Evolutionary composition of QoS-aware web services: a many-objective perspective

Aurora Ramírez[a], José Antonio Parejo[b], José Raúl Romero[a,*], Sergio Segura[b], Antonio Ruiz-Cortés[b]

[a]*Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, 14071, Córdoba, Spain*
[b]*Department of Computing Languages and Systems, University of Sevilla, ETSII, Avda. de la Reina Mercedes, s/n, 41012, Sevilla, Spain*

Abstract

Web service based applications often invoke services provided by third-parties in their workflow. The Quality of Service (QoS) provided by the invoked supplier can be expressed in terms of the Service Level Agreement specifying the values contracted for particular aspects like cost or throughput, among others. In this scenario, intelligent systems can support the engineer to scrutinise the service market in order to select those candidates that best fit with the expected composition focusing on different QoS aspects. This search problem, *a.k.a.* QoS-aware web service composition, is characterised by the presence of many diverse QoS properties to be simultaneously optimised from a multi-objective perspective. Nevertheless, as the number of QoS properties considered during the design phase increases and a larger number of decision factors come into play, it becomes more difficult to find the most suitable candidate solutions, so more sophisticated techniques are required to explore and return diverse, competitive alternatives. With this aim, this paper explores the suitability of many-objective evolutionary algorithms for addressing the binding problem of web services on the basis of a real-world benchmark with 9 QoS properties. A complete comparative study demonstrates that these techniques, never before

*Corresponding author. Tel.: +34 957 21 26 60
Email addresses: `aramirez@uco.es` (Aurora Ramírez), `japarejo@us.es` (José Antonio Parejo), `jrromero@uco.es` (José Raúl Romero), `sergiosegura@us.es` (Sergio Segura), `aruiz@us.es` (Antonio Ruiz-Cortés)

applied to this problem, can achieve a better trade-off between all the QoS properties, or even promote specific QoS properties while keeping high values for the rest. In addition, this search process can be performed within a reasonable computational cost, enabling its adoption by intelligent and decision-support systems in the field of service oriented computation.

Keywords: QoS-aware web service composition, many-objective evolutionary algorithms, multi-objective optimization

1. Introduction

Current service oriented applications need to integrate third-party services, like authentication or persistent storage, as part of their core features. This integration clearly benefits code reuse and modularity, though it also introduces additional concerns. In this sense, the Quality of Service (QoS) experienced by end-users will also depend on the QoS provided by these external services. For example, if the persistence service is unavailable, the performance of the entire application would probably drop or even the system itself would become useless. In such a scenario, a careful selection of the services to be integrated is critical to determine the composition that better achieves an appropriate overall QoS at affordable cost. Moreover, the Service Level Agreement (SLA), *i.e.* the piece of a service contract where the level of service is determined, could even bring more alternatives into play, considering that a specific service provider might offer several QoS configurations for the same service provision. For example, Amazon Web Services (AWS) establishes up to 8 different deployment plans (instances) for their computing services (EC2) (Wada et al., 2012), which combined with other configurable options like the operating system, the available CPU and backup settings can lead to 16,991 possible configurations (García-Galán et al., 2013). Although highly demanded because of its flexibility, considering a larger set of configuration alternatives implies increasing the number of QoS properties coming into play and, consequently, finding appropriate trade-offs among them becomes extremely difficult. For instance, notice that an investment in CPU and exclusive dedication would improve response time but increasing the cost.

Therefore, deciding which are the most appropriate services to be included in an application, or their specific QoS configurations, is a challenging task for designers, the automatic support acquiring even more relevance as the number of decision factors increases. Here, intelligent systems may help to support them by applying search techniques to the exploration and selection of design alternatives. Consequently, analysing how different search methods behave and how they are influenced by the problem structure, e.g. different number of services being orchestrated in different ways, becomes important for these systems to gain efficiency and effectiveness.

The so-called QoS-aware Web Service Composition (QoSWSC) problem has been identified as a key research problem in the service oriented computing (SOC) field (Papazoglou et al., 2007), which is actually NP-hard (Bonatti & Festa, 2005; Ardagna & Pernici, 2007). Although this problem was originally formulated as a single-objective optimisation problem, notice that multiple, often conflicting QoS properties need to be simultaneously considered to address this problem (Zeng et al., 2004; Wada et al., 2012). For example, availability, response time, throughput or invocation cost can be clearly opposed, *e.g.* improving the availability of a service probably would imply an increase in cost. In general, identifying priorities among these attributes is not straightforward, and designers have to ensure an appropriate trade-off between all of them according to their interests.

Given the large number of alternatives to be analysed, a computational optimisation approach can serve to efficiently find the best orchestration of candidate services (Canfora et al., 2005). Both single-objective and multi-objective evolutionary algorithms (MOEAs) constitute a commonly used alternative in the literature, where the latter return a set of solutions, each one achieving a different trade-off between all the objectives (Coello Coello et al., 2007). Actually, the use of multi-objective optimisation algorithms has proved to be a more convenient approach to deal with the QoSWSC problem, as recently discussed by Wada et al. (2012); Moustafa & Zhang (2013); Suciu et al. (2013); Trummer et al. (2014); Yu et al. (2015). Having a set of alternative solutions to choose among facilitates better comprehension of the different possible trade-offs between the QoS properties involved in the decision-making process. In contrast, this information would not be available in advance when a single-objective evolutionary approach is applied using a weighted sum.

The application of multi-objective evolutionary approaches to the QoSWSC problem has been mostly focused on the selection of well-known approaches, like NSGA-II (Nondominated Sorting Genetic Algorithm II) or MOGA (Multi-Objective Genetic Algorithm), and considering up to 5 QoS properties as optimisation objectives. Nevertheless, when a large number of objectives need to be considered, the performance of these classical MOEAs tends to drop off as the complexity of the resulting optimisation problem increases. This factor has led to the appearance of new specific approaches, like many-objective evolutionary algorithms, which have emerged as an effective alternative to efficiently explore highly dimensional objective spaces (Ishibuchi et al., 2008). Similarly, many-objective evolutionary algorithms operate in accordance to the precepts of the multi-objective approach. In fact, for situations where engineers need to deal with a large number of decision criteria, as during the design of complex web service compositions considering multiple QoS properties, many-objective optimisation provides an excellent support mechanism.

In this paper, the QoSWSC problem is addressed from the emerging many-objective perspective considering a large number of QoS properties. More specifically, our research question can be phrased as follows: *Is the application of many-objective algorithms appropriate to address in a generalisable way the QoSWSC problem considering a diversity and large number of QoS properties?* To accurately respond to this question, a comparative study of different evolutionary algorithms is proposed with the aim of analysing their suitability when 9 diverse QoS properties constrain the problem statement. Notice that the jointly optimisation of a large number of objectives constitutes a real challenge to any optimisation approach. It could be expected that these algorithms, primarily conceived to deal with such a complexity, will provide a better performance than that obtained by long-standing MOEAs in terms of both the obtained QoS values and the expected balance among then. Finally, we include an in-depth discussion of the empirical insights obtained from the most fitting algorithm in terms of a representative subset of properties, such as runtime and design-time properties. The experimentation with many-objective approaches provides valuable information about how robust and effective these search methods are, which brings the opportunity to incorporate them into intelligent systems aimed at supporting the resolution of more realistic formulations of the QoSWSC problem.

The rest of the paper is organised as follows. Section 2 introduces the multi-objective evolutionary optimisation, as well as the bases for the QoSWSC problem as an optimisation problem. Section 3 describes the related work and then Section 4 explains the specific features of the implemented evolutionary approach, including the definition of the QoS properties considered to evaluate how objectives are met. A detailed performance analysis of the algorithms is conducted in Section 5, where findings and outcomes are also discussed. Then, the threats to validity concerning the presented study are detailed in Section 6. Finally, Section 7 outlines some concluding remarks.

2. Background

In this section, the key concepts of multi- and many-objective optimisation are introduced. Next, the QoS-aware binding of web services is presented as a search problem.

2.1. Multi- and many-objective evolutionary algorithms

Multi-objective evolutionary algorithms are population-based metaheuristics devoted to solve multi-objective optimisation problems (MOPs) (Coello Coello et al., 2007). Just like evolutionary algorithms (EAs), MOEAs define a set of candidate solutions, *i.e.* the population of individuals, which are modified through some iterations seeking for the generation of better solutions. Usually, each solution, also called individual, is encoded using a fixed-length numerical array, *i.e.* the genotype. Continuing the simile, the phenotype is defined as the real-world representation of the genotype.

After the random creation of the initial population, the algorithm starts an iterative process, as shown in Algorithm 1. In every generation, some individuals are selected to act as parents, and genetic operators, like crossover and mutation, are applied to create new solutions. On the one hand, the crossover operator recombines genetic information of two parents, resulting in one or more descendants (*a.k.a.* offspring) that tend to be similar to their parents. Therefore, this operator is expected to promote convergence in the search process. On the other hand, the mutation phase produces some alterations on a given offspring, *e.g.* changing one value in the genotype, with the aim of introducing diversity in the population. The generation ends with the selection of the set of individuals, from both the current population and the offspring pool, that will take part in the next population. Frequently, this replacement mechanism tries to promote the survival of the best solutions found so far, ensuring that diversity is also preserved. The evolutionary process continues until a stopping criterion, *e.g.* a maximum number of generations, is reached.

The main difference between EAs and MOEAs lies on the evaluation of individuals, which determines "how good" a solution is in solving the optimisation problem. In EAs, a unique fitness function is defined to measure the quality of the solutions, so individuals can be directly compared using this fitness value. Instead of defining a specific set of weights to aggregate several objectives, a multi-objective approach treats each objective as an independent function. Not only the problem of determining the weight to be assigned to each function is removed, but also the limitations offered by the use of aggregation functions (Deb, 2001) are overcome. Among others, the use of such a scalarisation function assumes the linearity of the MOP and non-convexity of the PF. Besides, it is not possible to assure that there would be a one-to-one correspondence between the solution optimising the weighted sum and a supposedly non-dominated solution.

Given that MOPs are characterised by the presence of 2 or more objectives,

Algorithm 1 Pseudocode of an evolutionary algorithm

```
Require: $maxGenerations$, $populationSize$
 1: $population \leftarrow$ createPopulation($populationSize$)
 2: evaluate($population$)
 3: $generation \leftarrow 0$
 4: while $generation <= maxGenerations$ do
 5:     $parents \leftarrow$ selection($population$)
 6:     $offspring \leftarrow$ crossover($parents$)
 7:     $offspring \leftarrow$ mutation($offspring$)
 8:     evaluate($offspring$)
 9:     $population \leftarrow$ replacement($population \cup offspring$)
10:     $generation + +$
11: end while
```

each evaluated solution contains a set of objective values. As for their comparison, the Pareto dominance concept is frequently included as a discerning criterion to choose between two solutions, a and b, which is defined as follows: a is said to dominate b, if and only if a is better or equal than b for all the objectives, and better for at least one objective than b. If this condition is not satisfied, individuals are referred as equivalent or non-dominated. Thus, the purpose of any MOEA is to find the set of non-dominated solutions, i.e. the Pareto set, establishing different trade-offs among all the objectives. Mapping these solutions onto the objective space allows getting the Pareto front (PF).

Some of the most well-known proposals in the field, like SPEA2 (Strength Pareto Evolutionary Algorithm 2) (Zitzler et al., 2001) and NSGA-II (Deb et al., 2002), are strongly based on the Pareto dominance principle to guide the search towards the PF. On the one hand, SPEA2 assigns a strength value to each individual, i, considering both the number of solutions it dominates and the solutions dominating i. On the other hand, NSGA-II ranks the population by fronts, where each front comprises those equivalent solutions that dominate solutions allocated in the following fronts. Regarding diversity preservation, SPEA2 uses the k-nearest neighbour method to estimate the density at any point of the objective space, whereas NSGA-II proposes a crowding distance to discard between solutions belonging to the same front. Additionally, SPEA2 also defines an archive of solutions with a fixed size, where non-dominated solutions are kept.

Certainly, SPEA2 and NSGA-II have shown a good performance in a variety of problem domains when 2 or 3 objectives are considered. However, real-world applications might require the definition of a greater number of objectives, which has lead to a growing interest in solving the so-called many-objective optimisation problems. Although the actual difference between multi- and many-objective problems has not been clearly stated in the literature (Purshouse & Fleming, 2007), most authors agree today with the idea that many-objective problems require the presence of at least 4 objectives (von Lücken et al., 2014; Deb & Jain, 2014). With the increasing complexity of MOPs, concepts like the Pareto dominance and distances, which characterise the aforementioned algorithms, lose the efficiency required to properly guide the search (Khare et al., 2003; Praditwong & Yao, 2007), motivating the appearance of more sophisticated techniques. In this sense, advances within the field of many-objective optimisation are mainly focused on the adaptation of the dominance principle, the inclusion of specific diversity preservation mechanisms and the use of quality indicators as key features to control the evolution (Wagner et al., 2007).

For instance, MOEA/D (Multiobjective Evolutionary Algorithm based on Decomposition) (Zhang & Li, 2007) proposes a decomposition approach creating a number of subproblems to be simultaneously optimised. Each subproblem associates a different weight to each objective and, as a result, multiple search directions are explored during the search.

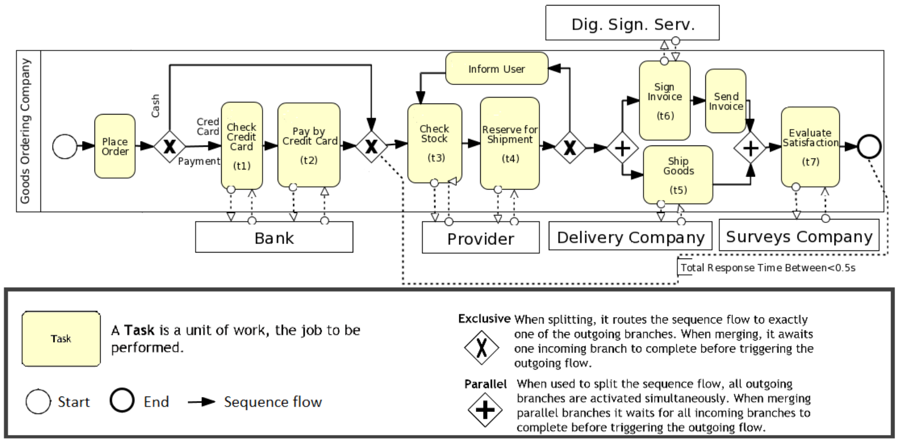
Modifying the classical Pareto dominance also serves to improve the performance of MOEAs, since the percentage of non-dominated solutions in a population rapidly grows when the number of objectives increases (Ishibuchi et al., 2008). ϵ-MOEA (Deb et al., 2003) defines a special type of dominance, called ϵ-dominance, which can be applied on an objective space divided in hypercubes or grids. Thus, solutions are compared considering the hypercubes they belong to, instead of its objective values. Moreover, the evolution process tries to generate an unique solution for each hypercube in favour of diversity, saving them in an archive. Similarly, GrEA (Grid-based Evolutionary Algorithm) (Yang et al., 2013) also proposes a landscape partition, though the grids are dynamically created in each generation. The sorting approach defined by NSGA-II is considered, as well as some diversity metrics based on the grids.

Another kind of algorithms are the so-called indicator-based approaches. An indicator allows summarising the quality of the overall PF in a real value (Coello Coello et al., 2007), so it can be used to guide the search process. This idea is explored by IBEA (Indicator-based Evolutionary Algorithm) (Zitzler & Künzli, 2004), which proposes a generic multi-objective evolutionary algorithm where the selected indicator is used in both the selection and the replacement stages. Another interesting approach is HypE (Hypervolume Estimation Algorithm) (Bader & Zitzler, 2011), in which the hypervolume (HV) indicator is estimated using Monte Carlo simulations. HV is one of the most frequently used measures to evaluate PFs, serving to calculate the hyper-area covered by the Pareto front. Another relevant indicator is spacing (S), which computes the diversity of the solutions composing the Pareto set.

Finally, NSGA-III (Deb & Jain, 2014) is a reference-point-based method that modifies the behaviour of NSGA-II regarding its diversity preservation technique. Instead of computing the crowding distance, this algorithm defines a set of well-distributed points that are used to promote the individuals that are close to these points at the replacement step.

2.2. QoS-aware binding of composite web services as an optimisation problem

The QoSWSC problem can be defined as the search for the best subset of candidate services to accomplish a composite service within a specific workflow. More precisely, the set of services requiring binding (henceforth named tasks) is identified. For each task t_i, the set of service providers available $S_i = \{s_{i,1}, \dots, s_{i,m}\}$ (named candidate services) is determined. This set can



A **Task** is a unit of work, the job to be performed.

Exclusive When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Parallel When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Figure 1: Goods ordering composite service, adopted from (Parejo et al., 2014)

be obtained by searching in a service registry, or by analysing the set of QoS configurations available in the SLA of the service. Thus, even when a single provider is available, multiple candidate services could be evaluated (one for each alternative QoS configuration provided by the SLA).

To illustrate this conceptual framework, Fig. 1 shows a goods ordering service using the Business Process Modelling Notation (BPMN). The example presents a business process composed of 7 tasks $(t_1, \ldots, t_7)$ with alternative providers including the payment process, the stock management, the delivery and the request of survey questions on the user satisfaction. Note that some of these tasks need to be developed following a specific sequence (*e.g.* t_1 and t_2), whilst others require more complex building blocks. For example, if a product is not available, the application reports about the delay, waiting for some time before repeating t_3 and t_4, *i.e.* a loop will be executed. It is worth noting that the same provider must be chosen for the tasks t_3 and t_4, since the reservation in t_4 refers to the stock of the specific provider queried in t_3. This constraint is denoted in the diagram using the elevation event of BPMN linked to both tasks (an arrow up inscribed in a circle). Next, t_5 and t_6, which belong to two different branches, can be performed in parallel. Finally, the completion of a user satisfaction survey is requested in task t_7.

Table 1: Service providers, candidate services and QoS values for the Goods ordering composite service

Task and services														
Actor	Bank				Provider				Delivery		Dig. sign.		Surveying	
Provider	A		B		C		D		E	F	G	H	I	J
Task	t_1	t_2	t_1	t_2	t_3	t_4	t_3	t_4	t_5	t_5	t_6	t_6	t_7	t_7
Candidate service	$s_{1,A}$	$s_{2,A}$	$s_{1,B}$	$s_{2,B}$	$s_{3,C}$	$s_{4,C}$	$s_{3,D}$	$s_{4,D}$	$s_{5,E}$	$s_{5,F}$	$s_{6,G}$	$s_{6,H}$	$s_{7,I}$	$s_{7,J}$
QoS properties														
Cost (in cents)	1.00	2.00	1.50	5.00	1.00	2.00	1.00	5.00	1.00	2.00	1.00	2.00	1.50	5.00
Exec. time (in seconds)	0.20	0.20	0.10	0.15	0.20	0.20	0.40	0.25	0.20	0.20	0.20	0.20	0.10	0.15

Once the structure of the composition and its tasks have been defined, a mechanism to choose among candidate services has to be specified. Here, the goal is to find the binding of services (χ) that maximises the global QoS (χ^*) according to the consumers' preferences. The set of QoS properties that need to be satisfied, such as the execution time, availability or cost, among others, is denoted by $\mathbb{Q}$. For each QoS property $q \in \mathbb{Q}$, a global QoS level, Q_q, can be reached from the individual QoS values stipulated by the agreement of each candidate service appearing in χ. Table 1 details the candidate services for the example of Fig. 1 and its QoS values in terms of cost and execution time. For instance, invoking the payment service t_2 of the provider A, $s_{2,A}$, costs 0.02\$. In this case, notice that the global cost Q_{cost} of a composite web service containing a loop inside will depend on the number of iterations performed.

In order to obtain the set of Q_q values of a specific binding of services, the QoS values of each $s_{i,j}$ in χ are aggregated using a utility function, U_q, where the specific expression to calculate such a function clearly depends on the nature of q. Thus, utility functions express user preferences, *i.e.* the obtained values allow users to decide if a given solution fulfils their expectations or satisfies the existing constraints for a given QoS property. For instance, a total cost of 0.2\$ could be fair for some users, but excessive for others.

Each utility function U_q also needs to consider the sort of blocks taking part in the composition workflow. The existence of conditional branches entails the possibility that only the services allocated in one branch will be executed, being a different scenario than the invocation of a sequence of them. In this sense, U_{time} for a sequence can be established as the sum of the execution time of all the services that compound that sequence, whilst for a branch the overall value can be defined as the maximum execution time of any path in this branch. However, U_{cost} for a branch is computed as the sum of the cost of tasks in each branch.

Furthermore, specific runtime conditions for loops and alternative branches also influence the calculation of every Q_q. On the one hand, the presence of a loop implies that one service could be invoked many times. On the other hand, not all the services in a conditional structure will be executed in every invocation. In such cases, the total cost will be affected by the choice among alternatives and the number of iterations in the loop, respectively. Since these parameters are unknown in advance, an estimation of the expected behaviour is usually adopted in the literature when defining the problem statement (Canfora et al., 2008). For the goods ordering service example, the average number of iterations per loop could be estimated to include the expected number of querying (t_3) and reservation (t_4) of products in stock. Similarly, the probability of executing each branch of the workflow should be considered. For example, a use of credit card in 80 per cent of payments could be assumed, whilst t_3 and t_4 are executed twice on average before bringing the order to completion. In total, the estimated global cost for a binding $\chi = (A, B, D, D, F, H, J)$ can be computed as $Q_{Cost}(\chi) = \text{Cost of switch}(\chi) + \text{Cost of Loop}(\chi) + \text{Cost of fork}(\chi) + Cost_7(\chi) = 0.8*(0.01+0.02)+2*(0.01+0.05)+(0.02+0.02)+0.05 = 0.23\$$. Since those values are estimations, the actual global QoS values provided can differ significantly in some invocations.

The final assignment of services into tasks is often obtained after a complex decision-making process, being its aim to find the solutions χ^* that maximise the utility of the global QoS values provided by the application. In the motivating example, where only two providers are available for each task, 128 (*i.e.* 2^7) different bindings are possible.

3. Related Work

The QoSWSC problem was introduced as a single-objective optimisation problem in (Zeng et al., 2004), where integer programming was applied to its resolution. Since then, several non-evolutionary approaches have been used to address this problem. An improved discrete immune optimisation algorithm based on particle swarm optimisation (IDIPSO) has been proposed in (Zhao et al., 2012), whereas in (Parejo et al., 2014) the authors used GRASP with path relinking to provide appropriate solution to the problems that required a response in short execution time. Those approaches are compiled in (Strunk, 2010; Jula et al., 2014). The first genetic approach was proposed by (Canfora et al., 2005), also considering a single-objective problem statement.

More recently, the optimisation problem has been addressed from a multi-objective perspective, applying either metaheuristics or other kind of approaches. In (Li & Yan-xiang, 2010) the multi-objective chaos ant colony optimisation algorithm is proposed, showing that it can outperform MOGA under specific conditions when dealing with 3 objectives (*i.e.* cost, time, and reliability). Precisely, MOGA is the basis of the evolutionary framework presented in (Wada et al., 2012), which also considers 3 QoS properties (*i.e.* throughput, latency and cost) to guide the search for Pareto optimal solutions. Reinforcement learning was the selected technique in (Moustafa & Zhang, 2013) to jointly optimise availability, response time and cost, where the experimentation was carried out over a synthetic dataset with only 4 tasks. Particle swarm optimisation (PSO) was also adopted in (Yin et al., 2014) to optimise 3 objectives. In this case, sequence and parallel structures constituted the only available building blocks to define the workflow of the composition.

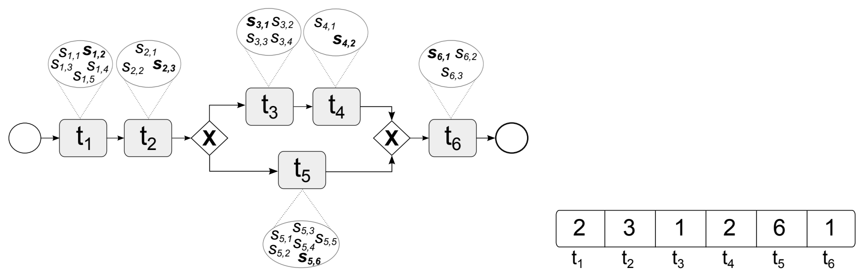
Metaheuristics approaches like EAs, scatter search and PSO were compared to exact methods in (Trummer et al., 2014) in order to optimise up to 5 objectives (*i.e.* response time, availability, throughput, successability, and reliability), extracted from the QWS Dataset (Al-Masri & Mahmoud, 2008). This dataset was also used in (Zhang, 2014) considering all the available QoS properties. In this case, the optimisation problem was solved using PSO, though the experimental study was only performed in terms of the execution time.

Variants of NSGA-II were proposed in (de Campos et al., 2010) to deal with 5 different objectives, where preference relations were included with the aim of improving the performance of the original algorithm. Finally, MOEA/D was the algorithm selected by (Suciu et al., 2013) to solve a 3-objective variant of the QoSWSC problem. An adaptation of this algorithm using two differential evolution approaches was compared against NSGA-II and GD3 algorithms, also considering different configurations of the number of tasks and candidate services.

A first recent approach using an evolutionary algorithm specifically conceived to deal with many objectives in order to solve the QoSWSC problem is (Yu et al., 2015). In such a paper an adaptation of NSGA-III, named F-MGOP, is proposed. Nevertheless, this work is strictly focused on data-intensive services, and F-MGOP is only compared against SPEA2 and NSGA-II in the empirical validation. In this case, the number of objective functions is limited up to 4 run-time properties (latency, execution cost, availability and accuracy), not usually constituting a real challenge to many-objective algorithms. On the other hand, the study here presented performs a wider and extensive empirical comparison considering diverse algorithms from the different families of many-objective evolutionary approaches and QoS properties of different nature. Furthermore, the conducted analysis (see Section 5) serves to generalise to any service composition the conclusions stated in (Yu et al., 2015) regarding the suitability of many-objective algorithms for optimizing the QoS of data-intensive compositions.

4. Optimisation model

In this section, the common elements of the evolutionary approach are presented, including the encoding of solutions, the genetic operators and the selection and replacement strategies. Evaluation is performed by calculating the


$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,4}$ $s_{1,5}$
$s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,1}$ $s_{3,2}$ $s_{3,3}$ $s_{3,4}$
$s_{4,1}$ $s_{4,2}$
$s_{5,1}$ $s_{5,2}$ $s_{5,3}$ $s_{5,4}$ $s_{5,5}$ $s_{5,6}$
$s_{6,1}$ $s_{6,2}$ $s_{6,3}$
t_1 t_2 x t_3 t_4 t_5 x t_6
2 3 1 2 6 1
t_1 t_2 t_3 t_4 t_5 t_6

(a) Phenotype (b) Genotype

Figure 2: Phenotype and genotype of a candidate individual

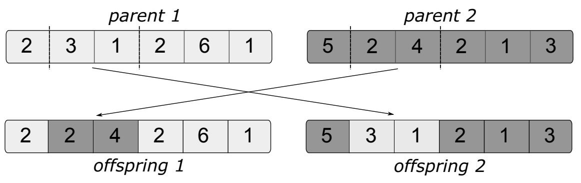
objective functions as explained later.

4.1. The evolutionary approach

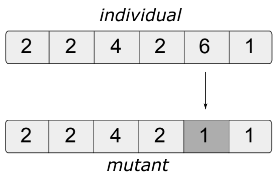
The genotype/phenotype mapping used in this work follows the approach proposed by (Canfora et al., 2005), and extensively used in the literature (Wada et al., 2012; Parejo et al., 2014). Here, each solution is encoded as an integer array, where each position in the genotype represents a task and its corresponding value, the selected service that will provide it. Consequently, the length of the genotype is equal to the number of tasks appearing in the workflow. Fig. 2 shows the correspondence between the phenotype and the genotype of each individual.

Initialisation. The initialisation of the population is a random procedure, *i.e.* for each task, a web service (highlighted in bold typeface) is randomly chosen from its list of candidate services.

Operators. The adopted genetic operators are those proposed by (Canfora et al., 2005). The *two points crossover* establishes two cut-points in the genotype of the parents, so each descendant is created by recombining one part of one parent and two parts of the other parent. The generated individuals represent two new compositions where the service assigned to each task is inherited from one of its two parents. An example of this procedure is depicted in Fig. 3a. With regard to the mutation procedure, the *one locus mutator* is performed after the crossover



(a) Crossover



(b) Mutator

Figure 3: Genetic operators

by choosing a random gen from the genotype of an offspring in order to change its value. As a result, a new binding is generated for the task associated to this position, as shown in Fig. 3b.

Selection and replacement. Notice that both strategies are defined by each evolutionary algorithm, since they are not domain-specific. Table 2 summarises the characteristics of these procedures for all the algorithms described in Section 2.1. The update mechanism of the external archive is also detailed when required. In this sense, two special considerations have been made regarding MOEA/D and ϵ-MOEA, since both algorithms do not define a maximum size for the archive of solutions. Looking for the fairest scenario, both algorithms have been adapted in order to return as many non-dominated solutions as the other algorithms can manage at each generation, *i.e.* the population size. Thus, MOEA/D will be executed without considering the archive, as recommended by its authors (Zhang & Li, 2007), and the non-dominated solutions will be extracted from the final population. In the case of ϵ-MOEA, if the number of non-dominated solutions is greater than the population size, the archive of solutions will be truncated by a post-processing step using the method proposed by SPEA2. Nevertheless, the final number of solutions would slightly vary from one algorithm to another due to their different ability to explore the search space.

Table 2: Selection and replacement strategies defined by each evolutionary algorithm

Algorithm	**Selection**	**Replacement**	**Archive update**
SPEA2	comparison of fitness values, combining a strength value and density information	offspring replace the current population	updated with non-dominated solutions at each generation
NSGA-II	tournament selection based on the ranking position (front) and the crowding distance	progressively stores individuals by fronts, crowding distance takes a decision on the last front	-
MOEA/D	for each individual, two random neighbours generate an offspring	each offspring is compared against its neighbours	not used
ϵ-MOEA	an individual within the population is recombined with another one belonging to the archive	the new individual replaces one or more members of the population according to the Pareto dominance	the ϵ-dominance and the already filled hypercubes determine whether the new individual is included or not
GrEA	tournament selection based on dominance and grid measures	progressively stores individuals by fronts, grid measures takes a decision on the last front	-
IBEA	tournament selection based on the indicator (fitness value)	removes individuals with worst fitness value	-
HypE	tournament selection based on the HV estimation	progressively stores individuals by fronts, minimum loss of HV is considered in the last front	-
NSGA-III	random selection	progressively stores individuals by fronts, reference points takes a decision on the last front	-

4.2. Objective functions

In the QoSWSC problem, the objective functions are those quality attributes that have to be optimised in order to achieve the best possible global quality of the composite web service. The 9 QoS properties of candidate services defined in the QWS Dataset (Al-Masri & Mahmoud, 2008) have been considered in this work:

- *Response time (T).* The required time to send a request and receive the response from the service, expressed in milliseconds.
- *Availability (A).* The ratio (percentage) of successful invocations.
- *Reliability (R).* A measure of the amount of error messages generated during the service execution, *i.e.* the ratio of error messages to the total messages.
- *Throughput (G).* The number of invocations to the service per second.
- *Latency (L).* The time required to respond to a request, expressed in milliseconds.
- *Successability (U).* The ratio (percentage) of requests that were correctly replied.
- *Compliance (C).* The ratio (percentage) of conformance with the WSDL (Web Services Description Language) specification proposed by the World Wide Web Consortium.
- *Best practices (B).* The ratio (percentage) of accomplishment of the WS-I Basic Profile, which establishes a set of requirements to promote interoperability.
- *Documentation (D).* The ratio (percentage) of description tags of the WSDL used in the service documentation, *e.g.* service name, operation name, etc.

As mentioned in Section 2.2, each candidate service provides its own QoS values, which should be combined in order to obtain the overall QoS value for each property in the composite service. Notice that each specific property covers a different quality aspect of a service. For instance, some properties like C, B or D may be more related to the design and development process, which may affect the correctness, error handling and maintainability of the system; others like T and L are strongly influenced by the service provider and the network infrastructure, similarly to R, A, G and U, which may depend on the runtime conditions and other external factors or operating errors. In most cases, a combination of different factors will determine the choice of the QoS properties to be globally considered.

In order to conduct the evaluation of the quality objectives, an utility function is applied for each QoS property q considering the type of building blocks in the given workflow (see Table 3). For instance, the response time (T) of a sequence of service invocations can be obtained as the sum of the individual T values of these services, whilst a loop containing that sequence should also take into account the expected number of iterations, k. P_i stands for the probability of executing the branch i.

It is worth mentioning that the last four functions, *i.e.* U, C, B and D, have been specifically designed for this work as a secondary contribution, whilst the rest were adopted from the literature (Zeng et al., 2004; Ardagna & Pernici, 2007; Canfora et al., 2005; Wang et al., 2007; Strunk, 2010). With the exception of T and L, all the properties have to be maximised. Furthermore, for the sake of simplicity, the former have been properly inverted in a preprocessing step, and their aggregation functions have been adapted accordingly.

Table 3: Utility functions

QoS property	*Sequence*	*Loop*	*Branch*	*Fork*
Response time (T)	$\sum_{i=1}^{m} T(a_i)$	$k \cdot \sum_{i=1}^{n} T(a_i)$	$\sum_{i=1}^{m} P_i \cdot T(s_i^b)$	$\min_{i=1}^{p} T(s_i^f)$
Availability (A)	$\prod_{i=1}^{m} A(a_i)$	$(\prod_{i=1}^{n} A(a_i))^k$	$\sum_{i=1}^{m} P_i \cdot A(s_i^b)$	$\prod_{i=1}^{p} A(s_i^f)$
Reliability (R)	$\prod_{i=1}^{m} R(a_i)$	$(\prod_{i=1}^{n} R(a_i))^k$	$\sum_{i=1}^{m} P_i \cdot R(s_i^b)$	$\min_{i=1}^{p} R(s_i^f)$
Throughput (G)	$\min_{i=1}^{m} G(a_i)$	$\min_{i=1}^{m} G(a_i)/k$	$\sum_{i=1}^{m} P_i \cdot G(s_i^b)$	$\min_{i=1}^{p} G(s_i^f)$
Latency (L)	$\sum_{i=1}^{m} L(a_i)$	$k \cdot \sum_{i=1}^{n} L(a_i)$	$\sum_{i=1}^{m} P_i \cdot L(s_i^b)$	$\min_{i=1}^{p} L(s_i^f)$
Successability (U)	$\prod_{i=1}^{m} U(a_i)$	$(\prod_{i=1}^{n} U(a_i))^k$	$\sum_{i=1}^{m} P_i \cdot U(s_i^b)$	$\sum_{i=1}^{p} U_i \cdot U(s_i^f)$
Compliance (C)	$(\sum_{i=1}^{m} C(a_i))/n$	$(\sum_{i=1}^{m} C(a_i))/n$	$\sum_{i=1}^{m} P_i \cdot C(s_i^b)$	$(\sum_{i=1}^{m} C(a_i))/n$
Best practices (B)	$(\sum_{i=1}^{m} B(a_i))/n$	$(\sum_{i=1}^{m} B(a_i))/n$	$\sum_{i=1}^{m} P_i \cdot B(s_i^b)$	$(\sum_{i=1}^{m} B(a_i))/n$
Documentation (D)	$(\sum_{i=1}^{m} D(a_i))/n$	$(\sum_{i=1}^{m} D(a_i))/n$	$(\sum_{i=1}^{m} D(s_i^b))/n$	$(\sum_{i=1}^{p} D(s_i^f))/n$

5. Experimentation

In this section, the proposed experimental framework[1] is detailed. Firstly, the design of the experimentation is motivated by explaining how the obtained results have been validated. Next, the experimentation set-up and algorithm parametrisation are described. The analysis of the results is provided not only from the evolutionary perspective, but also on the degree to which each evolutionary algorithm fits the optimisation of the QoS properties under study. In addition, a more thorough analysis of the approach, including its advantages and limitations, is finally discussed at the end of this section.

5.1. Experimentation rationale

The research methodology applied in this work is an empirical study based on a sequence of controlled experiments, as the standard methodology in optimisation approaches. This methodology enables the isolation and control of the factors that might influence the performance of the algorithms and the quality of the resulting service compositions, thus providing a fair comparison framework. The experimentation has been formulated according to the intrinsic characteristics of the optimisation problem under consideration: (a) the number of tasks

[1]Additional material regarding the problem instances, experimentation results and statistical tests is available for reproducibility from `http://www.uco.es/grupos/kdis/sbse/RPRSR15`

and the number of candidate services per task mostly determine the size of search space; (b) the specific QoS values of each candidate are required to compute the global QoS value; and (c) the composition structure of the workflow (*i.e.* nested loops, parallel flows, alternative branches, etc.) determines which utility functions are computed.

Bearing these factors in mind, combining all the different configurations and complexities would imply an extremely large number of executions, leading to an unaffordable combinatorial explosion. Therefore, two representative experiments have been conducted in order to ensure meaningful results and conclusions:

Experiment #1 It considers web service compositions having a maximum of 10, 20, 30, 40, or 50 tasks, where each task contains a different set comprised of 1 to 11 candidate services, according to the parametrisation given in Table 4. Their composition is randomly chosen, where a total of 15 problem instances have been generated, *i.e.* 3 instances per each maximum number of tasks, each one associated to a different set of candidate services but sharing the workflow. Thus, this experiment is based on a wide spectrum of problem sizes.

Experiment #2 In order to validate the conclusions drawn from Experiment #1, Experiment #2 should serve to prove that the fixed structure, *i.e.* the workflow, does not have a marked influence on the outcomes. Therefore, 15 different structures of composition were generated for 3 representative instances, *i.e.* 10, 30 and 50 tasks, leading to a total of 45 problem instances. If the relative performance of the algorithms does not change in terms of an statistical significant difference, then it could be inferred that the conclusions are valid under the conditions of Experiment #1. All the problem instances used in these experiments were generated by the instance generator proposed in (Parejo et al., 2014), whose parameters are shown in Table 4.

All the experiments are conducted using the dataset proposed by (Al-Masri

Table 4: Problem instances generation parameters

Composition structure parameters	
Max. number of tasks	10, 20, 30, 40, 50
Prob. control flows	0.20
Prob. loops	0.45
Prob. branches	0.45
Prob. flows	0.10
Max. nesting levels	5.00
Runtime flow parameters	
Iterations per loop	Gaussian distribution ($\mu = 5.00$, $\sigma = 1.50$)
Number of branches	2.00
Candidate service parameters	
Candidates per task	Gaussian distribution ($\mu = 5.00$, $\sigma = 2.00$)
Candidates for each task	Randomly chosen from the dataset

& Mahmoud, 2008), which provides the specific QoS values for each candidate service. This dataset has been extensively used in the QoS-aware services computing area (Strunk, 2010) and, particularly, in the recent literature about the QoSWSC problem. It is comprised of QoS values obtained after monitoring 2,507 real world and publicly available web services, which contributes to make more realistic experiments within the field.

The results of both experiments have been analysed similarly. Each algorithm has been executed 30 times for each problem instance considering different random seeds. Then, the hypervolume and spacing indicators have been computed over the returned Pareto fronts, taking average values, to compare the quality of the set of solutions returned by each evolutionary algorithm (see Section 2.1). Both indicators vary in the range [0,1] and should be maximised. Since the hypervolume requires all the objective values to fall into the range [0,1], a post-processing step has to be performed in order to normalise the objective values of all the solutions returned. Next, three non-parametric statistical tests (Derrac et al., 2011; Arcuri & Briand, 2011) have been executed to assess the differences in the performance of the algorithms in terms of the aforementioned indicators. Firstly, the Friedman test for multiple comparison is carried out, where the null hypothesis, H_0, establishes that all the algorithms perform equally well. This test provides a ranking of algorithms, and a critical value to decide whether H_0 can be rejected at a certain level of significance $(1 - \alpha)$. However, this test can only report the existence of significant differences, so a post-procedure has to be applied in order to reveal the sort of these differences. This is the case of the Holm test, where the best algorithm found by the Friedman test is compared against the rest of algorithms.

Additionally, the Cliff's Delta test has been executed to perform pairwise comparisons using an effect-size measurement (Arcuri & Briand, 2011). This method allows classifying the difference between pairs of algorithms as negligible, small, medium or large on the basis of specific thresholds (Romano et al., 2006).

5.2. Experimental set-up

The optimisation approach as well as the proposed experiments have been coded in Java. The evolutionary algorithms have been implemented using the JCLEC framework (Ventura et al., 2007; Ramírez et al., 2015). The experiments were run on a HPC cluster of 8 compute nodes with Rocks cluster 6.1 x64 Linux distribution. Each node comprises two Intel Xeon E5645 CPUs with 6 cores at 2.4 GHz and 24 GB DDR memory.

Table 5 shows the parametrisation of the different evolutionary implementations. Notice that some part of the configuration is common to all the algorithms. Both the population size and the maximum number of evaluations have been fixed after some preliminary experimentation. Crossover and mutation probabilities have been configured in accordance to the values proposed by (Canfora et al., 2005). The rest of parameters have been set following each author's recommendation.

As for the specific set-up of the algorithms, it should be noted that IBEA applies the ϵ-indicator to guide the search, since the exact computation of hypervolume would be prohibitive. The Tchebycheff approach has been selected as the evaluation mechanism applied by MOEA/D, which allows the presence of objective functions with different scales as happens in this optimisation prob-

Table 5: Parameter set-up

Common parameters	
Population Size	165
Max. evaluations	33,000
Crossover probability	0.70
Mutation probability	0.10
SPEA2	
Parents selector	Binary tournament
Archive size	165
k-th neighbour	12
MOEA/D	
Neighbourhood size (τ)	10
Max. replacements (Nr)	4
No. weight vectors	165 (H=3)
GrEA	
Divisions (div)	10
IBEA	
Scaling factor (k)	0.05
HypE	
Sampling points (M)	10,000
NSGA-III	
No. reference points (boundary layer)	165 (p=3)
No. reference points (inside layer)	66 (p=2)

lem. Moreover, ϵ-MOEA has been modified to internally set the lengths of the hypercubes since they depend on the specific problem instance being solved. Given a workflow composition, the minimum and maximum values that a solution could reach for each QoS property are estimated and used to define 10 hypercubes with equal length, *i.e.* the same number of hypercubes established for GrEA.

5.3. Experiment #1

In this section, the results provided by Experiment #1 are shown and discussed. A comparative study is first presented in terms of the evolutionary performance. Secondly, relevant aspects of the search process regarding the trade-offs reached between QoS properties are discussed. Finally, algorithms are also compared in terms of their execution time.

5.3.1. Results and statistical analysis

After the execution of all the algorithms, the Friedman test determines the ranking position obtained by each algorithm regarding the HV, as shown in the third column of Table 6, where ϵ-MOEA is reported to achieve the best ranking. In addition, this test reports a statistics, z, at the specified significance level ($\alpha = 0.01$) in order to determine whether H_0 can be rejected. If the obtained value, which follows a F-Distribution, is greater than a critical threshold, then there exist significant differences among the algorithms. In this case, that condition is satisfied given that the critical value, 2.8272, is lower than the respective statistics, $z = 63.1879$. In order to reveal the sort of those differences, the Holm post-procedure is executed, being its outcomes shown in the fourth column of Table 6. In this case, the test has indicated that the control algorithm (ϵ-MOEA) is better than those algorithms having an α/i value lower than 0.025. Consequently, only HypE and NSGA-II perform equally well than ϵ-MOEA, *i.e.* all of them maintain a good trade-off between convergence and divergence along the search process and, as a result, these algorithms return an equivalent set of high-quality solutions.

Table 6: Statistical comparison of hypervolume in Experiment #1

i	*Algorithm*	*Ranking (Friedman)*	α/i *(Holm)*
7	NSGA-III	8.0000	0.0071
6	SPEA2	6.1333	0.0083
5	GrEA	6.0000	0.0100
4	MOEA/D	4.8000	0.0125
3	IBEA	4.2667	0.0167
2	NSGA-II	3.4000	0.0250
1	HypE	2.0000	0.0500
0	**ϵ-MOEA**	**1.4000**	

Next, as explained in Section 5, the Cliff's Delta test allows analysing the relative performance of pairs of algorithms. Table 7 compiles the results for all the possible pairwise comparisons, each cell showing the estimated difference in terms of HV and indicating whether such a value can be considered as negligible (n), small (s), medium (m) or large (l) at the significance level $\alpha = 0.01$. For instance, looking at the fourth row, it can be observed that, even when the previous statistical tests do not reveal a significant difference among ϵ-MOEA, NSGA-II and HypE, the difference between ϵ-MOEA and NSGA-II (0.46) has been classified as medium. On the other hand, the difference between ϵ-MOEA and HypE (0.21) is rather small. In addition, differences among SPEA2, MOEA/D and GrEA are reported as nearly negligible.

Table 7: Results of the Cliff's Delta test for hypervolume (n=negligible, s=small, m=medium, l=large) ($\alpha = 0.01$)

Algorithm	SPEA2	NSGA-II	MOEA/D	ϵ-MOEA	GrEA	IBEA	HypE	NSGA-III
SPEA2	-	-0.50 (l)	-0.19 (s)	-0.79 (l)	-0.05 (n)	-0.37 (m)	-0.69 (l)	0.96 (l)
NSGA-II	0.50 (l)	-	0.23 (s)	-0.46 (m)	0.48 (l)	0.16 (s)	-0.28 (s)	0.98 (l)
MOEA/D	0.19 (s)	-0.23 (s)	-	-0.55 (l)	0.13 (n)	-0.13 (n)	-0.48 (l)	0.96 (l)
ϵ-MOEA	0.79 (l)	0.46 (m)	0.55 (l)	-	0.69 (l)	0.58 (l)	0.21 (s)	0.92 (l)
GrEA	0.05 (n)	-0.48 (l)	-0.13 (n)	-0.69 (l)	-	-0.29 (s)	-0.60 (l)	0.96 (l)
IBEA	0.37 (m)	-0.16 (s)	0.13 (n)	-0.58 (l)	0.29 (s)	-	-0.44 (m)	0.95 (l)
HypE	0.69 (l)	0.28 (s)	0.48 (l)	-0.21 (s)	0.60 (l)	0.44 (m)	-	0.93 (l)
NSGA-III	-0.96 (l)	-0.98 (l)	-0.96 (l)	-0.99 (l)	-0.96 (l)	0.95 (l)	-0.99 (l)	-

Table 8: Statistical comparison of spacing in Experiment #1

i	*Algorithm*	*Ranking (Friedman)*	α/i *(Holm)*
7	IBEA	8.0000	0.0071
6	HypE	6.5333	0.0083
5	GrEA	6.4000	0.0100
4	NSGA-III	4.7333	0.0125
3	ϵ-MOEA	4.0000	0.0167
2	SPEA2	2.7333	0.0250
1	MOEA/D	2.6000	0.0500
0	**NSGA-II**	**1.0000**	

Focusing on the spacing indicator, Table 8 shows the results of Friedman and Holm tests. The resulting z value is 202.1765, so strong differences on the performance of the algorithms might be expected ($2.8272 << z$). Again, the Holm test determines that the control algorithm performs better than those algorithms having an α/i value lower than 0.025. NSGA-II is shown to be able to generate a greater variety of solutions than that provided by most of the many-objective algorithms, even though it has not significant differences with SPEA2 and MOEA/D.

Table 9 compiles the obtained differences among the 8 algorithms in terms of S after performing the Cliff's Delta test. As can be observed, SPEA2, NSGA-II, MOEA/D and ϵ-MOEA clearly outperform the rest of algorithms, since most of the differences are classified as large.

Table 9: Results of the Cliff's Delta test for spacing (n=negligible, s=small, m=medium, l=large) ($\alpha = 0.01$)

Algorithm	SPEA2	NSGA-II	MOEA/D	ϵ-MOEA	GrEA	IBEA	HypE	NSGA-III
SPEA2	-	-0.88 (*l*)	-0.07 (*n*)	0.41 (*m*)	0.93 (*l*)	0.93 (*l*)	0.93 (*l*)	0.71 (*l*)
NSGA-II	0.88 (*l*)	-	0.93 (*l*)	0.96 (*m*)	0.93 (*l*)	0.93 (*l*)	0.93 (*l*)	0.93 (*l*)
MOEA/D	-0.07 (*n*)	-0.93 (*l*)	-	0.54 (*l*)	0.93 (*l*)	0.93 (*l*)	0.93 (*l*)	0.80 (*l*)
ϵ-MOEA	-0.41 (*m*)	-0.96 (*l*)	-0.54 (*l*)	-	0.96 (*l*)	0.93 (*l*)	0.98 (*l*)	0.27 (*s*)
GrEA	-1.00 (*l*)	-1.00 (*l*)	-1.00 (*l*)	-0.96 (*l*)	-	0.95 (*l*)	0.02 (*n*)	-1.00 (*l*)
IBEA	-1.00 (*l*)	-1.00 (*l*)	-1.00 (*l*)	-1.00 (*l*)	-0.95 (*l*)	-	-0.93 (*l*)	-1.00 (*l*)
HypE	-1.00 (*l*)	-1.00 (*l*)	-1.00 (*l*)	-98 (*l*)	-0.02 (*n*)	0.93 (*l*)	-	-0.99 (*l*)
NSGA-III	-0.71 (*l*)	-1.00 (*l*)	-0.80 (*l*)	-0.27 (*s*)	0.93 (*l*)	0.93 (*l*)	-0.93 (*l*)	-

From the obtained results, it is worth noticing that SPEA2 and NSGA-II behave similarly with respect to S, since they reach the first positions in the ranking. However, NSGA-II clearly outperforms SPEA2 in terms of HV. In this sense, NSGA-II has shown good scalability when a high number of objectives is considered, only being overtaken by some specific many-objective approaches like ϵ-MOEA and HypE.

Regarding the many-objective evolutionary algorithms, the decomposition approach proposed by MOEA/D, which is aimed at maintaining diversity during the search, allows obtaining good S values. However, it has also a negative impact on the level of optimisation reached by the solutions, as shown in Table 6.

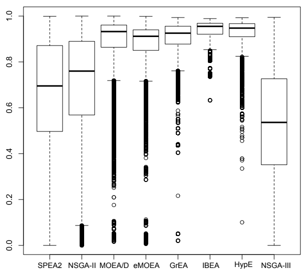
Table 10: Experiment #1: Best algorithms for each QoS property (expressed as percentages)

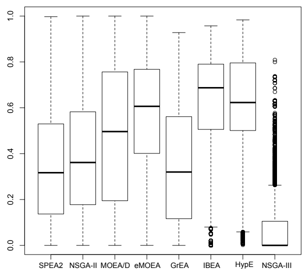
QoS Property	SPEA2	NSGA-II	MOEA/D	ϵ-MOEA	GrEA	IBEA	HypE	NSGA-III
Response time (T)	0.00	0.00	0.00	0.00	6.67	**53.33**	40.00	0.00
Availability (A)	0.00	6.67	0.00	13.33	0.00	**40.00**	**40.00**	0.00
Reliability (R)	0.00	0.00	0.00	13.33	0.00	6.67	**80.00**	0.00
Throughput (G)	0.00	0.00	0.00	13.33	0.00	6.67	**80.00**	0.00
Latency (L)	0.00	0.00	0.00	0.00	0.00	33.33	**66.67**	0.00
Successability (U)	0.00	0.00	0.00	6.67	0.00	40.00	**53.33**	0.00
Compliance (C)	0.00	0.00	0.00	6.67	13.33	26.67	**53.33**	0.00
Best practices (B)	13.33	0.00	6.67	**40.00**	0.00	20.00	20.00	0.00
Documentation (D)	0.00	0.00	0.00	33.33	6.67	**40.00**	20.00	0.00

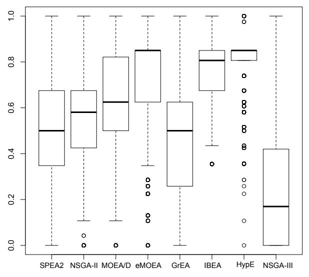
The same behaviour is observed when GrEA and NSGA-III are executed, both of them having problems to properly converge to the PF. Just the opposite situation comes about with IBEA or HypE, since both algorithms return better results for the HV than for S. As a general matter, the joint optimisation of both indicators is a complicated task, and only ϵ-MOEA and NSGA-II have achieved good ranking positions in both cases.

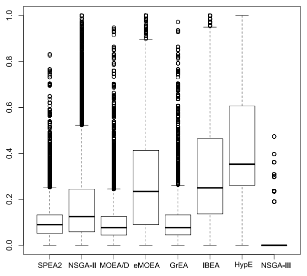
5.3.2. Evolutionary influence on QoS properties

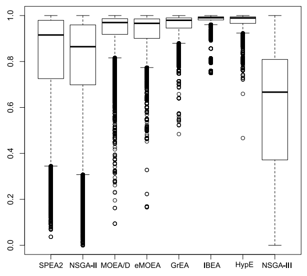
This section discusses the existing relation between algorithms and QoS properties. Table 10 provides the big picture of how good each algorithm is for a QoS property. This has been performed by calculating the average values of each property within the PF and, next, counting the number of times that each algorithm achieves the highest value for each property. All the problem instances were considered. Such values are expressed as percentages, the best value for each QoS property being shown in bold typeface. As can be observed, some specific many-objective approaches like ϵ-MOEA, IBEA and HypE reach the best percentages. Moreover, ϵ-MOEA and HypE have the ability to generate high quality solutions for some specific QoS properties without demoting the trade-off among all of them (see Table 6). Notice that the best set of alternative solutions are provided by HypE and ϵ-MOEA algorithms, what can become a relevant factor when software engineers have not a clear preference on

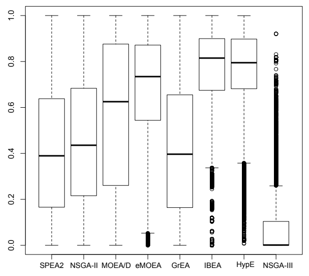

(a) Response time (T)

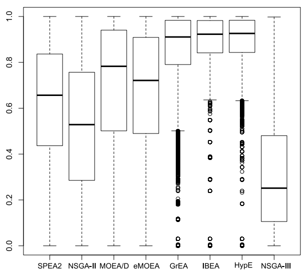

(b) Availability (A)

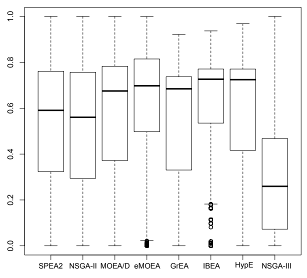

(c) Reliability (R)


(d) Throughput (G)


(e) Latency (L)


(f) Successability (U)


(g) Compliance (C)


(h) Best practices (B)

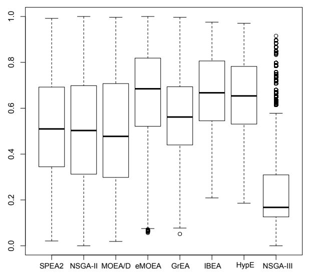

(i) Documentation (D)

Figure 4: Box plots of the distribution of QoS values in the Pareto front found by each algorithm

the properties to be optimised.

The separation of QoS properties in runtime (G, A, L, U, R, and T) and design-time (C, B, or D) can provide further insights. For the first case, Table 10 shows that HypE provides the best average values, since it achieves the best average values for 5 out of the 6 runtime properties. Regarding design-time properties, this algorithm also provides the best average values for the standards compliance (C) property, even when for the three properties IBEA has reached the best trade-off. Consequently, if runtime properties are promoted against design-time properties, HypE seems to be the most appropriate choice. Similarly, this algorithm provides the best global trade-off.

Figure 4 shows how the QoS values returned by each algorithm are distributed. For the sake of clarity, all the QoS properties are normalised, and have to be maximised. Notice that differences among algorithms become more distinct. Firstly, ϵ-MOEA, IBEA and HypE obtain not only similar distributions for design-properties, but also a good balance among the rest of attributes. Secondly, NSGA-II provides a wide range of QoS values for all the properties, even when the specific values are lower than those obtained by the aforementioned approaches. In addition, it can be observed that some QoS properties are easier to optimise than others. For instance, latency (L) is highly optimised by most of the algorithms, values greater than 0.8 being frequently achieved. On the contrary, differences between the algorithms are more noticeable for availability (A) and reliability (R).

5.3.3. Analysis of computational cost

As the number of QoS properties increases and the use of more sophisticated algorithms becomes more necessary, it is important to confirm that the execution time is suitable to perform the decision-making process. Even though this approach is framed within the design phase, where requirements related to execution time can be met more flexibly, an excessive computational cost could still limit the general adoption of many-objective evolutionary algorithms. Fig. 5 shows the average execution time for all the problem instances used in this experiment, depicting the scalability of the algorithms with respect to the number

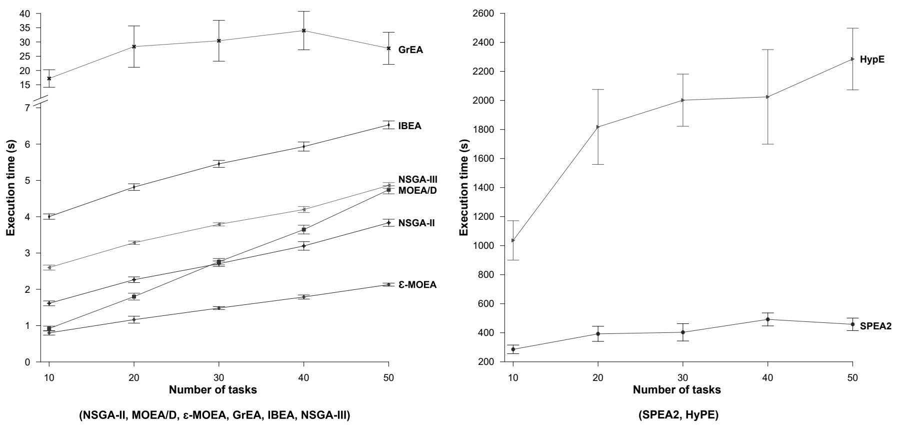


Figure 5: Average execution time relative to the number of tasks

of tasks. Error bars represent the standard deviation. As can be seen, most of the algorithms require just a few seconds to compute the overall Pareto front, a soft linear increase being observed as the number of tasks grows. Only SPEA2 and, especially, HypE require several minutes to find all the solutions comprising the Pareto front. The software engineer should consider whether such time level is manageable for the project conditions. In addition, it should be noted that obtaining high quality solutions is independent of the execution time required by each specific algorithm. For instance, NSGA-II and ϵ-MOEA are reported as efficient algorithms while they provide the best set of solutions according to the experiment conducted in Section 5.3.1.

5.4. Experiment #2

This experiment serves to analyse the influence of the composition structure on the evolutionary performance or on the optimisation of the QoS properties. Experimentation is performed similarly to Experiment #1.

Table 11 shows the comparison of algorithms in terms of the hypervolume indicator, and reveals the outcomes of the Friedman test considering all the problem instances generated for Experiment #2. As can be observed, the ranking positions for the algorithms are the same that those obtained in Experiment

#1, except for IBEA. In this case, z is equal to 220.9533, whereas the critical value is 2.6977. Consequently, since $2.6977 < z$, it can be concluded that there exist significant differences between the algorithms, and the threshold given by the Holm test, 0.05, indicates that ϵ-MOEA is statistically better than the rest of algorithms, except for HypE.

Table 11: Statistical comparison of hypervolume in Experiment #2

i	*Algorithm*	*Ranking (Friedman)*	*α/i (Holm)*
7	NSGA-III	8.0000	0.0071
6	SPEA2	6.4222	0.0083
5	GrEA	5.7778	0.0100
4	IBEA	4.6667	0.0125
3	MOEA/D	4.6444	0.0167
2	NSGA-II	2.9556	0.0250
1	HypE	1.9556	0.0500
0	**ϵ-MOEA**	**1.5778**	

Table 12: Statistical comparison of spacing in Experiment #2

i	*Algorithm*	*Ranking (Friedman)*	*α/i (Holm)*
7	IBEA	8.0000	0.0071
6	HypE	6.6222	0.0083
5	GrEA	6.3333	0.0100
4	NSGA-III	4.3333	0.0125
3	ϵ-MOEA	3.9778	0.0167
2	SPEA2	4.3463	0.0250
1	MOEA/D	2.4889	0.0500
0	**NSGA-II**	**1.0000**	

The statistical tests have been computed for the spacing indicator as well (see Table 12). In this case, z is equal to 453.6330, whereas the critical value remains the same. Again, significant differences are revealed after executing the Holm test, which rejects all the hypothesis when comparing the control algorithm, NSGA-II, against the rest of evolutionary approaches.

As previously performed in Experiment #1, Table 13 shows the existing relation between optimisation algorithms and QoS properties. Notice that ϵ-MOEA

Table 13: Experiment #2: Best algorithms for each QoS property (expressed as percentages)

QoS Property	SPEA2	NSGA-II	MOEA/D	ϵ-MOEA	GrEA	IBEA	HypE	NSGA-III
Response time (T)	0.00	0.00	0.00	0.00	4.44	46.67	**48.89**	0.00
Availability (A)	2.22	2.22	0.00	13.33	0.00	24.44	**57.78**	0.00
Reliability (R)	0.00	0.00	2.22	6.67	0.00	15.56	**75.56**	0.00
Throughput (G)	0.00	0.00	0.00	4.44	2.22	13.33	**80.00**	0.00
Latency (L)	0.00	0.00	0.00	0.00	2.22	33.33	**64.44**	0.00
Successability (U)	2.22	2.22	0.00	13.33	0.00	17.78	**64.44**	0.00
Compliance (C)	0.00	0.00	0.00	2.22	8.89	**57.78**	31.11	0.00
Best practices (B)	6.67	2.22	0.00	**44.44**	8.89	17.78	20.00	0.00
Documentation (D)	4.44	0.00	0.00	**40.00**	6.67	20.00	28.89	0.00

generates the best solutions in terms of documentation (D) and best practices (B), whereas IBEA seems to promote solutions with good values of standards compliance (C). Again, HypE is mostly focused on the search of solutions that satisfy the 6 runtime properties.

As can be observed, the obtained results remain rather similar to those discussed above. IBEA is likely to be the only exception to this statement, as in Experiment #1 had reach the best position to deal with design-time properties. More specifically, in this case IBEA performs worse regarding availability (A) and documentation (D) and, even when it behaves much better in terms of standards compliance (C), it is outperformed by ϵ-MOEA and HypE globally for the design-time properties. Between the latter approaches, ϵ-MOEA tends to demote the standards compliance (C), whereas HypE reaches a better balance among all the QoS properties. Nevertheless, notice that the three algorithms behave very similarly for the three design-time properties according to the distribution of QoS values (see Fig. 4), which implies that any minor change in the selected problem instances could modify their ranking positions.

With regard to the composite structure, it does not affect the relative performance of algorithms from an evolutionary perspective, as can be observed from the results. In global terms, the ranking positions remain the same, and the observed strengths and weaknesses of each evolutionary approach to explore the search space are due to other characteristics of the QoSWSC problem, i.e. its highly combinatorial nature and the number of objectives. The observed differences between both experiments can be explained by the fact that Experiment #2 considers a greater number of problem instances, and the addition of new workflow structures. This might influence the returned QoS values and, consequently, the results of the indicators.

5.5. Discussion of results

Understanding the advantages and limitations of the experimental findings can give awareness of the applicability of the proposed approach. Regarding its advantages, the comparative study has provided novel evidences of the performance of six many-objective algorithms to solve the QoSWSC problem, comparing them with two classical multi-objective algorithms. In this sense, results have shown that differences on the evolutionary performance of the algorithms are mostly due to the number of objectives and, consequently, their behaviour is shown to be significantly robust in all cases. On the one hand, many-objective approaches like ϵ-MOEA and HypE have proven to be more effective search methods than multi-objective algorithms in terms of the obtained QoS values. More specifically, experimental results show that ϵ-MOEA provides the best values for the QoS properties being optimised, whereas HypE reaches a better trade-off between the values of the target QoS properties. Even so, they tend to obtain less variety of web service compositions.

An in-depth analysis of the solutions returned by each algorithm points out that some algorithms, such as HypE and IBEA, are able to generate solutions with highly optimised values for some specific QoS properties of runtime (reliability and throughput) and design-time (documentation), respectively. At the same time, they maintain a good balance among the rest of properties. Hence, many-objective approaches become especially well-suited in those cases in which these properties are of particular interest to the engineer. To the best of our knowledge, this kind of study had never been conducted before in the context of the QoSWSC problem. If properly used, this information can be also exploited by the intelligent system aimed at providing the engineer with valuable heuristics for the selection of the most appropriate algorithm for each QoS property.

This proposal has some limitations, too. For instance, some of the algorithms are only suitable to solve the problem at design-time due to their high computational complexity. Nevertheless, notice that their execution time is still affordable at this stage of the development. Even some of the many-objective algorithms applied here, like IBEA, NSGA-III and MOEA/D, are faster than multi-objective approaches like SPEA2. Furthermore, a low execution time does not necessarily conflict with the generation of high quality solutions, as demonstrated by ϵ-MOEA. From the point of view of the decision-making process, the engineer could consider as a drawback the need of selecting a specific solution from the final Pareto front. This is usual when dealing with Pareto-based approaches, which could be configured or adapted to return a smaller set of solutions to choose from.

6. Threats to validity

As any research methodology, the experimental study proposed here presents limitations that should be clearly pointed out. These are described next in terms of internal and external validity threats, including the decisions taken to mitigate their impact.

Internal validity. This refers to whether there is sufficient evidence to support the conclusions and the sources of bias that could compromise those conclusions. In order to minimise the impact of external factors in the obtained results, all the algorithms were executed 30 times per problem instance (market of candidate services and structure of the composition) to compute averages. Moreover, statistical tests were performed to ensure the significance of the differences identified between the results obtained by the compared proposals. Finally, the experiments have been executed in a remote cluster of computers, so a stable experimentation platform was provided.

External validity. This is concerned with how the experiments capture the research objectives and the extent to which the drawn conclusions can be generalised. This can be mainly divided into limitations of the approach and generalisability of the conclusions. Regarding the limitations, experiments did not reveal significant differences for all the pairwise comparisons between algorithms. Nonetheless, the obtained results have provided solid insights when comparing the general behaviour of multi-objective evolutionary algorithms, mostly focused on maintaining diversity, and many-objective approaches, which tend to work better in terms of hypervolume.

Regarding the generalisability of conclusions, the parameters and the size of the analysed problem instances were chosen considering the most common values used in the literature (Strunk, 2010). Additionally, Experiment #2 was performed to ensure the generalisability of the results independently of the specific composition structure used in the problem instances (workflow layout). In this case, up to 45 different problem instances were randomly generated in order to compare the performance of the proposals with disparate composition structures. Since the rankings of the algorithms mostly remain unaltered with respect to Experiment #1, and differences are statistically significant in all cases, it can be concluded that results are generalisable for the composition structures having the applied parameters. Finally, our experimental study does not take into account neither global QoS constraints nor interdependence constraints. Although most of the selected algorithms can be adapted to deal with constrained problems, conclusions regarding their performance cannot be extrapolated from the current study.

7. Concluding Remarks

This paper presents a comparative study on the suitability and performance of different multi- and many-objective algorithms to deal with the QoS Web Service Composition problem, which has been identified as a key issue in the field of SOC. This problem has been already addressed from a multi-objective perspective in the near past, when only a small number of properties was under study, *e.g.* cost or availability. Having a few properties leads to an objective space where multi-objective evolutionary algorithms work well. However, in real-world environments these approaches have shown their unsuitability as the number of objectives increases, *e.g.* considering at the same time both runtime and design attributes. Even so, the trade-off among all of them has to be still preserved, and the choice of candidate services becomes a harder task demanding more sophisticated optimisation techniques.

A comparative study of 2 multi-objective and 6 many-objective evolutionary algorithms has been proposed to address a 9-objective (QoS properties) QoSWSC problem, taking into consideration those aspects that the engineer might find in a real environment. This is the first generalisable and extensive application of specific many-objective evolutionary algorithms to solve this optimisation problem, where factors like the number of tasks or the composition structure influence its complexity. Therefore, experiments have also considered a wide range of problem instances using real QoS values.

Experimental results confirm that many-objective algorithms are a suitable option to face QoSWSC problems considering a large number of objectives at design time. Among the implications that can be derived from the conducted analysis, it is worth mentioning the ability of many-objective algorithms like ϵ-MOEA, HypE and IBEA to optimise specific QoS properties. Additionally, the experimental study has shown that the proposed approach is not specifically influenced by the way how the problem is formulated in terms of its structure composition and tasks.

As many-objective optimisation has turned out to be an interesting paradigm to move one step forward in the automatic composition of web services, future research is planned to explore even more complex formulations of the QoSWSC problem. As a next step, adding constraints like service dependencies or the satisfaction of thresholds for certain QoS properties will allow analysing their influence on the search process. In this application domain, it is of particular relevance to study how constraint-handling techniques can be effectively integrated into many-objective evolutionary approaches. Similarly, combining many-objective algorithms with prioritisation techniques would allow focusing the search on those QoS properties of highest interest to the engineer.

In addition, authors plan to explore the possibility of combining the approaches proposed in this paper, aimed at addressing the QoSWSC problem at design time, with other techniques more appropriate to enable the optimisation process at runtime (Parejo et al., 2014). With such a combined approach, the entire life-cycle of the service compositions could be covered, including design-time service selection, optimisation at deployment-time, and run-time reconfiguration. Another important step forward is the application of this approach to a real case study using popular services like Amazon EC2 and PayPal. Finally, we consider relevant for the expert system to let the engineer get involved by the search algorithm using human-in-the-loop models, so that it could explore the search space guided by the experts decisions.

Acknowledgements

Work supported by the Spanish Ministry of Science and Technology and the Andalusian R&I&D, projects P12-TIC-1867, TIN2012-32273, TIC-5906, and FEDER funds. This research was also supported by Spanish Ministry of Economy and Competitiveness, projects TIN2014-55252-P and TIN2015-71841-REDT, and the Spanish Ministry of Education under the FPU program (FPU13/01466).

References

Al-Masri, E., & Mahmoud, Q. H. (2008). Investigating web services on the world wide web. In *Proceedings of the 17th International Conference on World Wide Web* WWW '08 (pp. 795–804). New York, NY, USA: ACM. URL: `http://dx.doi.org/10.1145/1367497.1367605`. doi:`10.1145/1367497.1367605`.

Arcuri, A., & Briand, L. (2011). A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the*

33rd International Conference on Software Engineering ICSE '11 (pp. 1–10). New York, NY, USA: ACM. URL: `http://dx.doi.org/10.1145/1985793.1985795`. doi:`10.1145/1985793.1985795`.

Ardagna, D., & Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering*, *33*, 369–384. URL: `http://dx.doi.org/10.1109/TSE.2007.1011`. doi:`10.1109/TSE.2007.1011`.

Bader, J., & Zitzler, E. (2011). HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, *19*, 45–76. URL: `http://dx.doi.org/10.1162/EVCO_a_00009`. doi:`10.1162/EVCO_a_00009`.

Bonatti, P. A., & Festa, P. (2005). On optimal service selection. In *Proceedings of the 14th International Conference on World Wide Web* WWW '05 (pp. 530–538). New York, NY, USA: ACM. URL: `http://dx.doi.org/10.1145/1060745.1060823`. doi:`10.1145/1060745.1060823`.

de Campos, A., Pozo, A., Vergilio, S., & Savegnago, T. (2010). Many-objective evolutionary algorithms in the composition of web services. In *Proceedings of the 11th Brazilian Symposium on Neural Networks* SBRN'10 (pp. 152–157). IEEE. URL: `http://dx.doi.org/10.1109/SBRN.2010.34`. doi:`10.1109/SBRN.2010.34`.

Canfora, G., Penta, M. D., Esposito, R., & Villani, M. L. (2005). An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* GECCO '05 (pp. 1069–1075). New York, NY, USA: ACM. URL: `http://dx.doi.org/10.1145/1068009.1068189`. doi:`10.1145/1068009.1068189`.

Canfora, G., Penta, M. D., Esposito, R., & Villani, M. L. (2008). A framework for QoS-aware binding and re-binding of composite web services. *Journal of Systems and Software*, *81*, 1754–1769. URL: `http://dx.doi.org/10.1016/j.jss.2007.12.792`. doi:`10.1016/j.jss.2007.12.792`.

Coello Coello, C. A., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. (2nd ed.). Secaucus, NJ, USA: Springer-Verlag New York, Inc. URL: `http://dx.doi.org/10.1007/978-0-387-36797-2`. doi:`10.1007/978-0-387-36797-2`.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc.

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, *18*, 577–601. URL: `http://dx.doi.org/10.1109/TEVC.2013.2281535`. doi:`10.1109/TEVC.2013.2281535`.

Deb, K., Mohan, M., & Mishra, S. (2003). Towards a quick computation of well-spread pareto-optimal solutions. In C. s. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, & K. Deb (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 222–236). Springer Berlin Heidelberg volume 2632 of *Lecture Notes in Computer Science*. URL: `http://dx.doi.org/10.1007/3-540-36970-8_16`. doi:`10.1007/3-540-36970-8_16`.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197. URL: `http://dx.doi.org/10.1109/4235.996017`. doi:`10.1109/4235.996017`.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*, 3–18. URL: `http://dx.doi.org/10.1016/j.swevo.2011.02.002`. doi:`10.1016/j.swevo.2011.02.002`.

García-Galán, J., Rana, O., Trinidad, P., & Cortés, A. R. (2013). Migrating to the cloud - a software product line based analysis. In *Proceedings* *of the 3rd International Conference on Cloud Computing and Services Science* CLOSER 2013 (pp. 416–426). URL: http://dx.doi.org/10.5220/0004357104160426. doi:10.5220/0004357104160426.

Ishibuchi, H., Tsukamoto, N., & Nojima, Y. (2008). Evolutionary many-objective optimization: a short review. In *Proceedings of the IEEE Congress on Evolutionary Computation* CEC 2008 (pp. 2419–2426). URL: http://dx.doi.org/10.1109/CEC.2008.4631121. doi:10.1109/CEC.2008.4631121.

Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: a systematic literature review. *Expert Systems with Applications*, *41*, 3809–3824. URL: http://dx.doi.org/10.1016/j.eswa.2013.12.017. doi:10.1016/j.eswa.2013.12.017.

Khare, V., Yao, X., & Deb, K. (2003). Performance scaling of multi-objective evolutionary algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, & K. Deb (Eds.), *Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization* (pp. 376–390). Berlin, Heidelberg: Springer volume 2632 of *Lecture Notes in Computer Science*. URL: http://dx.doi.org/10.1007/3-540-36970-8_27. doi:10.1007/3-540-36970-8_27.

Li, W., & Yan-xiang, H. (2010). A web service composition algorithm based on global QoS optimizing with MOCACO. In C.-H. Hsu, L. T. Yang, J. H. Park, & S.-S. Yeo (Eds.), *Algorithms and architectures for parallel processing* (pp. 218–224). Springer Berlin Heidelberg volume 6082 of *Lecture Notes in Computer Science*. URL: http://dx.doi.org/10.1007/978-3-642-13136-3_22. doi:10.1007/978-3-642-13136-3_22.

von Lücken, C., Barán, B., & Brizuela, C. (2014). A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, *58*, 707–756. URL: http://dx.doi.org/10.1007/s10589-014-9644-1. doi:10.1007/s10589-014-9644-1.

Moustafa, A., & Zhang, M. (2013). Multi-objective service composition using reinforcement learning. In S. Basu, C. Pautasso, L. Zhang, & X. Fu (Eds.), *Service-Oriented Computing* (pp. 298–312). Springer Berlin Heidelberg volume 8274 of *Lecture Notes in Computer Science*. URL: `http://dx.doi.org/10.1007/978-3-642-45005-1_21`. doi:`10.1007/978-3-642-45005-1_21`.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: state of the art and research challenges. *IEEE Computer*, *40*, 38–45. URL: `http://dx.doi.org/10.1109/MC.2007.400`. doi:`10.1109/MC.2007.400`.

Parejo, J. A., Segura, S., Fernández, P., & Ruiz-Cortés, A. (2014). QoS-aware web services composition using GRASP with Path Relinking. *Expert Systems with Applications*, *41*, 4211–4223. URL: `http://dx.doi.org/10.1016/j.eswa.2013.12.036`. doi:`10.1016/j.eswa.2013.12.036`.

Praditwong, K., & Yao, X. (2007). How well do multi-objective evolutionary algorithms scale to large problems. In *Proceedings of the IEEE Congress on Evolutionary Computation* CEC 2007 (pp. 3959–3966). IEEE. URL: `http://dx.doi.org/10.1109/CEC.2007.4424987`. doi:`10.1109/CEC.2007.4424987`.

Purshouse, R., & Fleming, P. (2007). On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, *11*, 770–784. URL: `http://dx.doi.org/10.1109/TEVC.2007.910138`. doi:`10.1109/TEVC.2007.910138`.

Ramírez, A., Romero, J. R., & Ventura, S. (2015). An extensible JCLEC-based solution for the implementation of multi-objective evolutionary algorithms. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* GECCO Companion '15 (pp. 1085–1092). New York, NY, USA: ACM. URL: `http://dx.doi.org/10.1145/2739482.2768461`. doi:`10.1145/2739482.2768461`.

Romano, J., Kromrey, J. D., Coraggio, J., & Showronek, J. (2006). Appropriate statistics for ordinal level data: Should we really be using t-test and cohen's d for evaluating group differences on the NSSE and other surveys? In *Annual Meeting of the Florida Association of Institutional Research* (pp. 1–33).

Strunk, A. (2010). QoS-aware service composition: a survey. In *Proceedings of the 2010 IEEE 8th European Conference on Web Services* ECOWS (pp. 67–74). IEEE. URL: `http://dx.doi.org/10.1109/ECOWS.2010.16`. doi:`10.1109/ECOWS.2010.16`.

Suciu, M., Pallez, D., Cremene, M., & Dumitrescu, D. (2013). Adaptive MOEA/D for QoS-based web service composition. In M. Middendorf, & C. Blum (Eds.), *Evolutionary Computation in Combinatorial Optimization* (pp. 73–84). Springer Berlin Heidelberg volume 7832 of *Lecture Notes in Computer Science*. URL: `http://dx.doi.org/10.1007/978-3-642-37198-1_7`. doi:`10.1007/978-3-642-37198-1_7`.

Trummer, I., Faltings, B., & Binder, W. (2014). Multi-objective quality-driven service selection - a fully polynomial time approximation scheme. *IEEE Transactions on Software Engineering*, *40*, 167–191. URL: `http://dx.doi.org/10.1109/TSE.2013.61`. doi:`10.1109/TSE.2013.61`.

Ventura, S., Romero, C., Zafra, A., Delgado, J. A., & Hervás, C. (2007). JCLEC: a Java framework for evolutionary computation. *Soft Computing*, *12*, 381–392. URL: `http://dx.doi.org/10.1007/s00500-007-0172-0`. doi:`10.1007/s00500-007-0172-0`.

Wada, H., Suzuki, J., Yamano, Y., & Oba, K. (2012). E3: a multiobjective optimization framework for SLA-aware service composition. *IEEE Transactions on Services Computing*, *5*, 358–372. URL: `http://dx.doi.org/10.1109/TSC.2011.6`. doi:`10.1109/TSC.2011.6`.

Wagner, T., Beume, N., & Naujoks, B. (2007). Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, & T. Murata (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 742–756). Springer Berlin Heidelberg volume

4403 of *Lecture Notes in Computer Science*. URL: http://dx.doi.org/10.1007/978-3-540-70928-2_56. doi:10.1007/978-3-540-70928-2_56.

Wang, H., Tong, P., & Thompson, P. (2007). QoS-based web services selection. In *Proceedings of the IEEE International Conference on e-Business Engineering* ICEBE 2007 (pp. 631–637). URL: http://dx.doi.org/10.1109/ICEBE.2007.109. doi:10.1109/ICEBE.2007.109.

Yang, S., Li, M., Liu, X., & Zheng, J. (2013). A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, *17*, 721–736. URL: http://dx.doi.org/10.1109/TEVC.2012.2227145. doi:10.1109/TEVC.2012.2227145.

Yin, H., Zhang, C., Zhang, B., Guo, Y., & Liu, T. (2014). A hybrid multiobjective discrete particle swarm optimization algorithm for a SLA-aware service composition problem. *Mathematical Problems in Engineering*, *2014*, 1–14. URL: http://dx.doi.org/10.1155/2014/252934. doi:10.1155/2014/252934.

Yu, Y., Ma, H., & Zhang, M. (2015). F-MOGP: a novel many-objective evolutionary approach to QoS-aware data intensive web service composition. In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation* CEC (pp. 2843–2850). IEEE. URL: http://dx.doi.org/10.1109/CEC.2015.7257242. doi:10.1109/CEC.2015.7257242.

Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnam, J., & Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, *30*, 311–327. URL: http://dx.doi.org/10.1109/TSE.2004.11. doi:10.1109/TSE.2004.11.

Zhang, Q., & Li, H. (2007). MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*, 712–731. URL: http://dx.doi.org/10.1109/TEVC.2007.892759. doi:10.1109/TEVC.2007.892759.

Zhang, T. (2014). QoS-aware web service selection based on particle swarm optimization. *Journal of Networks*, *9*, 565–570. URL: `http://dx.doi.org/10.4304/jnw.9.3.565-570`. doi:`10.4304/jnw.9.3.565-570`.

Zhao, X., Song, B., Huang, P., Wen, Z., Weng, J., & Fan, Y. (2012). An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition. *Applied Soft Computing*, *12*, 2208–2216. URL: `http://dx.doi.org/10.1016/j.asoc.2012.03.040`. doi:`10.1016/j.asoc.2012.03.040`.

Zitzler, E., & Künzli, S. (2004). Indicator-based selection in multiobjective search. In X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, & H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature - PPSN VIII* (pp. 832–842). Springer Berlin Heidelberg volume 3242 of *Lecture Notes in Computer Science*. URL: `http://dx.doi.org/10.1007/978-3-540-30217-9_84`. doi:`10.1007/978-3-540-30217-9_84`.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: improving the strength pareto evolutionary algorithm. In *Proceedings of the Conference on Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems* (pp. 95–100).