in [1, |X_{i,j}|]$

The new code of clan updating operator is given as follows:

```
(1)  for C_i=1 to nClan (for all clans in population) do
(2)    for j=1 to N_i (for all elephants in clan C_i) do
(3)      $CR = \text{rand}(1, |X_{i,j}|)$;
(4)      $X_{\text{new},i,j} = Crossover(X_{i,j}, CR, X_{\text{best},i})$;
(5)      if $X_{i,j} = X_{\text{best},i}$ then
(6)        for $d = 1$ to $|X_{i,j}|$
(7)        $X_{\text{new},i,d} = \frac{1}{N_i} * \sum_{i=1}^{N_i} X_{i,j,d}$
(8)        End for d;
(9)      $X_{\text{new},i,j} = floor(B * X_{\text{center},i})$
(10)       end if;
(11)   end for j;
(12) end for C_i.
```

## 4.2. Multi Objective Discrete EHO for QoS-WSC

In order to adapt a multi-objective version of D-EHO to the QoS-WSC problem, we propose a Pareto-based approach with an external archive (EA) to store the Pareto front at each iteration and we adopt the algorithm structure proposed in [24].

The algorithm takes as input the workflow of the composite service as well as the set of available web services implemented by a multidimensional matrix containing the different values for each QoS attribute. As output the algorithm returns a composition set (solutions) contained in the External Archive EA. This mechanism protects good solutions from random effects. During the evolutionary process, dominated and duplicated solutions are removed from the archive.

Each elephant in the population (a composition solution) has a fitness value. The formula used in this paper is inspired by $SPEA2$ sort method [15], each individual from the population has a strength value equal to the number of dominated individuals in the population. The fitness of an individual $i$ in the population is computed as the sum of strengths of each individual dominating $i$ (dominator strengths). The goal is to minimize the fitness function(non-dominated solutions will have zero fitness). In order to discriminate elephants having the same fitness, additional density information is added, the $k$th nearest neighbor clustering method is used. The density function is the inverse distance to the $k$th nearest neighbor.

The proposed algorithm is given as follows:

```
(1) Begin;
(2) Initialization. Initialize the population P; the maximum generation MaxGen;
(3) Sort all the elephants in the population according to their fitness;
(4) Add the non dominated solutions found in P into EA;
(5) For it=1 to MaxGen
    (a) Divide the whole elephants' population into nClan clan according to their fitness;
    (b) Implement clan updating operator Section 3.1;
    (c) Implement separating operator Eg. (11);
    (d) Evaluate population by the newly updated positions;
    (e) Combine all clans into one population;
    (f) Sort all elephants in the population according to their fitness;
    (g) Update the External Archive EA:
        (i) Add all new non-dominated solutions into EA;
        (ii) Remove all solutions dominated in EA;
        (iii) If the archive is full then randomly select solutions to be deleted from EA;
(6) end;
(7) Return EA as the Pareto front;
(8) End.
```

First, an initial population of solutions is generated randomly, $P$ elephants. Elephants are sorted according to their fitness value. The algorithm maintains an external archive $EA$ to store non-dominated solutions based on the Pareto dominance. Then, the entire population is divided into $nClan$ sub-populations (clan) such that each clan $C_i$ containing $N_i$ elephants. Second, within each clan $C_i$, an evolution process is applied to improve all elephants in $C_i$ with clan updating operator. Then, a separating operator is implemented to replace only the worst elephant in each clan $C_i$.

After the local searching for all clans, all elephants in all clans are combined (mixed) and reordered. Then the external archive is updated, dominated and duplicated solutions are removed from the archive and if the archive is full, some solutions are removed according to a random selection. Once the termination condition is reached, the external archive containing the Pareto front is returned as the result.

## 5. Experimentation

In this section, we report the results of simulation experiments used to study the performance of the proposed algorithm (MO-D-EHO) in comparison with SPEA2 and MOPSO algorithms applied for QoS-WSC. The algorithms are implemented with Matlab R2012 on a HP dc7900 machine with 2 Intel Duo 2.4 GHZ processors and 2 GB RAM.

Table 2. Algorithms control parameters.

| Parameter | Value |
|---|---|
| Population size | 100 Individuals |
| Archive size | 100 Individuals |
| Maximum generation | 500 Generations |
| Number of clan | 5 Clans |
| Number of abstract services | 10, 30, 50 Services |
| Number of concrete services | 50, 100, 500, 1000 Services |
| Number of evaluation per scenario | 50 Times |

A brief description of the algorithms used for comparison is given in the following:

$MOPSO$: Multi-Objective Particle Swarm Optimization ($MOPSO$) is proposed by Coello Coello *et al.*, in 2004 [3]. It is a multi-objective version of $PSO$ which incorporates the Pareto Envelope to handle the multi-objective optimization problems. Like $PSO$, particle in $MOPSO$ are sharing information and moving towards global best particles and their own personal (local) best memory. However, unlike $PSO$, there is more than one criterion to determine and define the best (global or local). All of non-dominated particles in the swarm, are gathered into a sub-swarm called Repository, and every particle chooses its global best target, among members of this Repository. For personal (local) best particle, a domination based and probabilistic rules is used.

$SPEA2$: Strength Pareto Evolutionary Algorithm 2 ($SPEA2$) [29] is an extended version of $SPEA$ multi-objective evolutionary optimization algorithm. SPEA2 uses an archive with a predefined size for storing all the non-dominated solutions found at each generation. The external population is mixed with the current population. Each individual from the mixed population has a strength value equal to the number of dominated individuals in the mixed population. The fitness of an individual $i$ from the mixed population is computed as the sum of strengths of each individual dominating $i$ (dominator strengths). The goal is to minimize the fitness (non-dominated solutions will have zero fitness). In order to discriminate individuals having the same fitness, an additional density information is added. In the case of $SPEA2$, $k$th nearest neighbor clustering method is used. The density function is the inverse distance to the $k$th nearest neighbor. Binary tournament selection, crossover and mutation are used to produce offspring. The archive is updated at each generation.

## 5.1. Dataset and test case generation

The analysis is performed on several test cases. A test case is defined by an abstract workflow along with a certain number of alternative concrete services defined for each abstract service.

In order to ensure a fair comparison, we use the following values of the algorithm control parameters: An initial population of 100 individuals is generated randomly, then, the entire population is divided into 5 sub-populations (clan) such that each clan containing 20 individuals and we run each algorithm for 500 generations. An archive of 100 individuals is considered. For each test case with a different number of abstract services or concrete services, each algorithm is executed 50 times to evaluate the algorithms' performance. The different algorithms control parameters are shown in Table 2.

The test cases are defined by varying the number of abstract services in the workflow from 10 to 50. The number of concrete services varies from 50 to 1000 and three QoS attributes are considered. The QoS attributes of these concrete services are collected from real web services. A public database [1] containing 2507 records characterizing real web services is used as data source (random services are selected from this database). For the QoS, a problem with 3 objectives is considered as the maximization of reliability($RL$) and availability($AV$) and the minimization of response time($RT$).

If $CS$ a solution of the problem, the objectives are formulated as:

$$\mathrm{Min}([RT(CS), 1 - AV(CS), 1 - RL(CS)]) \tag{5.1}$$

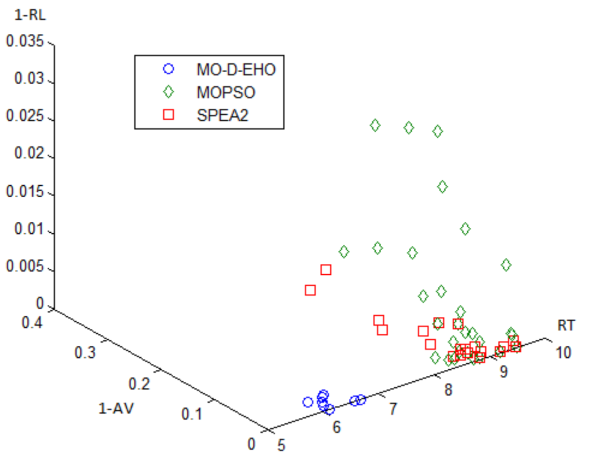

FIGURE 8. Graphical representation of solutions.

Where, the response time is the time taken to send a request and receive a response, the availability represents the number of successful invocations/total invocations and the reliability is defined as the probability of success (reliability = 1-probability of failure).

## 5.2. Metrics and result analysis

This section presents a set of comparative experiments between the proposed algorithm MO-D-EHO, SPEA2 [29] and MOPSO [3]. Several metrics are proposed in literature [27] for evaluating and comparing MOP, most of them suppose having the true Pareto front (Pf), while this front requires exact search which is impossible to get for large data set.

In this paper, first, we use the graphical representation of Pf given in Figure 8 (identified solutions), but a simple visual inspection of the Pareto front is not sufficient to decide which algorithm is better. In order to accurately compare the algorithms, an objective evaluation is required and the *Set Coverage* performance metric presented in Figure 9 is used to compare algorithms performances in term of dominance. Furthermore, we use *hypervolume* and *spacing* indicators which are computed in Figures 10 and 11 over the returned Pareto fronts, taking average values, to compare the quality of the set of solutions returned by each algorithm. Both indicators vary in the range [0,1], the Hypervolume should be maximized while the Spacing should be minimized.

### *5.2.1. Graphical representation of the identified solutions*

With a test case implementing 10 abstract services and 100 concrete services per abstract service, the Figure 8 presents the identified solutions for the three objectives.

This figure indicate that MO-D-EHO yields numerous good solutions with low value for all objectives. SPEA2 yield numerous good balanced solutions but some of them are dominated by some MO-D-EHO solutions, while MOPSO yield solutions that are, in general, dominated by other solutions.

### *5.2.2. Set Coverage-based comparisons*

Set Coverage (C) [2] is useful for comparing two solutions sets found by two different multi-objective algorithms. If X and Y are two set representing Pareto front approximations, the Set Coverage C(X, Y) is used to measure the percentage of solutions in Y dominated by at least one solution in X in the objective space.

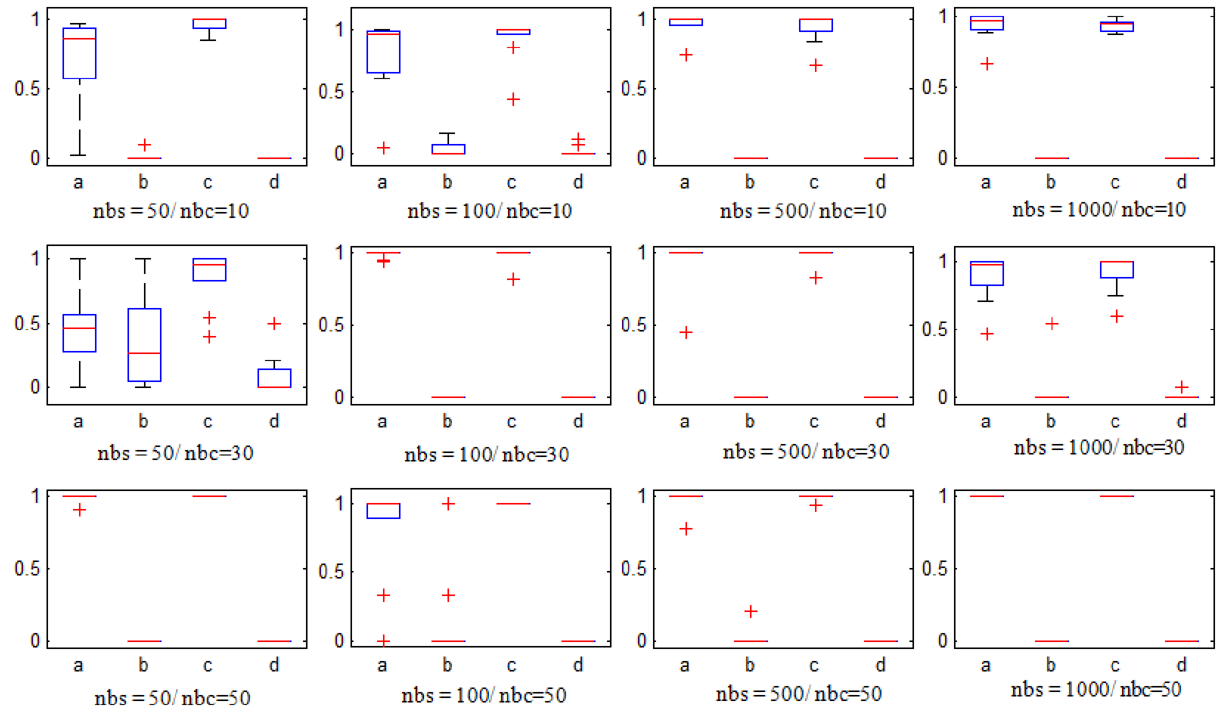

FIGURE 9. Set Coverage box-plots. Each rectangular sub-figure describes a set of Set Coverage values marked from (a) to (d) having the following correspondence: a.(MO-D-EHO, SPEA2), b.(SPEA2, MO-D-EHO), c.(MO-D-EHO, MOPSO), d.(MOPSO, MO-D-EHO).

The Set Coverage is given by the formula:

$$C(X,Y) = \frac{|\{u \in Y | \exists v \in X : v < u\}|}{|Y|} \tag{5.2}$$

If $C(X,Y) = 1$, then all solutions in Y are dominated by some solutions in X. If $C(X,Y) = 0$, then no solution in Y is dominated by a solution in X. When comparing two algorithms, the best algorithm has a higher value for $C(X,Y)$ and a lower value for $C(Y,X)$.

Figure 9 illustrates the Set Coverage box-plots. High average values of C(X, Y) indicate better performances for the algorithm X.

The results depicted in Figure 9 indicate that for all test cases, the median value of the box (a) and (c) which represent the set coverage C(MO-D-EHO, SPEA2) and C(MO-D-EHO, MOPSO) respectively, is always higher than that of the box (b) and (d) which represent the set coverage C(SPEA2, MO-D-EHO) and C (MOPSO, MO-D-EHO) respectively, which means that better performances are given by the MO-D-EHO algorithm compared with SPEA2 and MOPSO algorithms. In the case where (nbs=100,nbc=10) we have the median value for the box (a) and (c) equal to 1 where it is 0 for the box (b) and (d). So, the solutions provided by MO-D-EHO algorithm dominates the ones found by SPEA2 and MOPSO, and in this case, the MO-D-EHO algorithm offers better result in term of set coverage metric.

### *5.2.3. Hypervolume-based comparisons*

The hypervolume (HV)[30,31] metric is one of the most frequently used measures to evaluate an approximate front. For a multi-objective problem the hypervolume metric measures the volume bounded by the approximate

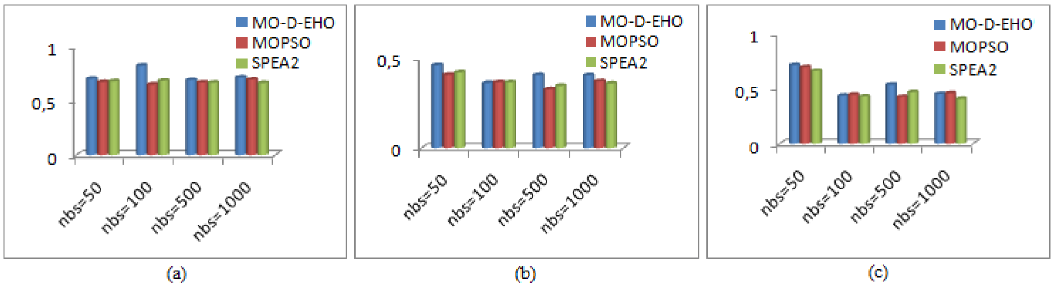

FIGURE 10. Hypervolume histogram.

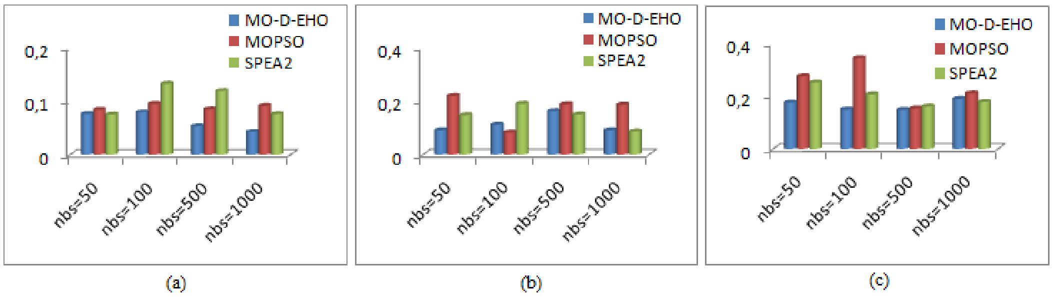

FIGURE 11. Spacing histogram.

front and a reference point(the worst point in our case). A higher value for hypervolume means that a greater portion of the Pareto front is covered. Therefore, an approximate front with a higher HV value is preferred [11,17].

Figure 10 illustrates the hypervolume histogram for the test cases with 50 to 1000 concrete services (nbs) and 10(a), 30(b) and 50(c) abstract services(nbc).

Experiment results illustrated in Figure 10 indicate that for the 12 test cases, we notice that except for 3 cases (nbc=30, nbs=100), (nbc=50, nbs = 100) and (nbc = 50, nbs = 1000) where the three algorithms yield a similar HV values; the MO-D-EHO algorithm yield high HV values for the majority of test cases which means that a greater portion of the Pareto front is covered by the solutions returned by the proposed algorithm.

### *5.2.4. Spacing-based comparisons*

Another relevant indicator is Spacing (Sp) [10], which computes the diversity of the solutions composing the Pareto front (Pf). The Sp metric is designed as [11,17]:

$Sp(Pf) = \sqrt{\sum_{i=0}^{N} (d_i - \bar{d})/(N-1)}$, where $N$ is the number of solutions in Pf, $d_i$ the minimal distance of the $i$th solution and all elements in Pf, and $d$ the average value of all $d_i$. The Spacing is designed to measure how evenly the members of an approximation set are distributed. A value of 0 for Sp means that all members of the approximation set are equidistantly spaced.

The results illustrated in Figure 11 present the Spacing values for test cases with 50 to 1000 concrete services (nbs) and 10 (a), 30 (b) and 50 abstract services (c). Lower values indicate a more efficient algorithm. These results show that, in most cases the MO-D-EHO algorithm offers lower values of Sp than the others considered algorithms, which means that MO-D-EHO is able to generate a greater variety of solutions than that provided by the two others algorithms.

Across the experimentations illustrated in this section, we notice that MO-D-EHO algorithm provides significant improvements than the compared algorithms in terms of Spacing, Hypervolume and Set Coverage metrics.

So, the proposed MO-D-EHO algorithm is able to produce a set of good Pareto optimal solutions and is the most appropriate for solving the QoS-WSC optimization problem.

## 6. Conclusion

This paper presents a discrete and multi-objective elephants herding optimization algorithm. The original EHO is modified by introducing a new updating operator which gives a discrete version of EHO. Then a multi-objective version is developed using a Pareto approach and a sort method inspired by SPEA2 algorithm for addressing multi-objective QoS aware web service composition. Compared with SPEA2 and MOPSO, MO-D-EHO; for a large dataset; is shown to be an efficient algorithm for solving the multi-objective QoS-WSC problem. The strength of the proposed algorithm is provided by the process of dividing and combining the population into clans which allows escaping to the local optimum. From experimental results, we show that our solution offers better performances in terms of Set Coverage, Hypervolume and Spacing metrics and significantly outperform the existing algorithms in terms of dominance and diversity of solutions. In the future, our research will focus on parallelization of the proposed algorithm. In addition, a decision method will be proposed to select a solution that best suits the needs, because, when using a Pareto-based approach, a set of non-dominated solutions is obtained.

## References

[1] E. Al-Masri and Q.H. Mahmoud, QoS-based discovery and ranking of web services. In *Proceedings of 16th International Conference on Computer Communications and Networks, ICCCN* (2007) 529–534.

[2] C. Artemio Coello Coello, L. Gary B. and V. Veldhuizen David, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer; 2nd Edition (2007).

[3] C. Artemio Coello Coello and M.S. Lechuga, MOPSO: a proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02* (2002) 1051–1056.

[4] G. Canfora, M.Di Penta, R. Espositio and M. Luisa Villani, An approach for QoS-aware service composition based on genetic algorithms. In *GECCO '05 Proceedings of the 2002 Congress on Evolutionary Computation* (2005) 1069–1075.

[5] W.-Ch. Chang, Ch.-Seh Wu and Ch. Chang, Optimizing dynamic web service component composition by using evolutionary algorithms. In *IEEE International Conference on Web Intelligence* (2005) 708–711.

[6] M. Cremene, M. Suciu, D. Pallez and D. Dumitrescu, Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition. *Inter. J. Appl. Soft Comput.* **39** (2016) 124–139.

[7] S.R. Dhore and M.U. Kharat. QoS based web services composition using ant colony optimization: mobile agent approach. *Inter. J. Adv. Res. Comput. Commun. Eng.* **1** (2012) 519–527.

[8] G. Jayjit J. and G. Piyush, A novel web service composition using ant colony optimization with agent based approach. *Inter. J. Emerging Technologies and Innovative Res.* **2** (2015) 1685–1688.

[9] M.C. Jaeger and G. Müehl, QoS-based selection of services: The implementation of a genetic algorithm. In *Commun. Distributed Syst. (KiVS), ITG-GI Confer.* (2007) 1–12.

[10] Sch.R. Jason, Fault tolerant design using single and multicriteria genetic algorithm optimization. In Technical report, DTIC Document (1995).

[11] S. Jiang, Y.-S. Ong, J. Zhang and L. Feng, Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Trans. Cybernetics* **44** (2014) 2391–2404.

[12] H. Kadima and V. Monfort, Les Web services Techniques, dmarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML. Edition Dunod (2003).

[13] S. Kalepu, Sh. Krishnaswamy and S. Wai Loke, Verity: A QoS metric for selecting web services and providers. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW03)* (2004) 131–139.

[14] A.L. Lemos, F. Daniel and B. Benatallah, Web service composition: A survey of techniques and tools. *ACM Computing Surveys* **48** (2015) 33, 41.

[15] L. Li, P. Cheng, L. Ou and Z. Zhang, Applying multi-objective evolutionary algorithms to QoS-aware web service composition. In *6th International Conference on Advanced Data Mining and Applications* (2010) 270–281.

[16] J. Liao, Y. Liu, X. Zhu, J. Wang and Q. Qi. A multi-objective service selection algorithm for service composition. In *19th Asia-Pacific Conference on Communications (APCC), Bali Indonesia* (2013) 75–80.

[17] N. Riquelme, Ch. Von Lcken and B. Baran, Performance metrics in multi-objective optimization. In *IEEE Latin American Computing Conference (CLEI)* (2015) 1–11.

[18] M.R. Timothy and J.S. Arora, Survey of multi-objective optimization methods for engineering. *Inter. J. Structural Multidisciplinary Optimiz.* **26** (2004) 369–395.

[19] G.-G. Wang, S. Deb and L. dos Santos Coelho, Elephant herding optimization. In *3rd International Symposium on Computational and Business Intelligence* (2015) 1–5.

[20] G.-G. Wang, S. Deb, Xiao-Zhi Gao and D. Santos Coelho Leandro, A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. *Inter. J. Bio-Inspired Comput.* **8** (2016) 394–409.

[21] W. Wang, Q. Sun, X. Zhao and F. Yang, An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication. *Inter. J. Comput. Intell. Syst.* **3** (2010) 18–30.

[22] Q. Wu and Q. Zhu, Transactional and QoS-aware dynamic service composition based on ant colony optimization. *Future Generation Comput. Syst.* **29** (2013) 1112–1119.

[23] Y. Yao and H. Chen, QoS-aware service composition using NSGA-II$_1$. In *ICIS '09 Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (2009) 358–363.

[24] S. Yassa, R. Chelouah, H. Kadima and B. Granado, Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *The Scientific World Journal* **2013** (2013) 350934.

[25] Sh. Yulu and Ch. Xi, A survey on QoS-aware web service composition. In *Third Inter. Confer. Multimedia Information Networking and Security (MINES)* (2011) 283–287.

[26] L. Zeng, Boualem B., A.H.H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, QoS-aware middleware for web services composition. *IEEE Trans. Software Eng.* **3** (2004) 311–327.

[27] E. Zitzler, K. Deb and L. Thiele, Comparison of multi-objective evolutionary algorithms: Empirical results. *J. Evolutionary Comput.* **8** (2000) 173–195.

[28] Zh.-Zhong Liu, X. Xue, J. quan Shen and W.-R. Li, Web service dynamic composition based on decomposition of global QoS constraints. *Inter. J. Adv. Manufacturing Technology* **69** (2013) 2247–2260.

[29] E. Zitzler, M. Laumanns and L. Thiele, SPEA2: improving the strength pareto evolutionary algorithm for multi-objective optimization. In Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, edited by K.C. Giannakoglou, D.T. Tsahalis,J. Priaux, K.D. Papailiou, T. Fogarty. International Center for Numerical Methods in Engineering (2001) 95–100.

[30] E. Zitzler and L. Thiele, Multi-objective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Trans. Evolutionary Comput.* **3** (1999) 257–271.

[31] E. Zitzler and L. Thiele, Multi-objective optimization using evolutionary algorithms- A comparative case study. In *Parallel Problem Solving from Nature.* Edited by A.E. Eiben, T. Bäck, M. Schoenauer, H.P. Schwefel. In Vol. 1798 of Lecture Notes in Computer Science. Springer, Berlin (1998).