Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 2, No. 1, 2008

QoS-Aware Services composition using Tabu Search and Hybrid Genetic Algorithms

Jose Antonio Parejo
Pablo Fernandez
Antonio Ruiz Cortés

Dept. Computer Languages and Systems.
University of Sevilla

Abstract. In a distributed services oriented environment, having a myriad of functionally equivalent services, Quality of Service(QoS) emerges as the key differential factor. In this scenario organizations can dynamically select partners for their core business processes expressed as Composite Web Services (CWS). As a consequence, QoS-aware composition should drive an effective selection by optimizing different factors and meeting constraints according to preferences of organizations. QoS-aware composition can be formulated as a NP-hard optimization problem. In order to deal with this hard problem, different heuristic techniques (such as genetic algorithms with different solution encodings or simulated annealing) had been proposed in the literature. In this paper we apply metaheuristic optimization techniques to this problem, specifically tabu search and an hybrid genetic algorithm. We compare these techniques with other proposals using experimental results, showing that our proposals provide improvements.

1 Introduction

In a distributed environment of services, organizations can dynamically select partners for their core business processes. Those business processes are implemented as Composite Web Services (CWS), typically expressed using BPEL. In this context, QoS properties (such as cost, performance and reliability) are key factors for selecting and composing services. QoS-aware selection of concrete services to support the composition can be formulated as an optimization problem [2] [4].
In this paper we afford the QoS-aware service selection problem using a framework solution that boosts the adoption of different metaheuristic techniques using a common runtime. Main contributions of the paper are shown below:

1. We pioneer the application of some heuristic techniques applied to this novel problem, such as tabu search (TS) and a hybrid variant of genetic algorithm (HGA).
2. We present the results of a empirical study of the performance of these techniques compared with other heuristics, such as a basic genetic algorithm (GA), and iterative steepest descent method (ISD). The results shown:

- Our implementation of the hybrid genetic algorithm performs better than the rest of techniques on large instances of the problem -providing better solutions for nearly all runs-.
- Our implementation of tabu search performs better than GA and ISD, but show a less stable behavior than HGA and only performs better on small-medium instances of the problem -between 10 and 30 tasks and less than 300 candidate services- with short computation times.

The rest of the paper is organized as follows: Section 2 gives a formal definition of the Composite Web Service Selection problem. In section 3 we describe optimization framework, and comment their advantages for the purpose of this paper, while in section 4 we introduce innovative techniques to deal with the problem. In section 5 we present the empirical study and discuss their results. Section 6 describes context and related work. Finally, in section 7 conclusions are drawn.

2 Composite Web Service Selection Problem definition

There are different languages that could be used to specify the composite web services, from UML activity diagrams, to BPMN or BPEL. In the remainder of the paper, we assume that the language used to express the composition contains a set of basic construction blocks (This set is summarized in table 1).

Table 1. Basic building blocks of the composite web service description language

basic Building Block	Description
Sequence	Denotes a sequential execution of a set of tasks
Switch	Denotes an alternative choice of the execution between sets of tasks
Fork	Denotes a parallel asynchronous execution of different sets of tasks
Loop	Denotes an iterative execution of a sequence of tasks

The composition will be expressed in terms of abstract services (tasks in the remainder of the paper). There exists a set of concrete candidate services for each task, that are functionally equivalent but have different behavior in terms of quality properties. In this sense, the solution space of the composite web service selection problem, is defined by all the possible combinations of candidate services for each task. We assume that each composite web service includes a single entry and exit task.

Given the previous composite web service definition and constraints the concept of execution path can be defined as in [3]: 'A sequence of tasks (abstract services) $\{t_1, t_2, ..., t_n\}$ such that t_1 is the initial task of the composition, t_n is the final task, and no t_i belongs to alternative branches'. Execution paths will be denoted by ep_k.

As remarked in [3], for each execution path ep_k a probability of execution $freq_k$

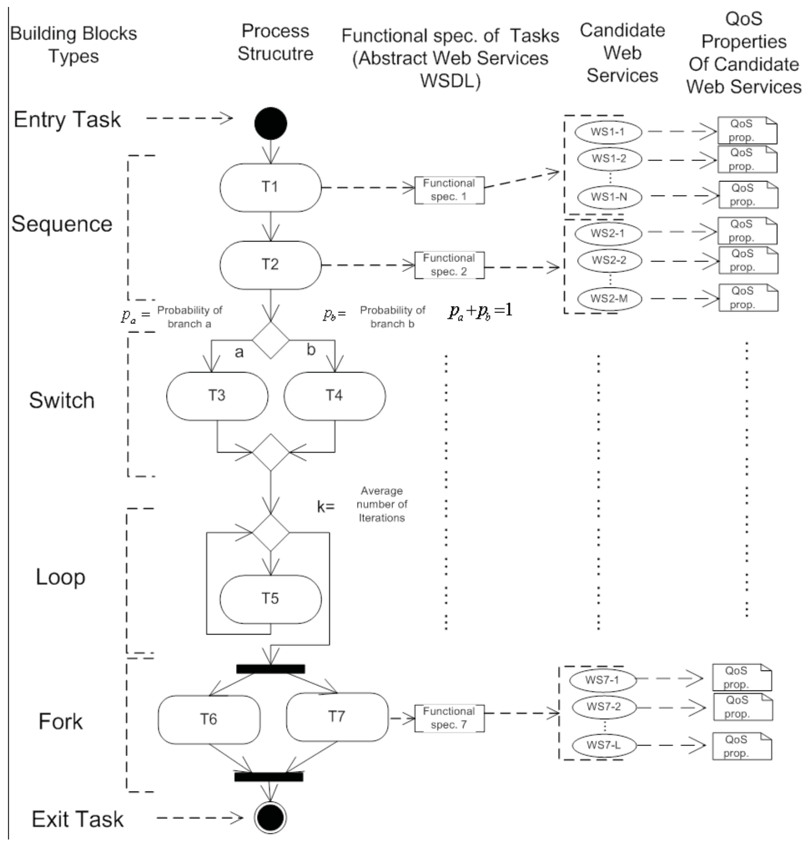

Fig. 1. Building blocks, process specifications and concrete services

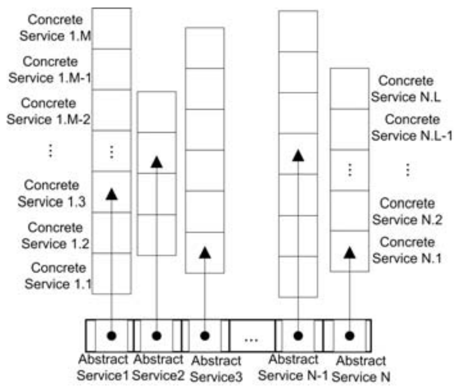

Fig. 2. Solution encoding

can be computed as the product of the probability of execution of the branch conditions included in ep_k. For a composite web service described in any language using the specified building blocks, and according to the stated constraints, all possible execution paths can be built using a simple algorithm.

2.1 QoS Model for Composite Web Services

In this paper, the quality model is based on a set of quality dimensions, which had been used in [3] and [5]. The approach to compute the global QoS of a selection of concrete services for an abstract composite web service is similar to the proposal stated in [5] and [7]. This formulation had been used extensively in the literature with some variations, mainly in the treatment of loops [3] [22]. In our work loops are annotated with an average number of executions k, and the computation of the overall quality values of the loop execution is based on this value. In a similar way, each choice of a switch is annotated with a probability of execution. In table 2 we summarize the aggregation functions applied for each QoS property defined for the composition of web services. The

Table 2. Aggregation functions per QoS Attribute

QoS Property	Aggregation Functions
Cost (c)	$\sum_{i=1}^{n} c_i$
Time (t)	$\sum_{i=1}^{n} t_i$
Reliability (rel)	$\prod_{i=1}^{n} rel_i$
Reputation (rep)	$\frac{\sum_{i=1}^{n} rep_i}{n}$
Security (sec)	$\min_{i=1}^{n} sec_i$
Custom attribute (atr)	$F(attr_i)_{i\in 1\ldots n}$

fitness of a solution is computed by the evaluation of the set of execution paths that conform the description of the composite web service multiplied by their probability of execution $freq_k$. We obtain the fitness of each execution path using each quality properties of the selected concrete services for each task, and applying a Simple Additive Weighting. The specific value of $freq_k$ for each execution path is obtained by monitoring the execution of the composite web service. In our model of composite web service selection it is possible to impose global and local constraints on quality dimensions, and concrete web services dependency constraints as in [3]. For the inclusion of the constraints in the fitness function we have used a relative weight over the feasibility distance as in [5]. We define the feasibility distance $D_f()$ as the number of not meet constraints divided by the total number of constraints. Our final Fitness function for a solution s is:

$$fitness(s) = (1 - w_{feasibility}) * (\sum fitness(ep_k, s) * freq_k) + w_{feasibility} * D_f(s) \tag{1}$$

where:

$$fitness(ep_k, s) = w_{cost} * Cost(ep_k, s) + w_{time} * Time(ep_k, s) + \\ w_{rel} * Reliability(ep_k, s) + w_{rep} * Reputation(ep_k, s) + w_{sec} * Security(ep_k, s) \quad (2)$$

From the previous definition, we can state that the value of $w_{feasibility}$ controls the penality we apply to individuals that violate constraints.

2.2 General solution encoding

In order to apply optimization techniques to our problem we must provide a suitable encoding of solutions. In our solution encoding we have used a structure similar to the vector-based encoding described in [5]. Solutions are encoded as a vector of integer values, with a size equal to the number of tasks. Each value in this vector represents the concrete service selected for each task. In figure 2 we show the structure graphically.

3 Optimization Framework

Most metaheuristic approaches for discrete optimization problems are usually implemented from scratch. In this paper we have used FOM [15] (Framework for Optimization using Metaheuristics), an object-oriented framework that can be used as a general tool for solving optimization problems using metaheuristics. The basic idea behind the framework is to separate the problem from the metaheuristic algorithms used to solve it. This separation allows to fully reuse different metaheuristic optimization components, and partially reuse problem definition in terms of the framework. In this sense, different metaheuristic techniques can benefit from a common solution encoding. Moreover, typically local search techniques based on neighborhood exploration (such as tabu search or simulated annealing), can use a common neighborhood generation structure. From this point of view, the usage of the framework for the purpose of this paper is fully justified, because we need to implement different heuristics and compare their results. In so doing, the usage of the framework reduces significantly the implementation effort and providing other benefits such as declarative configuration and parametrization of termination criteria, monitoring of different parameters during optimization process (including heuristic-specific parameters such as population diversity or number or rejected tabu moves per iteration).

4 Description of the Proposed Techniques

4.1 Tabu Search

Tabu search is based on a intelligent exploration of the solutions neighborhood [11], and had been extensively used because of their simplicity and practical effectiveness. In order to apply this technique to our problem, we must define the neighborhood of our solutions. Given a solution S defined using our general solution encoding, a different solution S_n will be a neighbor of S if there exists a simple move that applied to S_n has as result S. We define a simple movement as change in the selected concrete service for a given task defined in the composition. One of the most basic intelligent exploration strategies in tabu search is the usage of recent memory, that is usually implemented using a tabu list in order to avoid reverse moves -that typically generate cycles in the search path and lost of effectiveness-. In our implementation we use a fixed size list with a most recently used policy of insertion. In addition to the core tabu technique, our search algorithm incorporates an aspiration condition. This extension allows that tabu conditions do not prohibit the exploration of really promising solutions. In this sense, if a tabu move improves the best solution found, it will be applied.

4.2 Hybrid Genetic Algorithm

Genetic algorithms maintain a population of individuals that represent more or less efficient solutions to the problem [12]. This population evolves along generations until a termination criterion is reached. During each generation, operators are applied to the individuals generating changes in the individuals and the whole population. Our initial population is generated randomly, and we use the standard two-points crossover operator [9] and a mutation operator that selects the concrete service associated to a task and selects randomly a different service from the set of candidates.

Local improver: Various strategies of hybridization have been suggested in the literature when using genetic algorithms [18]. The usage of hybrid techniques allows to escape from local optima convergence and improve the convergence speed to the global optimum. In this paper we use a local improvement procedure. This procedure is based on an iterative neighborhood search so that a given solution is replaced with best neighbor found. In our initial implementation we explored the whole neighborhood of each individual of the population. However, the computational cost of this exploration for each individual prevents from the execution of enough evolution iterations to provide a sufficient diversification of search in the solution space. In this work we propose the random exploration of a percentage of the neighborhood.

5 Experimental Results

We have tested our composite web service problem model and algorithms in wide set of instances. Those instances are randomly generated based on a problem model by a problem generator. Additionally, we have designed experiments that are executed a repeatedly for each generated problems. The execution of the algorithms had been performed on a 2.7 GHz Dual Core Intel CPU, with 2

Table 3. Statistic Data models used to generate our problem instances

Data	Stat. Dist. (Small-Medium)	Stat. Dist. (Medium-Large)
Tasks	Uniform(min:4, min:20)	Uniform(min:20, max:60)
Execution paths	Uniform(min:4 max: 20)	Uniform(min:20 max: 60)
Tasks per execution path	Uniform(min:4, max:20)	Uniform(min:10, max:60)
Candidate services per task	Uniform(min: 2, max:15)	Uniform(min: 15, max:30)
Value of Cost	Uniform(min:0.1, max:1)	Uniform(min:0.1, max:1)
Value of Time	Uniform(min:0.1, max:1)	Uniform(min:0.1, max:1)
Value of Reliability	Uniform(min:0.9 max:1)	Uniform(min:0.9 max:1)
Value of Reputation	Uniform(min:0.1, max:1)	Uniform(min:0.1, max:1)
Value of Security	Uniform(min:0.5, max:1)	Uniform(min:0.5, max:1)

GB of RAM. In table 3 we summarize the data used to generate our problem instances.

In our experimental results we have applied 4 different optimization techniques, whose parameters are summarized below:

- A basic genetic algorithm (GA).
  - Population size: 50 individuals.
  - Crossover probability: 0.7
  - Mutation probability: 0.1
  - Random selection for mutation.
  - Tournament of three individuals for selection for Crossover.
- The hybrid genetic algorithm described previously in this work (HGA).
  - Population size: 50 individuals.
  - Crossover probability: 0.7 (it is also the value of death rate, because the size of our population is constant)
  - Mutation probability: 0.2
  - Random selection for mutation.
  - Tournament selection for Crossover of three individuals.
  - Percentage of explored neighbors: 5 for small size problems and 2 for medium.
- The tabu search algorithm described previously in this work (TS).
  - Memory structure of 40 moves for small problem instances and 100 moves for medium, with a most recently used policy.
  - An aspiration criteria to select better than current optimal solutions.
- Iterative Steepest Descent [19]using random initial solutions (ISD).

General parameters of the experiment:

- Weight of QoS properties: w_{cost}=0.4, w_{time}=0.2,w_{rel}=0.1, w_{rep}=0.1, w_{sec}=0.2.
- Feasibility Weight: $w_{feasibility}$=0.5.
- Global Constraints:
  - Cost¡=0.8*(Max length of computed execution Paths)
  - Time¡=0.8*(Max length of computed execution Paths)

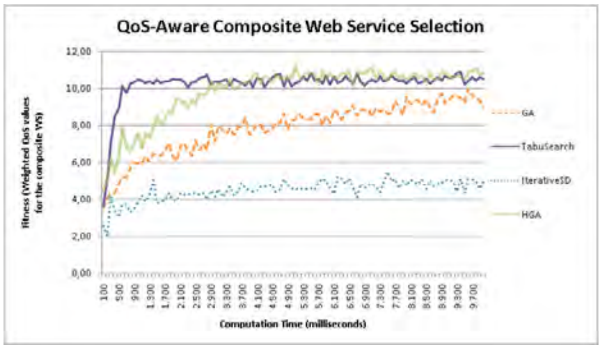

Fig. 3. Performance of metaheuristics for small-medium size composite web services

– Number of repeated executions for each problem instance and execution time: 5
– Number of problems generated for each problem size: 5

The results shown in figure 3 and 4, represent the average fitness obtained by the different heuristics to the problems generated with small and medium size problem models. Both figures show the performance of heuristics obtained with different computation times ranging from 100 milliseconds to 10 seconds. Figure 3 shows that for execution times minor to 2 seconds and small problem sizes, TS performs better than the rest of the heuristics, and that HGA performs better than GA. For longer execution times, HGA performs better than the rest but the difference with TS is relatively small. Figure 4 shows that for medium size problem instances the results are different. The size of the neighborhood of solutions in this kind of problem makes TS perform bad for short execution times -the search is not diversified enough, because TS performs a smaller number of iterations. HGA shows the best and stable performance of all the heuristics evaluated in this case. All solutions found as final results for each technique were feasible.

However, our proposals have a main drawback: the parameters of this heuristics must be well tuned to obtain good results. When using tabu search during our experimentation phase we must fine tune the size of memory, because for many instances of the problems search cycles obtaining poor results, a strategy of long term memory could help to overcome this problem. Moreover, our hybridization strategy is based on exploration of the neighborhood, the percentage of the neighborhood explored for each individual is a key factor to determine. The algorithm provides results similar to a basic GA if this percentage is small. If this percentage is high, the exploration can obtain good improvements but our algorithm can evolve less generations and results are worst than using a basic GA. Moreover, experiments with bigger problem instances show that basic GA provides better results because of the growth of the neighborhood size.

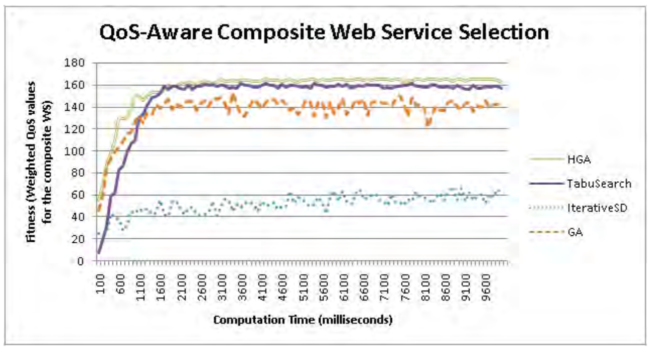

Fig. 4. Performance of metaheuristics for medium-large size composite web services

6 Related Work

Dynamic selection and late binding of web services are amongst the more promising capabilities that SOA brings. In fact it has been identified as a main research area in SOA systems [14], and therefore is being addressed as a main research field by both academy and industry. In this area, QoS-aware composite web services selection appears as a challenging problem. Two kind of optimization strategies had been formulated for this problem in literature [22] [2]:

- Local service QoS-Aware selection. In those approaches, the best candidate service for each task is selected according to the quality properties and stated preferences. This kind of methods has two main drawbacks:
  - Obtained solutions are suboptimal respect to the overall quality of the composite web service.
  - Global constraints according the structure of the composite web service and their quality properties can not be imposed
- Global composite service QoS-aware selection. The whole set of concrete services that are used to implement the composite services are optimized according to their QoS properties. Therefore, QoS global constraints from the whole composition perspective can be formulated.

Global composite service QoS-aware optimal selection has been identified as a NP-hard problem [2], [4]. In order to deal with this complex problem different approaches have been proposed:

- **Usage of Integer** [22] [1]**,Linear** [6] **or Mixed (I/L) Programming techniques** [3] [17]. Although this approaches provide the global optimum of the problem, and their performances is better for small size instances of the problem, genetic algorithms outperforms these techniques for large size instances [5]. Moreover metaheuristics are more flexible, because those techniques can consider non-linear composition rules, constraints and fitness function formulations [2].

– **Usage of heuristic techniques**. Given the intractable nature of the problem, the usage of heuristics is a logical choice. In [13] some specific heuristics are developed to solve the service composition problem. Applications of metaheuristics to this problem, are present in the literature, mainly using different Genetic Algorithm based approaches, incorporating variants to the work presented in [5], either on the encoding scheme or the fitness function or QoS model [10] [20] [21], or using population diversity handling techniques [23] [24]. In [8] a multi objective evolutionary approach is used to identify a set of optimal solutions according to different quality properties without generating a global ranking. In [16] fuzzy logic is used to relax the QoS constraints, in order to find alternative solutions when it is not possible to find any solution to the problem. Some authors have proposed the usage of simulated annealing [21]. Our work starts from [5], using a similar QoS model adding some additional quality properties and a modified, simple additive weighting fitness function. Moreover we adopted the executions paths based description of process structure used in [3] among others.

7 Conclusions

In this paper we address the optimal QoS-aware selection in composite web services. We had proposed metaheuristic based algorithms: hybrid genetic algorithm and tabu search, and compared their performance against other two techniques: iterative steepest descent, and a basic Genetic Algorithm. Experimental results show that hybrid genetic algorithm performs better than the basic version for small and medium problem sizes, and that the tabu search based algorithm is not better except for small problem instances and short run times (that can be caused by the absence of long term memory). Future work will perform a more intensive experimentation and tunning of parameters for each technique and analysis of behavior under stronger constraints, in order to confirm the conclusions obtained here, and the usage of other metaheuristics such as ant systems and simulated annealing or different optimization techniques such as Linear/Integer Programming solvers.

8 Acknowledgments

This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472), and Andalusian Government project ISABEL (TIC-2533).

References

1. R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint driven web service composition in meteor-s. In *SCC '04: Proceedings of the 2004 IEEE International Conference on Services Computing*, pages 23–30, Washington, DC, USA, 2004. IEEE Computer Society.

2. D. Ardagna and B. Pernici. Global and local qos guarantee in web service selection. In *Business Process Management Workshops*, pages 32–46, 2005.
3. D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *Software Engineering, IEEE Transactions on*, 33(6):369–384, 2007.
4. P. A. Bonatti and P. Festa. On optimal service selection. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 530–538, New York, NY, USA, 2005. ACM.
5. G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. An approach for qos-aware service composition based on genetic algorithms. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1069–1075, New York, NY, USA, 2005. ACM.
6. V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti. Efficient provisioning of service level agreements for service oriented applications. In *IW-SOSWE '07: 2nd international workshop on Service oriented software engineering*, pages 29–35, New York, NY, USA, 2007. ACM.
7. J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308, April 2004.
8. D. Claro, P. Albers, and J. Hao. Selecting web services for optimal composition. In *Proc. Int?l Conf. Web Services (ICWS ?05)*, 2005.
9. J. Dreo, A. Petrowski, and E. Taillard. *Metaheuristics for Hard Optimization*. Springer, 2003.
10. C. Gao, M. Cai, and H. Chen. Qos-driven global optimization of services selection supporting services flow re-planning. In *Advances in Web and Network Technologies, and Information Management*, Lecture Notes in Computer Science, pages 516–521. Springer, 2007.
11. F. Glover. Tabu search: part i. *ORSA Journal on Computing*, 1:190–206, 1989.
12. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine learning*. Addison Wesley, 1989.
13. M. C. Jaeger, G. Mühl, and S. Golze. Qos-aware composition of web services: An evaluation of selection algorithms. In *Lecture Notes in Computer Science*, Lecture Notes in Computer Science, pages 646–661. Springer, 2005.
14. M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, November 2007.
15. J. A. Parejo, J. Racero, F. Guerrero, T. Kwok, and K. Smith. Fom: A framework for metaheuristic optimization. *Lecture Notes in Computer Science*, 2660:886–895, 2003.
16. M. D. Penta and L. Troiano. Using fuzzy logic to relax constraints in ga-based service composition. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, June 2005.
17. Y. Qu, C. Lin, Y. Wang, and Z. Shan. Qos-aware composite service selection in grids. *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*, pages 458–465, Oct. 2006.
18. J. Renders and S. Flasse. Hybrid methods using genetic algorithms for global optimization. *IEEE Trans. on Systems, Man, and Cybernetics. Part B: Cybernetics*, 26(2), 1996.
19. J. A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer, 2005.

20. S. Su, C. Zhang, and J. Chen. An improved genetic algorithm for web services selection. In *Distributed Applications and Interoperable Systems*, volume 4531/2007 of *Lecture Notes in Computer Science*, pages 284–295. Springer, 2007.
21. H. Wang, P. Tong, P. Thompson, and Y. Li. Qos-based web services selection. *icebe*, 0:631–637, 2007.
22. L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.
23. C. Zhang, S. Su, and J. Chen. Efficient population diversity handling genetic algorithm for qos-aware web services selection. In *Computational Science ? ICCS 2006*, volume 3994/2006 of *Lecture Notes in Computer Science*, pages 104–111. Springer, 2006.
24. C. Zhang, S. Su, and J. Chen. Diga: Population diversity handling genetic algorithm for qos-aware web services selection. *Comput. Commun.*, 30(5):1082–1090, March 2007.