A Multi-Objective Approach for QoS-aware Service Composition

Marcel Cremene†, Mihai Suciu‡, Florin-Claudiu Pop*, Denis Pallez‡ and D. Dumitrescu†

*Technical University of Cluj-Napoca, Romania
†Babes-Bolyai University of Cluj-Napoca, Romania
‡Université de Nice Sophia-Antipolis, France

Abstract—Service composition is one of the main research areas in the Software Oriented Computing (SOC) domain. A certain functionality may be offered by different services having different Quality of Service (QoS) attributes. QoS optimization problem is multi-objective by its nature. Some multi-objective optimization algorithms have been used for solving this problem. Their advantages, compared to single-criteria algorithms, are: (i) the aggregation of criterion functions is avoided and (ii) the user has the possibility to select a posteriori one of the Pareto optimal solutions. There is a small number a such proposals and very few comparative tests. Comparative tests with the most known multi-criteria optimization algorithms are presented. Differential evolution based multi-objective algorithms seems to be a good trade-off between performance and complexity. The converge time of multi-criteria optimization algorithms is comparable to the single-criteria algorithms.

Index Terms—Service Oriented Computing, Multi-criteria Optimization, Differential Evolution, Quality of Service

I. INTRODUCTION

Service Oriented Architecture (SOA) implementations are more and more popular, diverse and widespread in enterprise distributed environments. This fact is due to their technical advantages over more traditional methods of distributing computing. These advantages include: delivering application functionality as services across several platforms, providing location independence, authentication and authorization support and dynamic search and connectivity to other services.

The QoS optimization problem is multi-objective by its nature. The user might prefer to see several good solutions (Pareto optimal) and decide which is the best for himself/herself while the standard optimization algorithms offer only one solution. It is more natural to let the user decide if he/she wants to pay a specific known price than to ask him/her to specify a priori how important is the price for him/her and without knowing the precise price value. By using multi-criteria optimization, we also avoid defining an aggregation function, which is not a trivial task.

II. PROBLEM STATEMENT

A composite service can be described as a process that involves the execution of several activities according to a workflow. An example workflow for a flight booking process is depicted in Fig. 1.

The QoS of the composite service is obtained by aggregating the QoS of the component services. Examples of aggregation rules are given in [1], [2] and [3].

Given m abstract services and n concrete services for each abstract service, there are \( n^m \) possibles combinations. Finding the QoS optimal solution is an non-deterministic polynomial-time hard (NP-hard) problem [1].

III. RELATED WORK

The problem stated previously is well known in domains like Service Oriented Computing (SOC) and Search-based Software Engineering (SBSE) [4]. We found it discussed in [1], [5], [6], [7]. G. Canfora et al. [1] compare a linear integer programming [8] based algorithm with a genetic algorithm. As a case study, they considered a workflow containing 8 distinct abstract services.

Their conclusion was that GA is able to deal with QoS attributes having non-linear aggregation functions. Also, GA can scale-up when the number of concrete services per abstract service increases.

Multi-objective algorithms applied to service composition. The idea of using multi-criteria optimization algorithms in the service composition domain is not new.

Taboada et al. [9] propose a custom genetic algorithm called Multi-objective Multi-State Genetic Algorithm (MOMS-GA) for general system design optimization. The system components have different performance levels, cost, weight and reliability. A multi-state approach means that the system and its components may have more than two states: completely working, partially working, partially failed and completely failed.

Yao et al. [3] propose an approach based on Non-dominated Sorting Genetic Algorithm-II (NSGA2) [10] algorithm. A solution is encoded using an integer vector, similar to [1].

Li Li et al. [2] propose another multi-criteria oriented approach based on the Strength Pareto Evolutionary Algorithm (SPEA2) [11] algorithm. Three criteria are considered: response time, cost and availability. The same type of genome encoding, integer vector based, is used. The algorithm parameters are very similar to the previous example: mutation rate 0.01, population size of 100 individuals, single-point crossover with a 0.95 rate and binary tournament selection. Population size is 100 and the archive population (specific to this algorithm) size is 200 individuals.

A scenario with 4 abstract services and from 90 to 500 concrete services per abstract service was considered. The tests
have shown that SPEA2 converges to the Pareto-optimal front in less than 100 generations.

**Remarks about the related work.** It seems that evolutionary methods are preferred for single and multi-objective approaches. Despite the fact that the QoS optimization problem is multi-objective by nature, these algorithms are less used than single-objective ones. We did not found a comparison between different multi-objective algorithms and an analysis about their complexity.

The scenarios presented in the selected literature are using a small number of abstract service (less than 5) and some of them also a small number of concrete services (less than 9). Thus, it is not obvious how scalable are these algorithms.

These facts motivated us to perform a comparative analysis using several different multi-objective algorithms and test more complex scenarios. The next section introduces the theoretical support for multi-objective optimization.

### IV. Multi-objective Optimization Algorithms

Let us consider $n$ objectives defined by the set $\{f_i\}_{i \in \{1,...,m\}}$ of real valued functions $f_i: X \to \mathbb{R}$, $X \subseteq \mathbb{R}^n$. $F: X \to \mathbb{R}^m$ is the vector valued function $F(x) = (f_1(x), ..., f_m(x))$.

A decision vector $x \in X$ is said to Pareto-dominate $y \in X$, denoted as $x \prec y$, if and only if

$$\forall i \in 1, ..., m, f_i(x) \leq f_i(y),$$

and

$$\exists j \in 1, ..., n \text{ such that } f_j(x) < f_j(y).$$

A solution $x \in X$ is Pareto-optimal if and only if $\not\exists y \in X$ such that $y \prec x$.

The **Pareto-Optimal Set** (POS) is defined as the set of all Pareto-optimal solutions

$$\text{POS} = \{x \in X | \not\exists y \in X, y \prec x\}.$$  

The **Pareto-Optimal Front** (POF) is defined as the set of all objective functions values corresponding to the solution in POS.

$$\text{POF} = \{F(x)| x \text{ is non-dominated}\}.$$  

**NSGA.** Non-dominated Sorting Genetic Algorithms 2 (NSGA2) as a multi-objective optimization algorithm has been proposed by [12].

NSGA2 is an elitist algorithm, based on a $\mu, \lambda$ selection ($\mu$ - parents $+ \lambda$ - childs), favouring good solutions through a ranking and sorting system based on Pareto-dominance.

**SPEA2.** Strength Pareto Evolutionary Algorithm 2 (SPEA2) [11] is an elitist multi-objective optimization algorithm. This algorithm uses an archive for storing the best solutions found at each generation.

**GDE3.** A multi-objective version GDE3 [13] has been proposed by Kukkonen and Lampinen in 2005. GDE3 modifies the selection operator in DE by introducing the concept of dominance. The trial vector is selected to replace the decision vector if Pareto dominates $x$. Condition $j = \text{rand } j$ indicates that at least one component of mutated vector $v_i$ is selected. If the vectors are indifferent to each other (neither is better) then both vectors are kept.

**POSDE.** POSDE, one of the first multiobjective DE approaches was introduced by [14]. POSDE uses a secondary population to retain the non-dominated solutions found at each generation in the evolutionary process. Diversity is achieved by a distance metric used to alter the fitness of each individual when it is compared with the elements of the archive.

**ε-MyDE.** ε-MyDE is a multi-objective algorithm based on Differential Evolution introduced in [15]. A secondary population is adopted in order to retain the non-dominated solutions found during the search process. The concept of ε-dominance [16] is used to assure a good distribution of solutions in the secondary population.

### V. Numerical Experiments

In this section we describe a set of comparative tests using the previously described algorithms applied to the service QoS optimization problem.

Each concrete service is defined by the QoS properties vector $Q_{i,j} = (t, r, c)$, $t$-response time, $r$-rating, $c$-cost. These properties correspond to the optimization objectives. The MOP goal is to find the best combination of concrete services that maximizes the overall rating and minimizes the overall cost and time.
The genome used to encode the solution is an integer vector for NSGA2 and SPEA2. Each gene is an integer value representing the index of the concrete service used.

DE algorithms use floating point encoding to resolve problems in a continuous domain so we need to use a discrete version. There are several approaches for solving a finite domain problem using DE. Our approach is based on [17] and uses a mapping between the discrete and continuous domain. The DE genome is a real value vector.

A. Experiments with two criteria

A first set of experiments was performed using a workflow with a pure sequential architecture and three algorithms: NSGA2, SPEA2 and GDE3. For these experiments we considered only two criteria: rating and time + cost. The results for 10 abstract services and 20 concrete services are depicted in Fig. 2.

B. Experiments with three criteria

A second series of experiments was performed using more complex workflows (including sequences, split, join, loops and switch blocks) and with three independent criteria: response time, rating and cost.

For all the algorithms the populations size was limited to 150 individuals, evolved for 300 generations, for DE CR=0.4 and F=0.3 was used. For HV a reference point (100,100,100) was chosen. For ϵ-MyDE a selection pressure of 0.8 was chosen. The size of the archive for POSDE, SPEA2 is 150 in order to assure a fair comparison.

In a case with with $m=10$ abstract services and $n=10$ concrete services per abstract service (low complexity) all algorithms behave similarly. A more complex scenario involving $m=20$ abstract services, each of them having $n=40$ concrete services was evaluated. The added complexity makes it harder to find a good PS. SPEA2 doesn’t find a good set while the DE approaches and NSGA seems to converge to the same PF.

In a case with $m=40$ abstract services and $n=40$ concrete services per abstract service. Only NSGA2 and GDE3 are able to find a Pareto front and assure good diversity of solutions. The other algorithms produce a PS in which most solutions are dominated.

C. Performance evaluation metrics

The simple visual inspection of the solution front, as depicted in Fig. 2, is not clear enough for deciding which algorithm is the best. A more objective evaluation is necessary.

According to [18] when comparing different algorithms we cannot rely only on one indicator. The metrics [19] used for comparison are: hypervolume (HV), spread (S) and set coverage (C).

Tables I illustrate the mean ($\mu$) and standard deviation($\sigma$) of HV and S metrics. A higher value for each metric is desirable. In all scenarios NSGA2 and DE approaches obtain a better metric value assuring a good spread of solutions.

If $X$ and $Y$ are two approximations of the PF, the set coverage $C(X,Y)$ is used to measures the percentage of solutions in $Y$ dominated by at least one solution in $X$.

Table II presents the Set Coverage indicators for NSGA2, SPEA2, GDE3 compared with all other algorithms tested.

In order to have an idea about the algorithms speed we measured the time required to find the solution set. Figure 3 presents the time required by all algorithms to determine a PS after 300 generations for all combinations of $m \in \{10,20,30,40,50\}$ abstract and $n \in \{10,20,30,40,50\}$ concrete services per abstract service. As expected, the execution time increases with the number of abstract services. It may be observed that all DE approaches require less time while NSGA2 is the slowest algorithm.

VI. CONCLUSIONS

Several multi-objective optimization algorithms are tested for solving the services QoS optimization problem. The tests indicate that NSGSA2 and DE-based algorithms are the best in terms of solutions diversity and Pareto optimality. NSGA2
seems to have the best performances demonstrated by the evaluation metrics. But NSGA2 proven to be also the slower algorithm. Therefore, DE approaches seem to be a better choice for addressing complex problems.

The time tests demonstrate that multi-criteria algorithms are able to solve the QoS optimization problems in a reasonable time: few seconds for very complex scenarios, on a standard PC machine. According to our tests, the time required by a multi-objective algorithm is comparable to that required by solving a single-objective problem.

ACKNOWLEDGMENT

This work was supported by CNCSIS-UEFISCSU, national project number PN II-TE code 252, contract 33/2010. It was also partially supported by a grant from the John Templeton Foundation (the opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation).

REFERENCES


### TABLE I

<table>
<thead>
<tr>
<th>m/n</th>
<th>HV</th>
<th>S</th>
<th>HV</th>
<th>S</th>
<th>HV</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10</td>
<td>3.2E+05</td>
<td>4.5E+02</td>
<td>3.5E+05</td>
<td>1.3E+04</td>
<td>4.8E+01</td>
<td>0.11</td>
</tr>
<tr>
<td>10/20</td>
<td>3.4E+05</td>
<td>4.5E+02</td>
<td>4.2E+05</td>
<td>1.5E+04</td>
<td>5.2E+01</td>
<td>0.10</td>
</tr>
<tr>
<td>10/40</td>
<td>4.6E+05</td>
<td>8.5E+02</td>
<td>4.1E+05</td>
<td>2.4E+04</td>
<td>4.9E+01</td>
<td>0.09</td>
</tr>
<tr>
<td>20/10</td>
<td>3.4E+05</td>
<td>2.1E+03</td>
<td>2.3E+05</td>
<td>1.1E+11</td>
<td>8.4E+01</td>
<td>0.08</td>
</tr>
<tr>
<td>20/20</td>
<td>2.7E+05</td>
<td>5.7E+03</td>
<td>1.3E+05</td>
<td>6.2E+12</td>
<td>5.3E+01</td>
<td>0.09</td>
</tr>
<tr>
<td>20/40</td>
<td>3.7E+05</td>
<td>9.3E+03</td>
<td>3.3E+05</td>
<td>3.0E+04</td>
<td>3.0E+01</td>
<td>0.08</td>
</tr>
<tr>
<td>30/10</td>
<td>4.0E+05</td>
<td>2.6E+03</td>
<td>3.9E+05</td>
<td>1.6E+04</td>
<td>4.8E+01</td>
<td>0.08</td>
</tr>
<tr>
<td>30/20</td>
<td>3.7E+05</td>
<td>5.5E+03</td>
<td>2.9E+05</td>
<td>9.3E+03</td>
<td>3.9E+01</td>
<td>0.08</td>
</tr>
<tr>
<td>30/40</td>
<td>4.1E+05</td>
<td>5.7E+03</td>
<td>2.6E+05</td>
<td>1.5E+04</td>
<td>5.2E+01</td>
<td>0.08</td>
</tr>
</tbody>
</table>

### TABLE II

<table>
<thead>
<tr>
<th>m/n</th>
<th>C(a,b)</th>
<th>C(a,ε)</th>
<th>C(a,d)</th>
<th>C(a,ε)</th>
<th>C(b,a)</th>
<th>C(b,ε)</th>
<th>C(b,d)</th>
<th>C(b,ε)</th>
<th>C(c,a)</th>
<th>C(c,ε)</th>
<th>C(c,d)</th>
<th>C(c,ε)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10</td>
<td>1.00</td>
<td>0.58</td>
<td>0.51</td>
<td>0.26</td>
<td>0.40</td>
<td>0.21</td>
<td>0.04</td>
<td>0.08</td>
<td>0.77</td>
<td>0.00</td>
<td>0.60</td>
<td>0.32</td>
</tr>
<tr>
<td>10/20</td>
<td>1.03</td>
<td>0.49</td>
<td>0.55</td>
<td>0.32</td>
<td>0.09</td>
<td>0.05</td>
<td>0.02</td>
<td>0.18</td>
<td>0.75</td>
<td>0.78</td>
<td>0.54</td>
<td>0.48</td>
</tr>
<tr>
<td>10/40</td>
<td>1.00</td>
<td>0.49</td>
<td>0.62</td>
<td>0.25</td>
<td>0.18</td>
<td>0.02</td>
<td>0.02</td>
<td>0.09</td>
<td>0.89</td>
<td>0.81</td>
<td>0.80</td>
<td>0.43</td>
</tr>
<tr>
<td>20/10</td>
<td>1.00</td>
<td>0.48</td>
<td>0.67</td>
<td>0.93</td>
<td>0.17</td>
<td>0.01</td>
<td>0.00</td>
<td>0.07</td>
<td>0.80</td>
<td>0.54</td>
<td>0.89</td>
<td>0.15</td>
</tr>
<tr>
<td>20/20</td>
<td>0.87</td>
<td>0.51</td>
<td>0.74</td>
<td>0.66</td>
<td>0.17</td>
<td>0.02</td>
<td>0.15</td>
<td>0.73</td>
<td>0.75</td>
<td>0.20</td>
<td>0.95</td>
<td>0.81</td>
</tr>
<tr>
<td>20/40</td>
<td>0.92</td>
<td>0.34</td>
<td>0.47</td>
<td>0.64</td>
<td>0.05</td>
<td>0.02</td>
<td>0.11</td>
<td>0.76</td>
<td>0.60</td>
<td>0.89</td>
<td>0.48</td>
<td></td>
</tr>
<tr>
<td>30/10</td>
<td>0.94</td>
<td>0.42</td>
<td>0.82</td>
<td>0.85</td>
<td>0.09</td>
<td>0.00</td>
<td>0.04</td>
<td>0.12</td>
<td>0.79</td>
<td>0.56</td>
<td>0.99</td>
<td>0.76</td>
</tr>
</tbody>
</table>


