International Journal of Grid Distribution Computing
Vol.7, No.4 (2014), pp.1-10
http://dx.doi.org/10.14257/ijgdc.2014.7.4.01

A Hybrid Particle Swarm Optimization Algorithm for Service Selection Problem in the Cloud

Wanchun Yang[1] and Chenxi Zhang[2*]

[1]*School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China*
[2]*School of Software Engineering, Tongji University, Shanghai 201804, China*
wcyang.tj@gmail.com, chenxizhang10@126.com

Abstract

With the growing number of alternative services in the cloud environment, users have put forward new requirements to solve the service dynamic selection problem quickly and efficiently. In this paper, an evaluation model of service process which considers concurrent requests and service association is proposed. This model evaluates the service process from three dimensions which are functional quality dimension, non-functional quality dimension and transactional dimension. To solve the service selection problem efficiently, we first design a novel coding strategy of particle, and then propose an approach based on hybrid particle swarm optimization algorithm which combines the crossover and mutation operators of genetic algorithm. The experimental results show that our proposed approach is feasible and effective.

Keywords: *Cloud Computing, Service Selection, Concurrent Requests, Service Association, Hybrid Particle Swarm Optimization*

1. Introduction

Recently, service-oriented computing, in particular cloud computing technology has been developed rapidly. Cloud computing model can combine various services to form new value-added services in order to satisfy end users' requirements [1, 2]. With the growing number of available services in the cloud environment, users have put forward new demands to solve the service selection problem quickly and efficiently.

The problem of service selection and composition has received a lot of attention by researchers. Zeng *et al.*, [3] formalized the service selection problem with global optimization as an integer linear programming problem. However, integer linear programming is not suitable for large-scale problems. In [4], a hybrid approach for efficient service composition with end-to-end QoS constraints was proposed. The approach first decomposed the global constraints into local constraints, and then used local selection to find the best services that satisfy these local constraints. In [5], a branch and bound algorithm to solve the QoS-aware service selection problem was proposed. The notion of skyline to reduce the number of candidate services was considered by many authors [6, 7].

To achieve better scalability, various kinds of evolution algorithms for the service selection problem are presented to find a solution which close to the optimum. Zhao *et al.*, [8] proposed an improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition. Fan *et al.*, [9] presented a cooperative evolution

[*] Corresponding Author

algorithm to solve the service selection problem. In [10], a hybrid clonal selection algorithm was proposed for QoS-aware service selection problem. In [11], an improved artificial bee colony approach for QoS-aware service selection was presented. In [12], a genetic algorithm with prioritized objective functions was proposed for service composition. In [13], a fuzzy-guided genetic algorithm was applied to the service selection problem.

The risk of failures may occur throughout the service execution. Therefore, transactional properties have been taken a high attention in the cloud. Sun *et al.*, [14] conducted a survey of SLA (Service Level Agreement) assurance from a transactional risk perspective in the cloud. Hammadi *et al.*, [15] proposed a framework for SLA assurance in cloud computing. In the framework, transactional risk assessment module was involved. Haddad *et al.*, [16] integrated transactional properties into QoS-aware service selection. Since they only performed local optimization, they cannot support global constraints in the service selection problem. In [17], a transactional-QoS driven approach for service composition based on Petri-net was proposed.

However, the aim of current researches in cloud-based service selection is to maximize the overall object only for single service process, and the researches ignore the service association in service selection problem. Furthermore, the execution efficiency of service selection methods needs to be improved further. In this paper, we introduce an evaluation model of service process which evaluates the service process from three dimensions which are functional quality dimension, non-functional quality dimension and transactional dimension. The comprehensive evaluation model also considers concurrent requests and service association problem. In this paper, the problem of service association is divided into two levels: the granularity problem in abstract process level; the service dependency problem in concrete process level. To solve the service selection problem efficiently, we first design a novel coding strategy, and then propose an approach based on hybrid particle swarm optimization algorithm which combines the crossover and mutation operators of genetic algorithm. The experimental results show that our proposed approach is feasible and effective.

The remainder of this paper is organized as follows: Section 2 briefly states the service selection problem. Section 3 introduces an evaluation model of service process, while Section 4 presents an approach based on hybrid particle swarm optimization. The analysis of simulated results is done in Section 5. Finally, Section 6 concludes our work.

2. Problem Statement

In the service selection problem, abstract service process (ASP) and service classes (SC) are given in advance, and then service selection dynamically bind the concrete service for each abstract service in the abstract process from the service classes. The aim of service selection is to construct a concrete service process (CSP) which meets users' constraints, and the objective functions are maximally optimized. The candidate services in each service class have similar function but different QoS and transactional properties. Figure 1 illustrates the service selection problem.

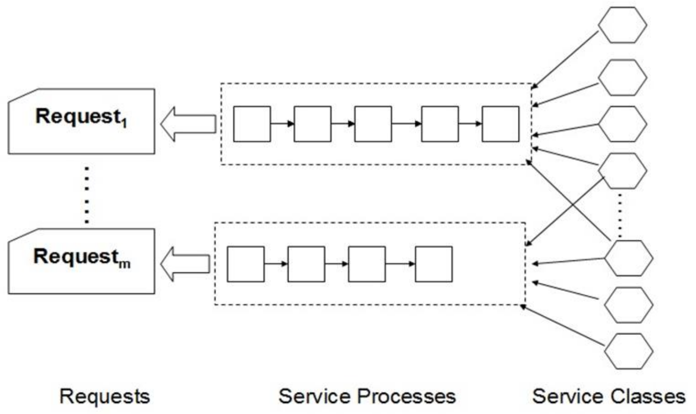


Figure 1. Service Selection Problem

As shown in Figure 1, assume M concurrent requests $M = \{asp_1, asp_2, ...asp_m\}$. Each abstract service in asp_i is implemented by different candidate services. So to the M concurrent requests, there are plenty of solutions. Therefore, performing an exhaustive search to find the best solution that satisfies end-to-end constraints is not efficient, as the number of possible combinations can be enormous. The service selection problem can be treated as a combinatorial optimization problem, which is known to be NP-hard problem.

3. The Comprehensive Evaluation Model

3.1. Functional Quality

The functional quality $SSD(as_i, s_i)$ represents the semantic similarity degree of abstract service as_i and concrete service s_i. $SSD(as_i, s_i)$ is defined as

$SSD(as_i, s_i) = (ISD(as_i, s_i) + OSD(as_i, s_i)) / 2$

Where ISD ($InputSimilarityDegree$) indicates the semantic similarity degree of the inputs between as_i and s_i; OSD ($OutputSimilarityDegree$) indicates the semantic similarity degree of the outputs. In this paper, we only describe the computing process of ISD. The computing process of OSD is similar to that of ISD.

The computing process of ISD is described as follows:

Input: $InputSet_1$ (the input set of as_i), $InputSet_2$ (the input set of s_i), $w = \{w_1, w_2, w_3, ...w_n\}$.

Output: $InputSimilarityDegree$

```
1. Set $InputSimilarityDegree$ =0;
2. Define $Array[0..n-1]$;
3. For each $Input_i \in InputSet_1$ do {
4.     $Array[i] = FindMaxSimilarity(Input_i, Inp_2)$;
5. }
6. $InputSimilarityDegree = \sum_{i=1}^{n} (w_i \times Array[i])$;
7. Return $InputSimilarityDegree$;
```

3.2. Non-functional Quality

Non-functional properties, that is, Quality of Service (QoS) can be used to measure a service. A service will be associated with a group of QoS criteria from the various aspects. In this paper, to reduce the complexity, we take the QoS including price, response time, availability and reliability into account for each service. A QoS vector of service s is given as follows:

$$QoS(s) = < Q_{price}(s), Q_{time}(s), Q_{availability}(s), Q_{reliability}(s) >$$

Where $Q_{price}(s)$ indicates how much a service requester needs to pay to the provider for invoking the service s, $Q_{time}(s)$ measures the time interval to get the response from the invoked service s, $Q_{availability}(s)$ refers to the probability that the service s is accessible during a certain period of time, and $Q_{reliability}(s)$ indicates the trustworthiness of the invoked service s. The QoS properties can be divided into positive properties and negative properties.

- Positive properties. The values need to reach maximization, such as availability and reliability.
- Negative properties. The values need to reach minimum, such as price and response time.

3.3. Aggregation Functions

In this paper, we only concern the sequential process model. Other models can come down to the sequential model. Assume there are M service processes simultaneously, $M = \{sp_1, sp_2, ...sp_m\}$. For each sp_i, the corresponding semantic similarity degree and QoS aggregation functions are illustrated in Table 1.

Table 1. Aggregation Functions

Aggregation type	Properties	**Aggregation function**
Summation	Semantic similarity degree	$\sum_{j=1}^{n} SSD_{ji}$
	Response Time	$\sum_{j=1}^{n} T_{ji}$
	Price	$\sum_{j=1}^{n} P_{ji}$
Multiplication	Availability	$\prod_{j=1}^{n} A_{ji}$
	reliability	$\prod_{j=1}^{n} R_{ji}$

3.4. Transactional Properties

In this paper, we adopt the universal transactional properties for a service: pivot, retriable and compensatable. Transactional rules which define the possible combinations of service in order to obtain a transactional service process are proposed in [16]. $T(sp_i)$ refers to the transactional properties of service process sp_i. $T(sp_i) \in \{p, c, r, cr\}$.

- $T(sp_i) = p$. As soon as all services of sp_i complete successfully, their effect remains evermore and cannot be undone. If one service of sp_i executes unsuccessfully, then all previously successful service ought to be compensated.
- $T(sp_i) = c$. All services of sp_i should be compensatable. A compensatable service can undo its effects when it is executed successfully.
- $T(sp_i) = r$. All services of sp_i should be retriable. A retriable service can be successfully executed after a limited number of invocations.
- $T(sp_i) = cr$. sp_i is not only compensatable but also retriable.

3.5. Service Association

The problem of service association is divided into two levels: the granularity problem in abstract process level; the service dependency problem in concrete process level.

3.5.1. Multi-granularity in Abstract Process: An abstract service process asp_i is defined as $< S_i, G_i, I_i, P_i >$, where S_i is a set of abstract services involved , G_i is a set of the granularity, I_i is the structure of abstract service process asp_i, P_i is a set of feasible abstract path. For example:

- $S_i = \{AS_i[1], AS_i[2], AS_i[3], AS_i[4], AS_i[5], AS_i[6], AS_i[7]\}$;
- $G_i = \{(AS_i[1], AS_i[2]), (AS_i[3], AS_i[4], AS_i[5])\} = \{AS_i[1,2], AS_i[3,5]\} = \{AS_i[6], AS_i[7]\}$, where $AS_i[6]$ is the functional combination of $AS_i[1]$ and $AS_i[2]$; $AS_i[7]$ is the functional combination of $AS_i[3]$, $AS_i[4]$ and $AS_i[5]$;
- $I_i : AS_i[1] \to AS_i[2] \to AS_i[3] \to AS_i[4] \to AS_i[5]$
- $P_i = \{P_i[1], P_i[2], P_i[3]\}$, where

  $P_i[1]$ indicates feasible abstract path $AS_i[1] \to AS_i[2] \to AS_i[7]$

  $P_i[2]$ indicates feasible abstract path $AS_i[6] \to AS_i[3] \to AS_i[4] \to AS_i[5]$

  $P_i[3]$ indicates feasible abstract path $AS_i[6] \to AS_i[7]$

3.5.2. Service Dependency in Concrete Process: In concrete service process, there may exist some concrete services that are dependent on each other. When selecting a concrete service for one abstract service, we must select a particular concrete service for another abstract service in service process.

A group of dependencies between the concrete services in sp_i are defined as $D_i = \{(S_{ijk}, S_{imn})\}$. (S_{ijk}, S_{imn}) means that if the j^{th} concrete service is selected for abstract service S_k in sp_i, then abstract service S_n in sp_i must select the m^{th} concrete service.

3.6. Comprehensive Evaluation Model

The fitness of several processes being optimized simultaneously is computed as follows:

$$\textit{Maximum} \quad \sum_{i=1}^{m}\left(\sum_{j=1}^{r} w_j \frac{Q_{ij}-Q_{ij}^{\min}}{Q_{ij}^{\max}-Q_{ij}^{\min}}+\sum_{j=r+1}^{p} w_j \frac{Q_{ij}^{\max}-Q_{ij}}{Q_{ij}^{\max}-Q_{ij}^{\min}}+w_s \frac{S_i-S_i^{\min}}{S_i^{\max}-S_i^{\min}}\right) \qquad (1)$$

Subject to

1) Service dependencies are satisfied in each sp_i.

2) Transactional rules are satisfied in each sp_i.

3) $w_j > 0, j = 1...p, w_s > 0, (\sum_{j=1}^{p} w_j) + w_s = 1$

Where m is the number of concurrent processes to be optimized. w_j and w_s are weights assigned by users. Q_{ij} is the value of the j^{th} QoS parameter of process sp_i. $Q_{ij}^{\max}$ is the maximum value of the j^{th} QoS parameter of process asp_i and $Q_{ij}^{\min}$ is the minimum value. S_i is the value of the semantic similarity degree of sp_i. $S_i^{\max}$ is the maximum value of the semantic similarity degree of asp_i and $S_i^{\min}$ is the minimum value.

4. Service Selection Approach based on Hybrid Particle Swarm Optimization (HPSO-SSA)

4.1. Particle Swarm Optimization

Particle swarm optimization (PSO) is an evolution algorithm inspired by bird flocking [18]. First of all, the N particles are initialized with random positions. A particle is attracted by the optimal position found by itself p_{pbest} and the optimal position found by the particle swarm p_{gbest} .During each iteration of PSO, the new position x_{id}^{t+1} and new velocity v_{id}^{t+1} of each particle are computed by the following equations:

$$v_{id}^{t+1} = w \cdot v_{id}^{t} + c_1 \cdot r_1 \cdot (p_{pbest}(t) - x_{id}^{t}) + c_2 \cdot r_2 \cdot (p_{gbest}(t) - x_{id}^{t})$$
$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \qquad (2)$$

where the parameter w is called the inertia weigh, which influences the exploration behavior of the particle swarm; c_1 and c_2 are the cognitive ratio and the social ratio; the random variables r_1 and r_2 are uniformly distributed in the scope of $(0,1)$.

We adopt the linearly varying inertia weight w [19] to enhance the performance of PSO. The corresponding equation is as follows:

$$w = w_{\max} - (w_{\max} - w_{\min})\frac{t}{T}, \qquad (3)$$

Where t is the current iteration number and T is the maximum number of iterations. The maximal and minimal weights $w_{\max}$ and $w_{\min}$ are set to 0.9 and 0.4.

4.2. Coding Strategy

A particle consists of integers, whereby the first integer means the identifier of service process, and the following integers are the concrete services of the process. As show in Figure 2 (a), the particle consists of several processes. The first process consists of 4 services, the second of 5 services, *etc.* $S[1]^2$ which is a concrete service can fulfill $AS[1]$, and $S[1,2]^4$ has coarse granularity as it can fulfill $AS[1]$ and $AS[2]$ altogether.

In Figure 2 (b), the particle consists of two service processes. The fitness value of the particle is computed by the fitness of each process. Moreover, the fitness value of each process is decided by the corresponding several feasible path.

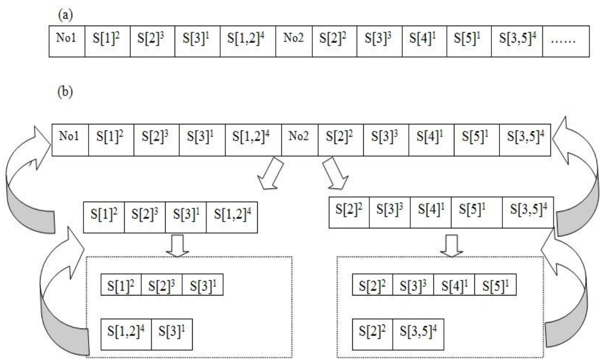

(a)
No1 S[1]² S[2]³ S[3]¹ S[1,2]⁴ No2 S[2]² S[3]³ S[4]¹ S[5]¹ S[3,5]⁴ ......
(b)
No1 S[1]² S[2]³ S[3]¹ S[1,2]⁴ No2 S[2]² S[3]³ S[4]¹ S[5]¹ S[3,5]⁴
S[1]² S[2]³ S[3]¹ S[1,2]⁴
S[2]² S[3]³ S[4]¹ S[5]¹ S[3,5]⁴
S[1]² S[2]³ S[3]¹
S[1,2]⁴ S[3]¹
S[2]² S[3]³ S[4]¹ S[5]¹
S[2]² S[3,5]⁴

Figure 2. Coding Strategy

4.3. Crossover and Mutation

To improve the performance of PSO, we incorporate the crossover and mutation operators into the PSO. Assume $p_j(t)$ is the j^{th} particle in the t iteration. $p_j(t)$ is paired with $p_{pbest}(t)$ and $p_{gbest}(t)$ separately, resulting in an offspring of two new individuals. When producing a new individual m_1, the crossover operator firstly identifies all the concrete processes in $p_j(t)$ that satisfy the constraints, and then copies these concrete processes to m_1. The rest concrete processes in m_1 are copied from $p_{pbest}(t)$. The mutation operator randomly selects a position and replaces the concrete service with an alternative concrete service of the abstract service.

4.4. Algorithm Design of HPSO-SSA

The corresponding algorithm description is as follows:

Step1. Initialize N particles, set the maximum iterations; code the position of the particle. Iteration number *k*=1.

Step2. Check to see if the particle satisfies the constraints. If it is satisfied, compute the fitness value of the particle by equation (1). If it is not the case, assign the worst fitness value for the particle.

Step3. Compare each particle with its p_{pbest}.If it is better than p_{pbest}, then let p_{pbest} be the current particle. Compare each particle with p_{gbest}. If it is better than p_{gbest}, then let p_{gbest} be the current particle.

Step4. Update the velocity and position of the particle with equation (2) and (3). Use the crossover and mutation operations on the particle.

Step5. If termination criteria is satisfied, get the best solution and stop the algorithm; otherwise, k=k+1 and goto step2.

5. Experiment and Analysis

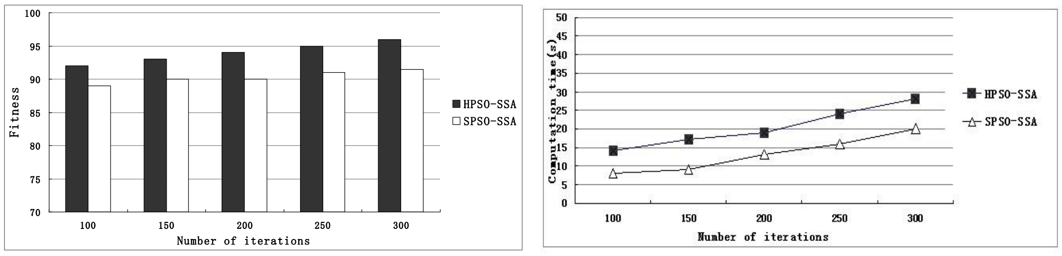
In this section, we conduct simulation experiments to evaluate the efficiency of the proposed algorithms on a PC with Pentium 2.0GHz processor, 4.0GB of RAM and

Windows7 SP1. In the experiment, we construct 15 abstract services and the corresponding set of the granularity. Each service process is generated consisting of up to 8 abstract services. The number of service dependencies in each process is fixed to 10. The number of concrete services for each abstract service is fixed to 80.

Four QoS properties including price, response time, availability and reliability are involved. The QoS properties and semantic similarity degree of every service are randomly generated in a certain range. The transactional property of each service is selected from {*p*, *c*, *r*, *cr*} randomly. This part will compare HPSO-SSA with SPSO-SSA (Service Selection Approach based on Standard PSO) on the number of iterations and the number of concurrent processes.

5.1. Performance vs. Number of iterations

We investigate the fitness value and computation time of HPSO-SSA with SPSO-SSA. The number of service processes to be optimized is fixed to 4. Figure 3(a) shows the fitness of HPSO-SSA with SPSO-SSA. As can be seen, the optimization of the fitness is better for HPSO-SSA than SPSO-SSA. After 300 iterations, HPSO-SSA achieves a fitness of almost 96%, whereas SPSO-SSA only reaches a value of 91%.



(a) Comparison on fitness (b) Comparison on computation time

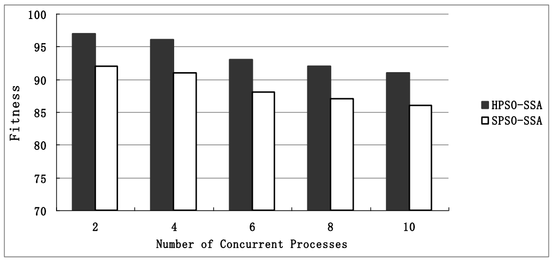
Figure 3. Performance vs. Number of Iterations

Figure 3(b) shows the increase in time for increasing numbers of iterations for SPSO-SSA and HPSO-SSA. It takes SPSO-SSA only about 20 seconds to run the optimization with 300 iterations, and HPSO-SSA needs about 28 seconds to compute. We can see that there is the trade-off between solution quality and computation time.

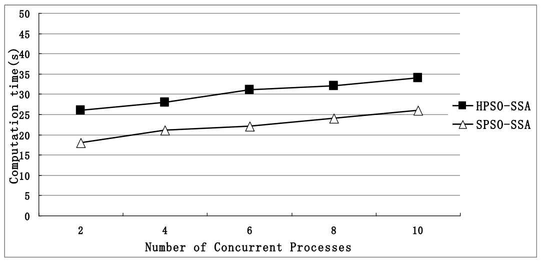
5.2. Performance vs. Number of Concurrent Processes

The fitness value and computation time with increasing numbers of concurrent processes is explored. The numbers of concurrent processes were ranging from 2 to 10 with increases of 2 (iteration was set to 300).

Figure 4(a) shows the fitness of HPSO-SSA and SPSO-SSA. The fitness of both HPSO-SSA and SPSO-SSA decreases with increasing numbers of concurrent processes. For 10 processes, the fitness of SPSO-SSA has decreased to approximately 86%, whereas the fitness of HPSO-SSA is 91%.



(a) Comparison on fitness



(b) Comparison on computation time

Figure 4. Performance vs. Number of Concurrent Processes

Figure 4(b) shows the computation time of all algorithms, the required computation time of the HPSO-SSA increases slowly, which makes our solution more scalable.

6. Conclusion

This paper solves the cloud-based service selection problem from three quality dimensions. We first introduce a comprehensive evaluation model of service process which considers concurrent requests and service association at the same time, and then propose an approach based on hybrid particle swarm optimization algorithm to improve the efficiency of service selection optimization. In our future work, we will research multi-objective evolution algorithms to solve the multi-objective service selection with global optimization. A set of Pareto optimal solutions can be obtained through optimizing diverse objective functions meantime. The recovery cost of the abnormal services in service selection is also left for future research.

Acknowledgements

The research is supported by the Fundamental Research Funds for the Central Universities, and National Key Basic Research Program of China under Grant (No.2010CB328106)

References

[1] A. Bouguettaya, S. Nepal, W. Sherchan, X. Zhou, J. Wu, S. Chen, L. Liu, H. Wang and X. Liu, “End-to-End Service Support for Mashups”, IEEE Transactions on Service Computing, vol. 3, no. 3, **(2010)**, pp. 250-263.
[2] A. Ngu, M. Carlson, Q. Sheng and Hye-young Paik, “Semantic-Based Mashup of Composite Applications”, IEEE Transactions on Service Computing, vol. 3, no. 1, **(2010)**, pp. 2-15.
[3] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam and H. Chang, “QoS-aware Middleware for Web Services Composition”, IEEE Transactions on Software Engineering, vol. 30, no. 5, **(2004)**, pp. 311-327.
[4] M. Alrifai, T. Risse and W. Nejdl, “A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS constraints”, ACM Transactions on the Web, vol. 6, no. 2, **(2012)**.
[5] M. Liu, M. Wang, W. Shen, N. Luo and J. Yan, “A Quality of Service (QoS)-aware Execution Plan Selection Approach for a Service Composition Process”, Future Generation Computer Systems, vol. 28, no. 7, **(2012)**, pp. 1080-1089.
[6] M. Alrifai, D. Skoutas and T. Risse, “Selecting Skyline Services for QoS-based Web Service Composition”, 19th International Conference on World Wide Web, **(2010)** April 26-30, Raleigh NC, USA.
[7] Q. Yu and A. Bouguettaya, “Efficient Service Skyline Computation for Composite Service Selection”, IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 4, **(2013)**, pp. 776-789.
[8] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng and Y. Fan, “An Improved Discrete Immune Optimization Algorithm based on PSO for QoS-driven Web Service Composition”, Applied Soft Computing 12, **(2012)**, pp. 2208-2216.
[9] X. Fan, X. Fang and C. Jiang, “Research on Web Service Selection based on Cooperative Evolution”, Expert Systems with Applications, vol. 38, no. 8, **(2011)**, pp. 9736-9743.

[10] X. Zhao, P. Huang, T. Liu and X. Li, “A Hybrid Clonal Selection Algorithm for Quality of Service-aware Web Service Selection Problem”, International Journal of Innovative Computing, Information and Control, vol. 8, no. 12, **(2012)**, pp. 8527-8544.
[11] X. Wang, Z. Wang and X. Xu, “An Improved Artificial Bee Colony Approach to QoS-Aware Service Selection”, IEEE International Conference on Web Services, **(2013)** June 27-July 2; Santa Clara Marriott, CA, USA.
[12] Y. Syu, Y. FanJiang, J. Kuo and S. Ma, “A Genetic Algorithm with Prioritized Objective Functions for Service Composition”, 26th International Conference on Advanced Information Networking and Applications Workshops, **(2012)** March 26- 29, Fukuoka-shi, Japan.
[13] M. Chen and S. Ludwig, “Fuzzy-guided Genetic Algorithm applied to the Web Service Selection Problem”, IEEE World Congress on Computational Intelligence, **(2012)** June 10-15; Brisbane, Australia.
[14] L. Sun, J. Singh and O. Hussain, “Service Level Agreement (SLA) Assurance for Cloud Services: A Survey from a Transactional Risk Perspective”, 10th International Conference on Advances in Mobile Computing and Multimedia, **(2012)** December 3-5, Bali, Indonesia.
[15] A. M. Hammadi and O. Hussain, “A Framework for SLA Assurance in Cloud Computing”, 26th International Conference on Advanced Information Networking and Applications Workshops, **(2012)** March 26-29, Fukuoka-shi, Japan.
[16] J. E. Haddad and G. Ramirez, “TQoS: Transactional and QoS-aware Selection Algorithm for Automatic Web Service Composition”, IEEE Transactions on Service Computing, vol. 3, no. 1, **(2010)**, pp. 73-85.
[17] E. Blanco, Y. Cardinale, M. Vidal, J. E. Haddad, M. Manouvrier and M. Rukoz, “A Transactional-QoS Driven Approach for Web Service Composition”, Lecture Notes in Computer Science, vol. 6799, **(2012)**, pp. 23-42.
[18] J. Kennedy and R. Eberhart, “Particle Swarm Optimization”, IEEE International Joint Conference on Neural Networks, **(1995)** November 27- December 1, Perth, Western Australia.
[19] Y. Shi and R. Eberhart, “Modified Particle Swarm Optimizer”, IEEE World Congress on Computing Intelligence, **(1998)** May 4-9, Anchorage, AK, USA.

Authors

Wanchun Yang received the M.S. degree in computer science and technology from Guizhou University, Guizhou, China, in 2007. He is currently a Ph.D. candidate in computer science and technology at Tongji University, China. His research interests include service-oriented computing and cloud computing.

Chenxi Zhang received the Ph.D. degree in computer science and technology from National University of Defense Technology, Hunan, China, in 1988. He is currently a professor in computer science and technology at Tongji University, China. His research interests include service-oriented computing and software engineering.