Adaptive MOEA/D for QoS-Based Web Service Composition

Mihai Suciu[1], Denis Pallez[2], Marcel Cremene[3], and Dumitru Dumitrescu[1]

[1] Babes-Bolyai University, Cluj Napoca, Romania
[2] University of Nice Sophia-Antipolis
[3] Technical University of Cluj-Napoca, Romania
mihai.suciu@ubbcluj.ro

Abstract. QoS aware service composition is one of the main research problem related to *Service Oriented Computing (SOC)*. A certain functionality may be offered by several services having different Quality of Service (QoS) attributes. Although the QoS optimization problem is multiobjective by its nature, most approaches are based on single-objective optimization. Compared to single-objective algorithms, multiobjective evolutionary algorithms have the main advantage that the user has the possibility to select *a posteriori* one of the Pareto optimal solutions. A major challenge that arises is the dynamic nature of the problem of composing web services. The algorithms performance is highly influenced by the parameter settings. Manual tuning of these parameters is not feasible. An evolutionary multiobjective algorithm based on decomposition for solving this problem is proposed. To address the dynamic nature of this problem we consider the hybridization between an adaptive heuristics and the multiobjective algorithm. The proposed approach outperforms state of the art algorithms.

1 Introduction

The composition of web services with optimal Quality of Service (QoS) parameters is a well known problem in the service oriented computing field. Given a business workflow that includes a set of abstract services and a set of concrete service, that implements each abstract service, the goal is to find the optimal combination of concrete services based on their QoS parameters. Given m abstract services and n concrete services for each abstract service, there are n^m possibilities.

The search space is a discrete one since for each abstract service we need to chose one concrete service (any combination is possible). This is a combinatorial optimization problem. Finding the solution with the optimal QoS is an NP-hard problem.

The problem stated previously is well known in domains like Service Oriented Computing (SOC) and Search-based Software Engineering (SBSE) [3], [6], [17], [25]. Various solutions are proposed based on different approaches such as: integer programming, genetic and hill climbing algorithms [1], [3], [29], [19].

M. Middendorf and C. Blum (Eds.): EvoCOP 2013, LNCS 7832, pp. 73–84, 2013.

Despite the fact that the QoS optimization problem is multiobjective by nature few approaches based on multiobjective algorithms can be found in the literature [15], [22], [24], [26], [28]. In most cases single-objective algorithms are used to solve this problem. The user might prefer to see several good solutions (Pareto optimal) and decide which is the best for himself. Criteria aggregation offers only one solution. It is more natural to let the user decide the importance of each objective than aggregating the objectives and ask the user to specify *a priori* his/her preferences (this is not a trivial task). By using multiobjective optimization, it is not necessary to define *a priori* an aggregation function. For solving the QoS web composition problem few applications based on multiobjective optimization algorithms can be found in the literature

There are many variants of EAs which have different control parameters: population size, operators used, crossover and mutation probabilities, etc. Selecting appropriate values is mainly done based on empirical studies, often a "trial and error" fashion is used for adjusting the values. Typically one parameter is adjusted at a time, which may lead to sub-optimal choices, since often it is not known how the parameters interact. Such an approach is time consuming. In the last couple of years there has been an increasing interest in designing methods that self-adapt these parameters [4], [9], [18].

The hybridization between a decomposition based multiobjective optimization algorithm and an adaptive technique is proposed. An algorithm based on a decomposition technique [2], [13] seems appropriate for solving this problem. The new approach is validated on some well known multiobjective test problems, and then we apply it to the web service composition problem. We compare our results with some state of the art algorithms from the literature.

2 Multiobjective Optimization Prerequisites

Let us consider m 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 *Multiobjective Optimization Problem (MOP)* can be defined as: find $x^* = (x_1, ..., x_n)$ which optimizes the vector function $F(x)$ and satisfies the defined constraints.

In the case of minimization the standard MOP may be written as:

$$MOP : \begin{cases} \text{minimize } F(x) = (f_1(x), f_2(x), ..., f_m(x)) \\ \text{subject to: } g_i(x) \geq 0, \ i{=}1,2,...,k, \\ \qquad\qquad h_j(x) = 0, j{=}k{+}1,...,q. \end{cases} \tag{1}$$

where $x = (x_1, ..., x_n) \in X$.

x is a *decision vector*. X represents the *parameter space* and Y is the *objective space*.

A decision vector $x \in X$ is said to *Pareto-dominate* $y \in X$ (x is better than y), denoted as $x \prec y$, if and only if $\forall i \in 1, ..., m, f_i(x) \leq f_i(y)$, and $\exists j \in 1, ..., n$ *such that* $f_j(x) < f_j(y)$.

A solution $x \in X$ is *Pareto-optimal* if and only if $\nexists y \in X$ such that $y \prec x$.

The *Pareto-Optimal Set* (POS) is defined as the set of all Pareto-optimal solutions $POS = \{x \in X | \nexists y \in X, y \prec x\}$.

The *Pareto-Optimal Front* (POF) is defined as the set of all objective values corresponding to the solution in POS. $POF = \{F(x) | x \, is \, nondominated\}$.

The QoS-based service optimization is a combinatorial multiobjective optimization problem. Using a decomposition technique many-criteria problems may be decomposed in a class of single-objective problems [2], [13]. Decomposition techniques have good performance for combinatorial problems, another advantage is the small computational load. One drawback to decomposition evolutionary approach is the dependency between the problem type and the algorithm parameters. The same algorithm must solve different instances of this problem. One instance is represented by the business workflow that describes the web services composition (the interconnections of the composing web services, see [20] for more details). Different web services are described by different workflows, thus the search space changes for each workflow making it a very dynamical problem. A set of parameters that work for one particular instance of the workflow may not yield good results for another workflow. It would be very difficult to tune parameters for each particular workflow. A self-adaptive technique seems the obvious choice for this kind of problem.

2.1 MOEA/D Technique

MOEA/D [30] is a multiobjective algorithm based on decomposition. It is a simple and powerful scalarizing based algorithm. MOEA/Ds advantages are scalability with the number of objectives, computational efficiency, high performance for combinatorial optimization problems.

It uses the weighted Tchebycheff approach in order to decompose the MOP in a number of single-objective problems, each problem is represented by an individual in the current population. The Tchebycheff norm is:

$$g_\lambda(x, z) = \max_{i=1,\ldots,m} \{\lambda_i | f_i(x) - z_i|\}, \forall x \in X$$

where z is an optimal point, the goal is to minimize g_λ.

Each sub-problem is characterized by different weight vectors. The number of uniformly distributed weight vectors used is equal to the number of subproblems that are to be optimized. The number of weights can be computed as $N = \binom{H+m-1}{m-1}$ where H is a predefined integer. These vectors satisfy the conditions:

$$\sum \lambda_i = 1 \, and \, \lambda_i \in \{0, \frac{1}{H}, ..., \frac{H}{H}\}, i = 1, ..., m.$$

Another key feature of MOEA/D is the use of neighboring solutions for generating offsprings more efficiently. Based on Euclidean distance for each λ weight vector T neighbors are computed. The set of T neighbors of λ is denoted by $B(\lambda)$. From this set the parents are selected. For each weight vector λ one offspring is generated by crossover and mutation. If the offspring is better than its parents it replaces them in the current population. A key step here is that the offspring

is also compared with each neighbor in $B(\lambda)$ based on its decomposition, if it is better it replaces that neighbor. This is an elitism step, it assures that only the best solutions propagate through the search.

An archive is used to store non-dominated solutions. The archive has no effect on the search. It is used only for storing the best individuals found during each generation. If a newly generated offspring is better than an individual inside the archive it simply replaces it.

Some adaptive techniques for MOEA/D parameters have been proposed in [5], [11], [12],[16], [31]. These techniques try to self adapt the algorithms parameters: weight vectors, neighbour size T, recombined individuals.

As a scalarizing function MOEA/D uses the Tchebycheff approach. In [11] the possibility of using different decomposition techniques, according to the complexity of the problem, is explored.

In [5] an adaptive mechanism for selective mating is proposed. The decomposed subproblems are classified in solved and unsolved, a subproblem is considered as solved if it is not improved in α generation. In the current generation if a subproblem is solved it is skipped, the unsolved ones are recombined and evaluated. Also the mating pool is adjusted. The mating candidates are selected according to Euclidean distance in decision space.

An adaptive scheme for weight vectors generation is proposed in [12].Here the weight vectors are adapted according to the geometrical characteristics of the Pareto front. Uniform distributed vectors are generated using Mixture Uniform Design. The hypervolume is used to evaluate these vectors and the Simplex Method is used to adapt them in order to maximize the hypervolume.

In [31] an adaptation scheme for the neighborhood size is proposed, it is shown that the adaptive version gives better results than the classic MOEA/D with the neighborhood size fixed.

All approaches do not consider the adaptation of the evolutionary mechanism used. So in this paper we address the adaptation of the parameters of the evolutionary mechanism of MOEA/D and we apply this new approach to web services composition.

2.2 Adaptation of Differential Evolution

Differential Evolution [23] is a continuous function evolutionary algorithm. There are many adaptive versions of Differential Evolution [7]. From all the techniques CoDE [27] and SaDE [21] seem the most simple and efficient.

Self adaptive Differential Evolution (*SaDE*) [21] is an adaptation technique that uses the experience from previous generations. It adapts the trial vector generation strategies and the parameters C_r and F. It assigns a probability to each generation strategies, after a predefined number of generations these probabilities are update by taking into account the best strategies (strategies that generate individuals that pass to the next generation are assigned a higher probability). Normally the algorithm uses four different trial vector generation strategies: *DE/rand/1/bin*, *DE/rand-to-best/2/bin*, *DE/rand/2/bin*, and

DE/current-to-rand/1. For the sake of simplicity for the multiobjective version we use only: *DE/rand/1/bin* and *DE/best/2/bin*.

Each strategy is initialized with a probability $p_i = 0.5$ and after a learning period LP this probability is updated according to [21]. *SaDE* also adapts Cr and F parameters. Cr is more sensible to problem type and complexity while F parameter is tied to convergence speed. The algorithm performance depends on Cr values and usually good values lie in a small interval.Cr values are random generated according to a distribution with mean values Cr_m and deviation *0.1*, initially $Cr_m = 0.5$. After 5 generations new Cr values are randomly generated using the distribution with $\mu = Cr_m$ and $\sigma = 0.1$. For 25 generations this process is repeated keeping Cr values for which trial vectors are selected in favor of parents to advance to the next generation. After these 25 generation the new mean for the distribution is computed using the successful Cr values discovered. F takes random values in the interval $(0, 2]$ with $\mu = 0.5$ and $\sigma = 0.3$.

Within most DE variants one strategy for generating the trial vector is used per generation and only one control parameter setting for each trial vector. For this reason the search ability of the algorithm could be limited. An improved version of adaptive DE is proposed in [27]. By combining several effective trial vector generation strategies with some suitable control parameter settings better results can be obtained. This new method is called Composite DE (*CoDE*) and it uses three trial vector generation strategies and three control parameter settings that are randomly combined to generate the trial vector.

Three trial vector strategies are chosen: *DE/rand/1/bin*, *DE/rand/2/bin*, and *DE/current-to-rand/1*. The three control parameter settings proposed: *(a)* $[F = 1.0, Cr = 0.1]$, *(b)* $[F = 1.0, Cr = 0.9]$, *(c)* $[F = 0.8, Cr = 0.2]$.

In each generation for each target vector three trial vectors are generated using the above strategies and a random control parameter setting chosen from the candidate pool. The best one is selected for the next generation.

The values in the parameter settings pool are chosen because they exhibit some specific advantages: a large value for Cr encourages diversity because little information is inherited from the target vector. For small Cr values the trial vector differs from the target vector only by one gene thus optimizing each parameter independently. For this case better results are obtained for separable problems. Large F values increase diversity thus promoting exploration and low values focus the search on neighborhoods increasing exploitation.

3 Proposed Approach

As the Qos-based web service optimization problem is a combinatorial one we use $MOEA/D$ algorithm to solve it. To cope with the dynamic nature of the QoS web service composition problem we endow $MOEA/D$ with an adaptation mechanism. $MOEA/D$ is based on DE. Some very simple and yet powerful adaptation techniques for DE have been propose [18]. We propose two adaptive variants of $MOEA/D$ obtained by considering the DE adaptive mechanisms *SaDE* [21] and *CoDE* [27]. The new models are called $MOEA/D_C$ (Algorithm 1) and $MOEA/D_S$ (Algorithm 2).

Algorithm 1. Adaptive $MOEA/D_C$

input : *N, T* - number of sub-problems, neighborhood size
output: *EP* - external population that holds the non-dominate solutions
1 Initialization: $EP = \varnothing$, generate weight vectors and compute $B(\lambda)$;
2 **for** $i \leftarrow 1$ **to** N **do**
3 generate 3 offsprings using a random combination between a trial vector generation strategy and the control parameters;
4 update the neighboring solutions;
5 update z^* and EP;
6 If stopping criteria is satisfied output the EP. Otherwise go to step 2;

Algorithm 2. Adaptive $MOEA/D_S$

input : *N, T, LP* - number of sub-problems, neighborhood size, learning period
output: *EP* - external population that holds the non-dominate solutions
1 Initialization: $EP = \varnothing$, generate weight vectors and compute $B(\lambda)$, $Cr_m = 0.5$;
2 **for** $i \leftarrow 1$ **to** N **do**
3 generate 2 offsprings based on the strategies *rand/1/bin* and *best/2/bin*;
4 update the neighboring solutions;
5 update z^* and EP;
6 after LP generations update Cr_m and the probabilities p_i for the trial vector generation strategies;
7 If stopping criteria is satisfied output the EP. Otherwise go to step 2;

In $MOEA/D_C$ the DE trial vector generation strategy $DE/rand/1/bin$ used in $MOEA/D$ is replaced with the $CoDE$ strategy - three trial vectors are created and the best one is kept (lines 3-4). In $MOEA/D_S$ the DE trial vector generation strategy is replaced with the $SaDE$ strategy and after LP generations the parameter Cr_m and the probabilities for using each trial vector generation strategy from the $SaDE$ candidate pool are computed (line 6). By using an adaptive scheme we avoid the drawback of manual tuning the algorithm parameters for each specific workflow.

The genome we use for our problem is depicted in figure 1. It consists of an array of integer values and has the length equal to the number of abstract services. Each gene stores the index of the concrete service that realizes the corresponding abstract service.

4 Numerical Experiments

For validating the new adaptive approaches we compare the classic version of $MOEA/D$ with $MOEA/D_C$ and $MOEA/D_S$. As a basis for comparison the *WFG* test suite [10] is considered. All *WFG* problems considered are bi-objective with $k = 4$ and $n = 24$. For all algorithms we use a population size $N = 100$, a neighborhood $T = 8$ and a maximum number of generations $g = 250$. For the

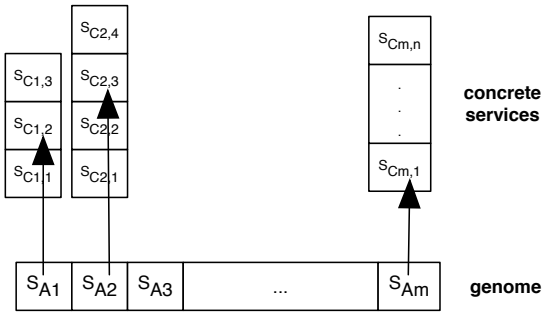

Fig. 1. Genome encoding

classic version of $MOEA/D$: $Cr = 0.7$, $F = 0.2$, and for $MOEA/D_S$ we use a learning period $LP = 25$. The best results are obtained with these parameter values.

We run each algorithm for 50 times and then we compute on the final population the mean values for the quality indicators: Inverted Generational Distance (IGD) and Hypervolume (HV). The IGD computes the average distance of the reference Pareto set (P^*) to the nearest solution in the solution set found (A). IGD indicates the spread of A, small values are desirable.

$$IGD(A, P^*) = \frac{1}{|P|} \sum_{u \in P} min_{v \in A} d(u, v)$$

The hypervolume represents the surface covered by the solution set and a reference point, high values mean that the front is near to the theoretical front and it assures a good spread. The reference point for the HV indicator is $(7, 7)$.

These values are presented in tables 1 and 2. The standard deviation for all experiments is less that 10^{-4}. From this tables we can see that $MOEA/D_C$ performs better for all WFG problems, it gives higher HV values. The IGD is the same for $MOEA/D$ and $MOEA/D_C$ for six of the eight problems and better that $MOEA/D_S$. The classic version of the algorithm outperforms the adaptive counterparts only for the WFG1 problem.

The advantages of the adaptive approach is illustrated in Figures 2, 3 for a stopping criterion of $g_{max} = 400$ generations. Higher solution diversity is assured because the DE parameters for crossover and mutation are chosen to balance the search and exploitation. The Pareto front found by the adaptive version is closer to the theoretical front. From our experiments we observe also an advantage in convergence speed, at the same generation the front found by the adaptive approach is closer to the real front.

After validating our approach we apply it to the QoS-based web service composition. We compute the hyprvolume indicator for various test scenarios.

Because $MOEA/D_C$ gives better results that the classic version of MOEA/D and $MOEA/D_S$ we apply it to the QoS-based web service composition problem.

Table 1. IGD adn HV for $MOEA/D$, $MOEA/D_C$, and $MOEA/D_S$ alforithms (average values over 50 independent runs). WFG 1-5 test problems are considered.

WFG	1		2		3		4		5	
Alg.	IGD	HV	IGD	HV	IGD	HV	IGD	HV	IGD	HV
MOEA/D	0.002	31.126	0.002	42.358	1.645	43.766	0.006	40.12	0.009	40.213
$MOEA/D_C$	0.003	**34.057**	0.002	**43.266**	1.597	**43.968**	0.006	**41.827**	0.009	**41.486**
$MOEA/D_S$	0.005	29.144	0.003	34.84	6.236	36.457	0.006	36.55	0.009	39.62

Table 2. IGD and HV values for $MOEA/D$, $MOEA/D_C$, and $MOEA/D_S$ alforithms (average values over 50 independent runs). WFG 6-8 test problems are considered.

WFG	6		7		8	
Alg.	IGD	HV	IGD	HV	IGD	HV
MOEA/D	0.009	41.358	0.01	42.091	0.004	40.652
$MOEA/D_C$	0.009	**41.486**	0.01	**42.242**	0.004	**40.782**
$MOEA/D_S$	0.009	39.355	0.009	34.71	0.004	33.162

We conducted experiments for 12 scenarios that include all combinations of $m \in \{10, 20, 30, 40\}$ abstract services and $n \in \{20, 30, 50\}$ concrete services. Three QoS parameters are considered: access time to a web service and cost (these two objectives need to be minimized), and the third objective is user rating (that needs to be maximized). We compare $MOEA/D_C$ with NSGA2 [8], and GDE3 [14] algorithms. For all algorithms the population was limited to 100 individuals which were evolved for 400 generations.

Figure 4 presents a case with a business workflow of $m = 20$ abstract services each of them having $n = 20$ concrete alternative services. For a low complexity workflow the GDE3 algorithm finds a more diverse final non-dominated set.

If we increase the complexity of the workflow to $m = 40$ abstract services and $n = 40$ concrete alternative services for each abstract service, the adaptive version is able to assure good solution diversity and and find non-dominated solutions compared to the other algorithms. Figure 5 depicts these results.

Table 3 presents the Hypervolume indicator values for some scenarios. It can be observed that for low complexity business workflows the classic non-adaptive algorithms produce better results. For $m \in \{10, 20\}$ GDE3 is better. When we increase the complexity of the problem, $m \in \{30, 40\}$, the search space is considerably bigger, the adaptive version assures better performance with respect to the hypervolume indicator. The hypervolumes sometimes increase although the problems complexity is higher because the larger search space yields solutions that are unavailable to smaller search spaces.

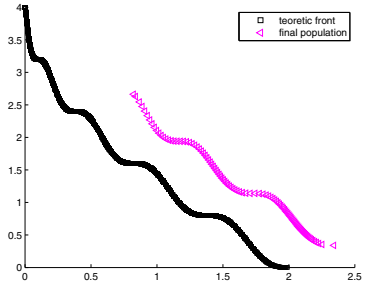

Fig. 2. $MOEA/D$ final population and theoretical front for WFG1 test problem

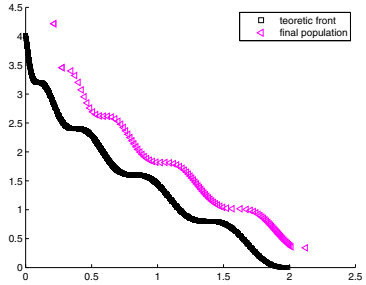

Fig. 3. $MOEA/D_C$ final population and theoretical front for WFG1 test problem

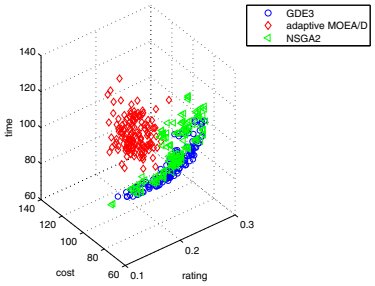

Fig. 4. Final populations for GDE3, NSGA2, and $MOEA/D_C$ algorithms for a business workflow with $m = 20$ abstract services and $n = 20$ concrete services

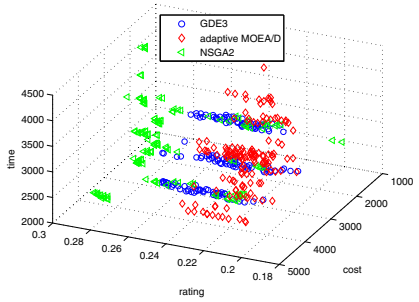

Fig. 5. Final populations for GDE3, NSGA2, and $MOEA/D_C$ algorithms for a business workflow with $m = 40$ abstract services and $n = 40$ concrete services

Table 3. Hypervolume indicator values for different workflow complexities $m \in \{10, 20, 30, 40\}$ and $n \in \{20, 50\}$ (mean values for 30 independent runs)

Algorithm	n/m							
	10/20	10/50	20/20	20/50	30/20	30/50	40/20	40/50
GDE3	**652.84**	**480.12**	**450.92**	**465.2**	465.4	440.3	410.3	411.5
NSGA2	649.49	475.28	444.12	461.7	461.8	424.2	402.6	390.9
$MOEA/D_C$	620.38	468.14	439.1	455.2	**470.3**	**480.6**	**468.1**	**465.7**

5 Conclusions

An adaptive approach for the NP-hard problem of composing web services based on their Quality of Service properties is proposed. An evolutionary multiobjective optimization approach for QoS problem is considered. A new adaptive version of MOEA/D is proposed addressing the considered combinatorial problem.

We compare the proposed algorithm with the classic version of MOEA/D then apply it to the QoS-based service composition problem and some state of the art multiobjective algorithms are considered for comparison.

The results show the potential of this approach. Better performance is obtained (with respect to multiobjective quality indicators) when the adaptive approach is applied to standard test problems and some business workflows of high complexity.

Acknowledgements. This research was supported by the national project code TE 252 financed by the Romanian Ministry of Education and Research CNCSIS-UEFISCSU. The first would like to thank for the financial support provided from program co-financed by the Sectoral Operational Programme Human Resources Development, Contract POSDRU/107/1.5/S/76841 with the title Modern Doctoral Studies: Internationalization and Interdisciplinarity.

References

1. Bahadori, S., Kafi, S., Far, K.Z., Khayyambashi, M.R.: Optimal web service composition using hybrid ga-tabu search. Journal of Theoretical and Applied Information Technology 9(1) (2009)
2. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4(1), 238–252 (1962)
3. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075 (2005)
4. Chakhlevitch, K., Cowling, P.: Hyperheuristics: Recent Developments. In: Cotta, C., Sevaux, M., Sörensen, K. (eds.) Adaptive and Multilevel Metaheuristics. SCI, vol. 136, pp. 3–29. Springer, Heidelberg (2008)
5. Chiang, T.-C., Lai, Y.-P.: Moea/d-ams: Improving moea/d by an adaptive mating selection mechanism. In: IEEE Congress on Evolutionary Computation, CEC 2011, pp. 1473–1480. IEEE (2011)
6. Comes, D., Baraki, H., Reichle, R., Zapf, M., Geihs, K.: Heuristic Approaches for QoS-Based Service Selection. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 441–455. Springer, Heidelberg (2010)
7. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Trans. Evolutionary Computation 15(1), 4–31 (2011)
8. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolutionary Computation 6(2), 182–197 (2002)

9. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
10. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
11. Ishibuchi, H., Sakane, Y., Tsukamoto, N., Nojima, Y.: Adaptation of Scalarizing Functions in MOEA/D: An Adaptive Scalarizing Function-Based Multiobjective Evolutionary Algorithm. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 438–452. Springer, Heidelberg (2009)
12. Jiang, S., Cai, Z., Zhang, J., Ong, Y.-S.: Multiobjective optimization by decomposition with pareto-adaptive weight vectors. In: ICNC, pp. 1260–1264 (2011)
13. Kathrin, K., Tind, J.: Constrained optimization using multiple objective programming. Journal of Global Optimization 37, 325–355 (2007)
14. Kukkonen, S., Lampinen, J.: GDE3: the third evolution step of generalized differential evolution. In: Congress on Evolutionary Computation, pp. 443–450 (2005)
15. Li, L., Cheng, P., Ou, L., Zhang, Z.: Applying Multi-objective Evolutionary Algorithms to QoS-Aware Web Service Composition. In: Cao, L., Zhong, J., Feng, Y. (eds.) ADMA 2010, Part II. LNCS, vol. 6441, pp. 270–281. Springer, Heidelberg (2010)
16. Liu, B., Fernández, F.V., Zhang, Q., Pak, M., Sipahi, S., Gielen, G.G.E.: An enhanced MOEA/D-DE and its application to multiobjective analog cell sizing. In: IEEE Congress on Evolutionary Computation, pp. 1–7 (2010)
17. Liu, X., Xu, Z., Yang, L.: Independent global constraints-aware web service composition optimization based on genetic algorithm. In: IASTED International Conference on Intelligent Information Systems, pp. 52–55 (2009)
18. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. 33(1-2), 61–106 (2010)
19. Parejo, J.A., Fernandez, P., Cortes, A.R.: Qos-aware services composition using tabu search and hybrid genetic algorithms. Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos 2(1), 55–66 (2008)
20. Pop, F.-C., Pallez, D., Cremene, M., Tettamanzi, A., Suciu, M.A., Vaida, M.-F.: Qos-based service optimization using differential evolution. In: GECCO, pp. 1891–1898 (2011)
21. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13(2), 398–417 (2009)
22. Ross, S.M.: Introduction to Probability Models, 9th edn. Academic Press, Inc., Orlando (2006)
23. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
24. Taboada, H.A., Espiritu, J.F., Coit, D.W.: MOMS-GA: A Multi-Objective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems. IEEE Transactions on Reliability 57(1), 182–191 (2008)
25. Vanrompay, Y., Rigole, P., Berbers, Y.: Genetic algorithm-based optimization of service composition and deployment. In: Proceedings of the 3rd International Workshop on Services Integration in Pervasive Environments, SIPE 2008, pp. 13–18 (2008)

26. Wada, H., Champrasert, P., Suzuki, J., Oba, K.: Multiobjective Optimization of SLA-Aware Service Composition. In: Proceedings of the 2008 IEEE Congress on Services - Part I, SERVICES 2008, pp. 368–375. IEEE Computer Society (2008)
27. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evolutionary Computation 15(1), 55–66 (2011)
28. Yao, Y., Chen, H.: QoS-aware service composition using NSGA-II. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS 2009, pp. 358–363 (2009)
29. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30, 311–327 (2004)
30. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation 11, 712–731 (2007)
31. Zhao, S.-Z., Suganthan, P.N., Zhang, Q.: Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes. IEEE Trans. Evolutionary Computation 16(3), 442–446 (2012)