Optimizing Service Selection Using Hybrid Multi-objective Genetic Algorithms

Bo Li[1], Changsheng Zhang[1], Baoxing Bai[2]

[1]Changchun University of Science and Technology, Changchun, China

[2]College of Information Science & Engineering, Northeastern University, China

Keywords: List the keywords covered in your paper. These keywords will also be used by the publisher to produce a keyword index. Optimizing Service Selection, Hybrid Multi-objective, Genetic Algorithms.

Abstract. Developments in web services lead increasing numbers of enterprises to focus themselves on core business by outsourcing parts of service flows. As an important measure of the web service, quality of service (QoS), determines the success of the combined services to some extent, highlighting the need for service users and providers to reach an agreement on QoS of web services. This agreement is known as a Service-Level Agreement (SLA).A particular task may be performed by many services with the same functions but different QoS values.A method for automatically selecting the services is thus needed.

Introduction

For some complicated tasks, users' service requests must be satisfied by a combination of different web services, because one web service alone is insufficient. With the advent of service-oriented architecture, the combined application is defined as the abstract flow comprising a set of abstract services. During operation, a specific web service is selected for each abstract service of the abstract service flow and is run to ensure loose coupling and flexibility of the combined application design. As an important measure of the web service, quality of service (QoS), determines the success of the combined services to some extent, highlighting the need for service users and providers to reach an agreement on QoS of web services. This agreement is known as a Service-Level Agreement (SLA).

A particular task may be performed by many services with the same functions but different QoS values. Users are usually unaware of the specific services involved, and all users must do is to specify end-to-end QoS constraints (e.g., average end-to-end response time, minimum overall throughput, and maximum overall cost) in the SLA, based on the tasks. QoS-based service selection (QSS) aims to select a series of services or service configurations that satisfy all application-levelQoS constraints, i.e., search the optimal web service combination for users to satisfy end-to-end QoS constraints for a given SLA.

Although the number of web services is increasing enormously, it is infeasible to obtain an optimal combination that satisfies the SLA requirements (end-to-end QoS constraints) via exhaustive search, because there are so many possible combinations in this case. Consider a service flow of 10 services, each of which has 1,000 potential services. So there are 1000^{10} possible combinations in total. This combination problem can be viewed as the well-known NP-hard problem.

Considerable attention has been paid to QoS-based selection and combination of web services over the past few years. In 2004, Liu et al. proposed a user feedback-based extendable QoS computation model which supports fair and open management of QoS data [1]. But QoS combination was neglected in their work. In [5-6], Zeng et al. preferred dynamics and quality-oriented service selection methods, using a global planning strategy to find the optimal service components and combine services. Nemhauser et al. proposed finding the optimal scheme for selecting service components via mixed linear programming [3]. In 2007, Ardagna et al. proposed a similar method, and extended the linear programming model and local constraints. For small-scale problems, linear programming is highly effective but poorly extendable, because a

search algorithm with exponential time complexity is used [2]. In 2007, Yu et al. proposeda heuristic algorithm that solves problems by searching for near-optimal solutions [4]. In their paper, two QoS-based combined service models (a combined model and a graph model) were proposed. The heuristic algorithm was introduced to each model, and the time complexities of both models were polynomial and exponential, respectively. In 2009, Lecue proposed using the web semantics for I/O parameters and QoS properties, and provided a genetic algorithm-based method for combining web semantic services.

QSS Mathematical Optimization Model

For the QSS problems in this paper, only the sequential service combination model is considered, and other combination models can be simplified or transformed to the sequential model. For example, in [7], some techniques were used to process multiple execution routes and cycles. To further simplify modeling, services with different QoS but the same functions are viewed as different web service versions.

Consider *S*to be the set of service types classified by web service functions. For each service type $S_j \in S$,S_j=$\{s_{j_1}, s_{j_2}, \cdots, s_{j_n}\}$ consists of web service versions that have been released with the same functions but different non-functional parameters.

The QoS value of the combined web service is dependent on both the components' QoS values and the combination structure. In this paper, only the QoS properties that need to be minimized are considered for combined services, because the QoS properties that need to be maximized can be transformed into minimization problems. Let the vector $Q_s = (q_1(s), q_2(s), \cdots, q_r(s))$ denote the QoS value of the service, *s*, where the function $q_i(s)$ provides the value of the i^{th} QoS property of *s* released by the service provider. $Q_{cs} = (q_1'(cs),\ q_2'(cs)\ , \cdots, q_r'(cs))$ denotes the QoS property vector of the combined service $cs = \{s_1, s_2, \cdots, s_n\}$,where $q_i'(cs)$ is the value of the i^{th} QoS property of *cs* and can be obtained by aggregating the QoS property values of relevant services. Assume the user has requirements on the aggregated QoS property values of the combined web service, and the requirements are represented by the vector $Q = (Q_1, Q_2, \cdots, Q_m), 1 \le m \le r$, then Q is named the global QoS constraint.

Definition (feasible solution): for an abstract process $P=\{S_1,S_2,...,S_n\}$, where $S_j \in S$ $(1 \le i \le n)$ denotes the type of services with the same functions, and a given global QoS constraint vector $Q = (Q_1, Q_2, \cdots, Q_m), 1 \le m \le r$, *cs*is a feasible solution to the QSS optimization problem only when *cs* contains a correct service and *cs*'s aggregated QoS value satisfies the global QoS constraint for each service type in *P*, i.e., $q_k'(cs) \le Q_k, \forall : k \in [1, m]$. All feasible solutions form the feasible region, D.

The QSS problem is the typical multi-target optimization problem.For the given abstract service flow and the users' global QoS constraints, the optimal service combination that satisfies users' requirements may be a set of Pareto optimal solutions. Hence, the objective of solving the QSS problem is to find the set of optimal service combinations that satisfy users' requirements.

Three types of QoS aggregate functions are considered in the model: (1)sum, (2)multiplication, (3)minimization. Examples are shown in Table.1.

Table.1:QoS aggregate functions

Aggregate types	Properties	Functions
Sum	f_1: Response time f_2: Cost f_3: Latency	$q'(cs)=\sum_{j=1}^{n} q(s_j)$
	f_4: Reputation f_5: Compliance	$q'(CS)=1/n\sum_{j=1}^{n} q(s_j)$
Multiplication	f_6: Availability f_7: Reliability f_8: Success ability	$q'(cs)=\prod_{j=1}^{n} q(s_j)$
Minimization	f_9: Throughput f_{10}: Relevancy	$q'(cs)=\min_{j=1}^{n} q(s_j)$

QoS property-based QoS functions of the QSS problems are defined in Table .1, and the mathematical optimization model of the QSS problems is defined as follows:

Min: $f(cs)=(f_1(cs), f_2(cs))$ (5.1)

subject to: $cs \in D$ (5.2)

$f_i(cs) \le Q_i$, $\forall i \in \{1,2,\cdots,10\}$, $Q_i \in Q$ (5.3)

Equation 5.1 is the two-dimensional optimization objective function, the objective of which is the end-to-end response time and the cost, which are relevant or conflicting. Equation 5.2 is the basic constraint, Equation 5.3 represents the QoS constraints, and Q is the users' QoS requirements.

Experimental Analysis of QSS

The proposed genetic Multi-objective optimization algorithm will be simulated in this section. To evaluate the performance of INSGA2, ISPEA2, IMOEA/D, and HDMOGA in solving the QSS problem, they are compared with NSGA2, SPEA2, and MOEA/D. In this paper, test instances of different scales are set up for the QSS optimization problem, including three test data sets a. data, c. data, and i. data, each of which contains 100,000 web services. The QoS parameters have different relations in each data set. The QoS parameters are negatively correlated for a_data, positively correlated for c. data, and independent for c. data. The other test data set, d. data, is small in scale and contains 2,500 services. To evaluate the performance of the algorithms under different data scales, the test instance shown in Table 5.2 is set up, where the combination length denotes the number of abstract service flows, the number of service candidates denotes the number of potential services that can be chosen in each abstract service class, and the QoS constraints denotes the users' service requirements.

Table.2: The instances for QSS

Test Data Sets	Test Cases	Combination Length	Number of Service Candidates	QoSConstraints
A. data (anti-correlation)	Test1	10	10000	$(5.00, 0.64, 0.47, 2.70\times10^{-4}, 5.76\times10^{-6}, 5.53\times10^{-4}, 0.80, 1.61)$
	Test2	20	5000	$(7.49, 0.43, 0.41, 9.46\times10^{-9}, 8.44\times10^{-10}, 7.75\times10^{-12}, 0.63, 1.24)$
	Test3	40	2500	$(16.2, 0.45, 0.39, 1.32\times10^{-15}, 1.63\times10^{-18}, 2.48\times10^{-20}, 0.62, 1.24)$
B. data (correlation)	Test4	10	10000	$(4.17, 0.35, 0.44, 8.34\times10^{-7}, 3.0\times10^{-6}, 5.92\times10^{-7}, 0.17, 2.45)$
	Test5	20	5000	$(6.99, 0.30, 0.27, 1.36\times10^{-11}, 2.15\times10^{-11}, 3.60\times10^{-12}, 0.17, 1.74)$
	Test6	40	2500	$(11.9, 0.26, 0.29, 3.09\times10^{-24}, 1.07\times10^{-22}, 4.63\times10^{-23}, 0.17, 1.29)$

C. data (independence)	Test7	10	10000	$(2.90, 0.30, 0.35, 1.54\times10^{-5}, 1.70\times10^{-7}, 2.09\times10^{-7}, 0.54, 1.93)$
	Test8	20	5000	$(5.57, 0.37, 0.34, 6.12\times10^{-11}, 8.12\times10^{-13}, 1.19\times10^{-12}, 0.25, 1.31)$
	Test9	40	2500	$(10.1, 0.27, 0.31, 6.03\times10^{-21}, 2.12\times10^{-23}, 1.02\times10^{-23}, 0.16, 1.20)$
D. data (independence)	Test10	5	500	No constraints
	Test11	10	250	No constraints
	Test12	20	125	No constraints

For test sets a_data, c_data, and i_data, due to the large number of services, the IS clustering method is first used in the proposed optimization algorithm to cluster potential services, and the number of service clusters varies with the length of the service combination. For test instances whose combination lengths are 10, 20, and 40, the number of clusters for each potential service class is set at 100, 50, and 25, respectively. The other three test instances are not clustered because of the small number of services.

Convergence analysis of all algorithms must be performed in this paper to obtain the maximum iteration time of all algorithms under different test cases and performance indexes after convergence. During the analysis of algorithm convergence, test instances with different computation complexity are selected (i.e., Test1, Test3, Test4, Test6, Test7, Test9, Test10, and Test12). The algorithm parameters use the originally determined values, that is, the evolved population, $N=50$, the external population, $M=200$, the crossover probability, $p_c=0.6$, and the mutation probability, $p_m=0.1$. For the computation of H, the reference point is set at (30, 30). Each algorithm is tested 10 times on each test instance. Figure 1 shows the experimental results.

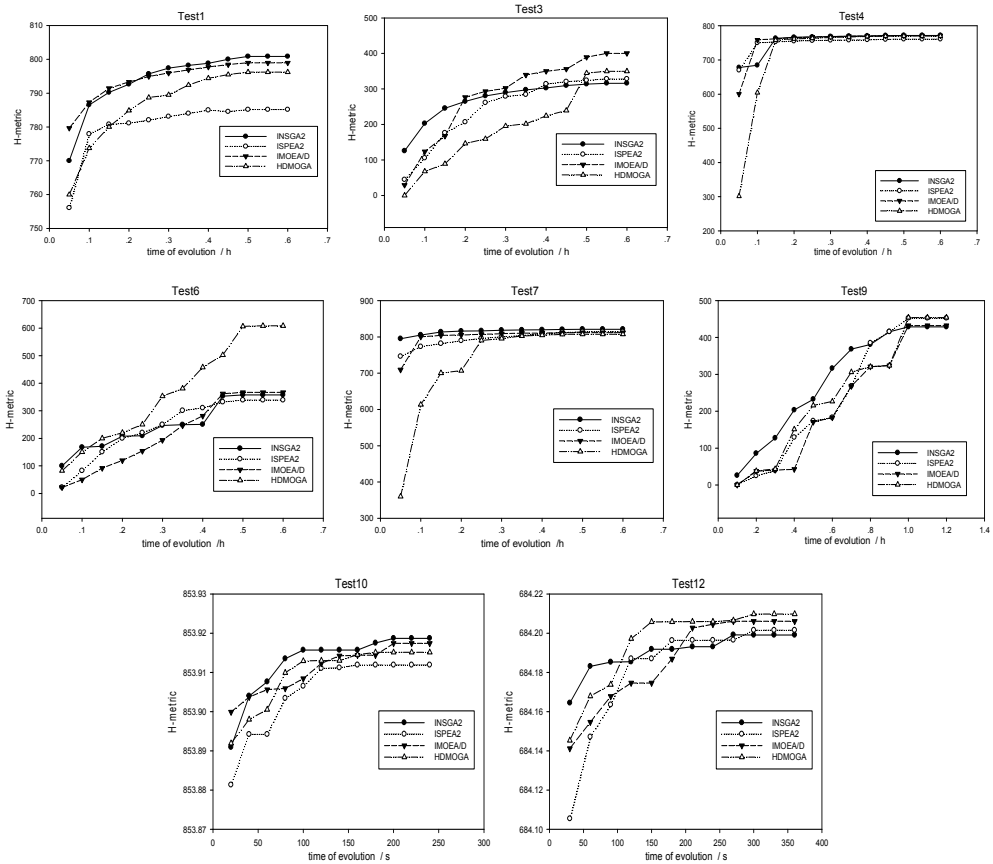

Figure 1: The convergence process of algorithms

From Figure 1, it can be seen that for Test1, Test3, Test4, Test6, Test7, Test9, Test10, and Test12, all algorithms converge after iterations of 0.45h, 0.5h, 0.15h, 0.45h, 0.25h, 1h, 200s, and 300s, respectively. Due to the differences among algorithms in the methods for evaluating individuals, the evolution time is used in this paper as the algorithms' termination condition. Based on convergence analysis, Table 3 shows the algorithms' conditions for terminating iterations for each test instance.

Table 3 Tests 'terminating conditions

Test	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Test11	Test12
T/h	0.40	0.45	0.5	0.15	0.3	0.45	0.25	0.6	1.0	0.055	0.07	0.075

Here, algorithm parameters will be optimized. The algorithms' conditions for terminating iterations in Table 3 will be used for parameter adjustment. The major parameters of the proposed algorithm include the size of the population, *N*, the size of the external population, *M*, the mutation probability, p_c, the mutation probability, p_m, the local optimization threshold, *L*, and the size of the neighborhood, *S*. M will not be adjusted, because it is dependent on user requirements. S is set at $\sqrt{N}$. So only *N*, p_c, p_m, and *L* will be adjusted.

Table4: Parameter values

Parameters	Value 1	Value 2	Value 3	Value 4	Value 5
N	10	20	50	70	100
p_c	0.5	0.6	0.7	0.8	0.9
p_m	0.05	0.10	0.15	0.20	0.25
L	10	20	30	40	50

Similarly, three test instances of different scales (Test1, Test5, and Test9) are selected for parameter optimization. Because the size of the test samples is very large when usingparameter combination as the testing method, the following test procedures are defined to test parameter values more objectively and reduce the test time: Firstly, fix two parameters,$Parameter_1$ and $Parameter_2$ empirically, and then change the value of another parameter,$Parameter_3$. Thus after testing, the value of $Parameter_3$ can be determined.The other three parameters can be determined in this way. For the parameter testing of this paper, each algorithm is tested 10 times for each test instance. The average value of *H* is then computed while the reference point is set at (100, 1). *N* is first tested and we set $p_c=0.7$, $p_m=0.1$, *and* $L=20$. The experimental results are shown in Figure 2.

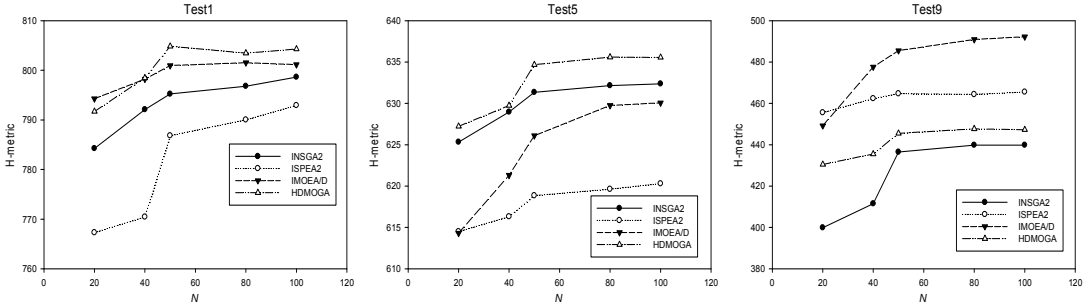

Figure 2:Parameter *N*'sadjustment process

The test results show that for the three test instances, all algorithms converge when N=50. Hence, N is set at 50. Next, *pc* will be tested, and we set N=50, p_m=0.1， and $L=20$. The experimental results are shown in Figure 3.

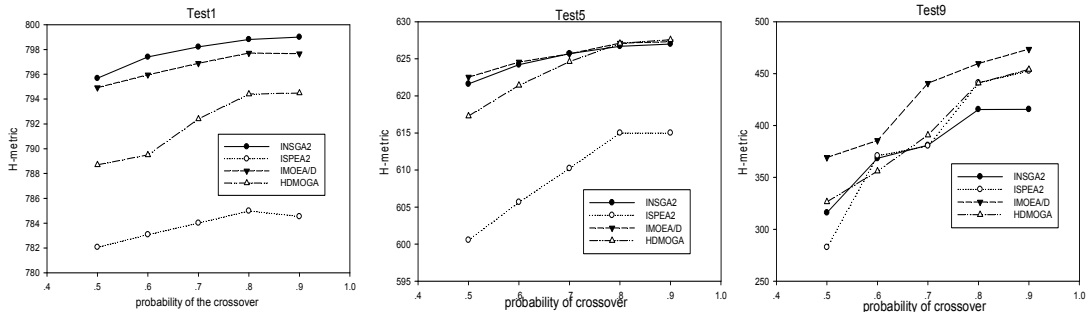

Figure 3 :Parameter p_c'sadjustment process

The test results, show that for the three test instances, all algorithms converge when p_c=0.8. Hence, p_c is set at 0.8. Next, p_m will be tested, and we set $N=50$, $p_c=0.8$,and $L=20$. The experimental results are shown in Figure 4.

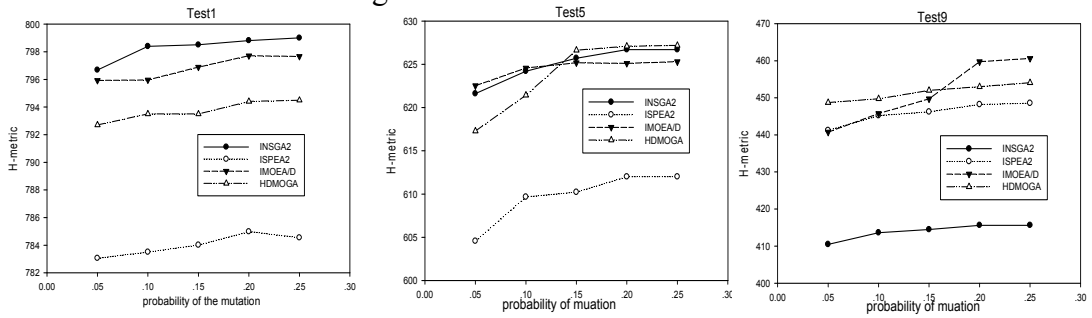

Figure 4:Parameter p_m'sadjustment process

From the test results, it can be seen that for Test1 with short service flow, Test5 with medium service flow, and Test9 with long service flow, all algorithms converge when p_m is 0.1, 0.15, and 0.2, respectively. Based on this, L will be tested by setting $N=50$, $p_c=0.8$, and $p_m=0.1$,0.15, and 0.2 for Test1, Test5, and Test9, respectively. Experimental results are shown in Figure 5.

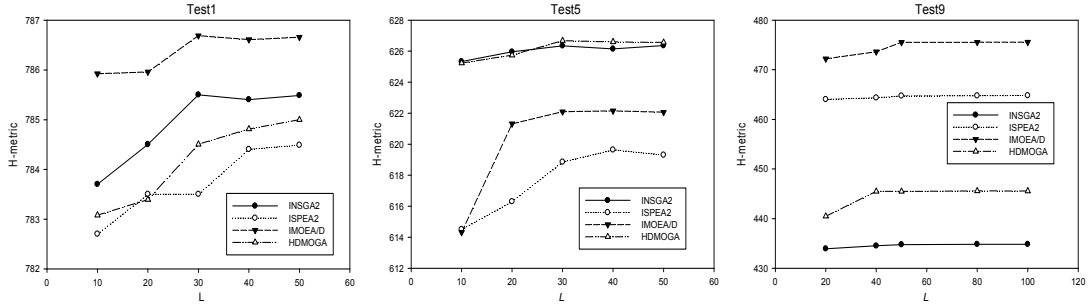

Figure 5:Parameter L'sadjustment process

The test results show that for the three test instances, H is suitable for all algorithms when L=30. Hence, $L$is set at 30.

According to the above convergence analysis and parameter adjustment, the conditions for terminating algorithm iterations and the parameter settings under different test instances are shown in Table 5.5.

Table 5.5: Algorithm parameters

Test	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Test11	Test12
T/h	0.40	0.45	0.5	0.15	0.3	0.45	0.25	0.6	1.0	0.055	0.07	0.075
N	50	50	50	50	50	50	50	50	50	50	50	50
M	200	200	200	200	200	200	200	200	200	200	200	200
p_c	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
p_m	0.1	0.15	0.2	0.1	0.15	0.2	0.1	0.15	0.2	0.1	0.15	0.2
L	30	30	30	30	30	30	30	30	30	30	30	30

Summary and Conclusions

In this chapter, QSS optimization was first modeled, and the crossover operations, mutation operations and local optimization algorithms were proposed. A preference-based population initiation strategy was devised, and finally, the hybrid Multi-objective genetic algorithm proposed in Chapter 3 was used to optimize QSS problems of different scales, and different metrics were used to evaluate different algorithms. Comparative experiments on the proposed hybrid Multi-objective genetic algorithm showed that different improved algorithms displayed different search abilities in different search spaces. Compared with unimproved algorithms, the improved INSGA2, ISPEA2, and IMOEA/D algorithms showed greater search ability across the entire target space.

Reference

[1] Y.liu,A.H.H.Ngu, and L.zeng. QoS computation andpolicing in dynamic webservice selection. In InternationalWorld Wide Web Conference.pp: 66–73, 2004.

[2] I.Maros. Computational Techniques of the Simplex Method.Springer, 2003.

[3] G.L.Nemhauser and L.A.Wolsey. Integer andCombinatorial Optimization. Wiley-Interscience, New York,NY, USA, 1988.

[4] T.Yu, Y.Zhang, and K.-J.Lin. Efficient algorithms for webservices selection with end-to-end QoS constraints. ACMTrans. on the Web, (1)1, 2007.

[5] L.Zeng, B.Benatallah, M. Dumas, J. Kalagnanam, and Q. Z.Sheng. Quality driven web services composition.InInternationalWorldWideWebConference,pp: 411-421,2003.

[6] L.Zeng, B.Benatallah, A.H.H.Ngu, M.Dumas, J.KalagnanamandH.Chang. QoSawaremiddleware forweb services composition. IEEE Trans. on SoftwareEngineering, (5)30:311–327, 2004.

[7] J.Cardoso,J.Miller, A.ShethandJ.Arnold. Quality ofservice for workflowsandwebserviceprocesses. Journal ofWeb Semantics, 1:281–308, 2004.

[8] PasiFränti and Olli Virmajoki. Iterative shrinking method for clustering problems. Journal of Pattern Recognition, (5)39:761-775, 2006.