Ant Colony Optimization Based Service flow Scheduling with Various QoS Requirements in Cloud Computing*

Hui Liu
School of Computer Science
Shanghai University
Shanghai City, China

Dong Xu
School of Computer Science
Shanghai University
Shanghai City, China
dxu@shu.edu.cn

HuaiKou Miao
School of Computer Science
Shanghai University
Shanghai City, China

Abstract—**There are a mass of researches on the issue of scheduling in cloud computing, most of them, however, are bout workflow and job scheduling. There are fewer researches on service flow scheduling. Here we propose a model of service flow scheduling with various quality of service (QoS) requirements in cloud computing firstly, then we adopt the use of an ant colony optimization (ACO) algorithm to optimize service flow scheduling. In our model, we use default rate to describe the radio of cloud service provider breaking service level agreement (SLA), and also introduce an SLA monitoring module to monitor the running state of cloud services.**

Keywords-Cloud Computing; ant colony optimization(ACO); Cloud service; Service level agreement; Service flow scheduling

I. INTRODUCTION

Cloud computing [1,2] is service-focused to provide high-quality and low-cost information services by pay-per-use model in which guarantees are offered by the cloud service providers through customized SLA. Cloud computing reduces the investment on hardware, software and professional skills [3]. With the development of cloud computing, the traditional “Information Applications” gradually are converted into “Information Services”. Services have become the focus of research centers and information technology. Cloud computing provides users three kinds of service patterns [4]: Infrastructure as a Service, Platform as a Service, and Software as a Service. Users can acquire various cloud services [5] conveniently via cloud service platforms.

At present most cloud services in the market are provided by large cloud service companies, e.g. Google, Amazon, and IBM. Although cloud services provided by these companies are able to meet current needs, they won’t satisfy users’ requirements gradually as the requirement diversity and the dramatic increase in the number of users. Therefore, with the development of cloud computing, future cloud services may be composed of the third party services all over the world rather than the actual production of these large companies.

Unlike grid computing [6], cloud computing is targeted at ordinary users. Different user has different service requirement and QoS requirement. For example, some users need stored cloud services, some users need computational cloud services, some users need high reliability cloud services, and some users need low-cost cloud services. Facing user’s service request, it usually needs to composite many services to meet different user’s QoS requirements. While there are many cloud services with the same functional properties in cloud computing. How can we schedule cloud services to meet user’s QoS requirements? Therefore, we propose service flow scheduling system model. Service flow is used to describe cloud services’ configuration process, and the reason is that it just likes a flow when configuring cloud services according to the precedence relations of configuration. To ensure that users obtain satisfactory services, we use ant colony optimization algorithm to scheduling service flow.

This paper is organized as follows. Section Ⅱ provides an overview on the current related works. The considered service flow scheduling system model is depicted in section Ⅲ. Section Ⅳ is the corresponding formulations of our model. Section Ⅴ proposes a service flow scheduling approach based on ACO algorithm. In section Ⅵ, some experimental results are presented to show the validity of the approach, before concluding.

II. RELATED WORK

There are many researches on scheduling in grid environment. For conventional applications with independent tasks, some simple heuristics such as Min-Min and Max-Min [7, 8] are used to satisfy the QoS constrains such as on time and /or cost. In paper [9], Suraj Pandey used partical swarm optimization (PSO) to schedule the workflow to reduce the cost of workflow. Cost was also treated as optimization objective in literature [10] and [11]. These researches only focus on single optimization objective, but they don’t have researches on more than one QoS requirements. Literature [12] proposed Multiobjective Differential Evolution(MODE), but the paper doesn’t have a research on multiple QoS requirements. Wei-Neng Chen and Jun Zhang [13] used ACO to solve workflow with various QoS requirements in grid computing. Genetic-

* This work is supported by National Natural Science Foundation of China(NSFC) under grant 60970007 and Shanghai Leading Academic Discipline Project, Project Number: J50103;
Corresponding author Dong Xu, Assoc Prof, E-mail: dxu@shu.edu.cn

based optimization techniques also have been used to tackle scheduling problem in Grid environment, for review, readers are referred to [14, 15, 16]. Although these approaches worked effectively in grid environment, they couldn't be directly applied to solve scheduling problem in cloud computing, because cloud computing is more commercialized than grid computing.

In cloud computing, Meng Xu [17] introduced a Multiple QoS Constrains Scheduling Strategy of Multi-Workflows (MQMW) to tackle workflow scheduling problem. Zhang jun Wu[18] used ACO, GA, and PSO to tackle workflow scheduling problem, the experimental results show that the performance of ACO based scheduling algorithm is better than others. Most researches mentioned above don't introduce SLA in their models.

III. THE MODEL OF SERVICE FLOW SCHEDULING SYSTEM

The model of service flow scheduling system that we proposed is shown in Fig.1. The service flow scheduling system consist of five models: cloud service agency, analysis module, scheduler module, SLA monitoring module, and cloud service repository. Cloud service agency is responsible for publishing cloud services, receiving service requirements, and signing SLA with cloud service provider. Analysis module is to analyze the required service being composited by which kinds of cloud services, then transforming service requirement into service flow. Scheduler module maintains a scheduling algorithm to generate optimal schedules according to user's QoS requirement. SLA monitoring module is to monitor the running condition of cloud services, and feed back the monitoring results to scheduler module. Cloud service repository contains all kinds of cloud services. The process of service flow scheduling in our model can be depicted as:

(1) Cloud service providers submit cloud services to cloud service agency, then cloud service agency will publish these cloud services in cloud service repository.

(2) User submits service requirement specification and QoS requirement document to cloud service agency. Following consultation, user signs SLA [19] with cloud service agency.

(3) Cloud service agency transmits the service requirement specification and QoS requirement document to analysis module. Analysis module transforms the service requirement into service flow and copy cloud services needed for service flow from cloud service repository, and then transmits service flow and the list of cloud services to scheduler module.

(4) Scheduler module generates an optimal schedule according to user's QoS requirement, and then transmits a group of optimal cloud services to Cloud service agency

(5) Cloud service agency negotiates and signs SLA with cloud service providers according to optimal cloud services that generated in scheduler module.

(6) The service flow scheduling system configures service for user according to SLA signed with them.

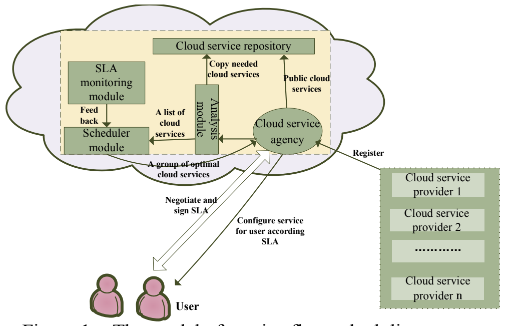


Figure 1. The model of service flow scheduling system

(7) SLA monitoring module monitors the running condition of cloud services, and feed back the monitoring results to scheduler module.

IV. FORMULATION

There are many cloud services which have the same functional properties but different non-functional properties in cloud service repository. We choose reliability, response time, cost, and security as parameters to evaluate cloud services.

Definition 1 Cloud service model. We model cloud service as $S(S.r, S.t, S.c, S.s, S.p)$. $S.r$, $S.t$, $S.c$, and $S.s$ stand for reliability, response time, cost, and security of cloud service, respectively. $S.p$ represents the cloud service provider.

Definition 2 Cloud Service Set. We define cloud service set SS as $SS=\{S_1, S_2, S_3, ..., S_m\}$. S_1, $S_2, S_3, ..., S_m$ are cloud services with the same functional properties but different QOS properties. m is the number of cloud services in cloud service set SS. We call cloud services in the same cloud service set the candidate cloud services.

Definition 3 Service flow. The user required service is composited of n cloud services, we model the service flow SF as a directed acyclic graph (DAG) $SF=(V, A)$, where $V=\{d_1, d_2, ..d_i...d_n\}$ corresponds to cloud service requirement of the service flow. The set of arcs A represents precedence relations to configure cloud service. Two examples of service flow are shown in Figure 2.

In our model, user can define QoS threshold and specify optimized objective in SLA. If the service flow SF consists of n service requirements, and every service requirement d_i in service flow has a selection domain $SS^i=\{S^i_1, S^i_2, S^i_3, ..., S^i_j, ..., S^i_{m_i}\}$, where SS^i is an cloud service set and m_i is the number of candidate cloud services in cloud service set SS^i. Every candidate cloud services in SS^i meets the functional requirement of d_i. Service flow scheduling is to find a schedule that satisfies the QoS constrains below and optimizes the objective that user specified.

1) Reliability Constrain. The reliability of service flow $SF.r$ is not less than $MinReliability$. $MinReliability$ is the minimum reliability threshold that user defined, and S_j^i is the cloud service that allocated to d_i.

$$SF.r = \underset{1 \le i \le n}{Min}\ S_j^i.r \ge MinReliability \tag{1}$$

2) Response Time Constrain. The response time of service flow $SF.t$ is no more than $MaxResponsetime$. $MaxResponsetime$ is the maximum response time threshold that user defined.

$$SF.t = \sum_{i=1}^{n} S_j^i.t * x_i \le MaxResponsetime \tag{2}$$

$$x_i = \begin{cases} 1 & d_i \, is\, on\, the\, critical\ path\, in\, SF's\ DAG \\ 0 & else \end{cases} \tag{3}$$

3) Cost Constrain. The cost of service flow $SF.c$ is no more than $Budget$. $Budget$ is the cost threshold that user defined.

$$SF.c = \sum_{i=1}^{n} S_j^i.c \times Demand_i \le Budget \tag{4}$$

Where $Demand_i$ is the expected time that user use cloud service S_j^i.

4) Security Constrain. The security of service flow $SF.s$ is not less than $MinSecurity$. $MinSecurity$ is the minimum security threshold that user defined.

$$SF.s = \underset{1 \le i \le n}{Min}\ S_j^i.s \ge MinSecurity \tag{5}$$

We define four optimization objectives in our paper.

1) Reliability Optimization. For reliability optimization, the goal of the scheduling algorithm is to find a schedule that satisfies (2) (4) (5) constraints and maximizes the value of $SF.r$.

$$Max(SF.r) \tag{6}$$

2) Response Time Optimization. For response time optimization, the goal of the scheduling algorithm is to find a schedule that satisfies (1) (4) (5) constraints and minimizes the value of $SF.t$.

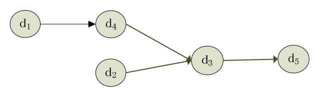
$$Min(SF.t) \tag{7}$$

3) Cost Optimization. For cost optimization, the goal of the scheduling algorithm is to find a schedule that satisfies (1) (2) (5) constraints and minimizes the value of $SF.c$.

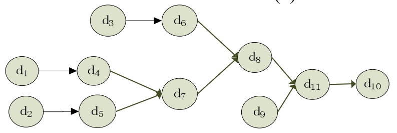
$$Min(SF.c) \tag{8}$$

4) Security Optimization. For security optimization, the goal of the scheduling algorithm is to find a schedule that satisfies (1) (2) (4) constraints and maximizes the value of $SF.s$.

$$Min(SF.s) \tag{9}$$



(a)



(b)

Figure 2. Examples of service flow
(Where there are 5 service requirements in SF1 shown in (a), and 10 in SF 2 shown in (b))

In essence, service flow scheduling problem in cloud computing is to select the highest cost-effective combination from various combinations of service requirements and cloud services.

V. SERVICE FLOW SCHEDULING BASED ANT COLONY SYSTEM

Ant colony optimization (ACO) [20-21] is a new heuristic algorithm which is originated from the foraging behavior of ants in nature. Ant colony system (ACS) algorithm [22] is one of the best ACO algorithms so far [23]. So we use ACS to tackle service flow scheduling problem in cloud computing.

A. Definition of Heuristic Information

An important parameter of ACS is heuristic information which guides the search direction of ants. We define five types of heuristic information.

1) Reliability heuristic information η_R. η_R always bias ants to select candidate cloud services with higher reliability.

$$\eta_R = \frac{S_j^i.r - Min\_reliability^i}{Min\_reliability^i} \tag{10}$$

Where $Min\_reliability^i = Min_{1 \le j \le m_i}\{S_j^i.r\}$. According to (10), the higher the reliability is, the greater η_R is. Equation (10) certifies that the value of η_R is normalized to the interval $[0,1)$.

2) Response time heuristic information η_T. η_T always bias ants to select candidate cloud services with shorter response time.

$$\eta_t = \frac{Max\_responsetime^i - S_j^i.t}{Max\_responsetime^i} \tag{11}$$

Where $Max\_responsetime^i = Max_{1 \le j \le m_i}\{S_j^i.t\}$. According to (11), the shorter the response time is, the greater η_T is. Equation (11) certifies that the value of η_T is normalized to the interval $[0,1)$.

3) Cost heuristic information η_C. η_C always bias ants to select candidate cloud services with lower cost.

$$\eta_c = \frac{Max\_cost^i - S_j^i.c}{Max\_cost^i} \tag{12}$$

Where $Max\_cost^i = Max_{1\le j\le m_i}\{S_j^i.c\}$. According to (12), the lower the cost is, the greater η_C is. Equation (12) certifies that the value of η_C is normalized to the interval $[0,1)$.

4) Security heuristic information η_s. η_s always bias ants to select candidate cloud services with higher security.

$$\eta_s = \frac{S_j^i.s - Min\_security^i}{Min\_security^i} \tag{13}$$

Where $Min\_security^i = Min_{1\le j\le m_i}\{S_j^i.s\}$. According to (13),the higher the security is, the greater η_s is. Equation (13) certifies that the value of η_s is normalized to the interval $[0,1)$.

Definition 4 Default Rate. Default rate is used to represent the ratio that cloud service provider violate SLA. For a cloud service provider A, the formula of default rate is defined as follows:

$$p_{BreakSLA} = \frac{Ten_{BreakSLA}}{Ten_A} \tag{14}$$

Where $Ten_A = \sum_{j=1}^{NumA} Ten_j$, Ten_A is the total number of tenements of cloud service provider A. Ten_j is the number of tenements of the j-th cloud service. $NumA$ is the total number of cloud services provided by cloud service provider A. $Ten_{Break\,SLA}$ is the number of cloud service provider A violating SLA. $p_{BreakSLA} \in [0,1]$.

5) Heuristic information η_{CSP_A} . η_{CSP_A} always bias ants to select candidate cloud services provided by cloud service provider with lower default rate.

$$\eta_{\mathrm{CSP}_A} = 1 - p_{BreakSLA} \tag{15}$$

According to (15), the higher the default rate of cloud service provider is, the greater the heuristic information η_{CSP_A} is. The value of η_{CSP_A} is normalized to the interval $[0,1)$.

B. Services flow scheduling algorithm based ACS

The service flow scheduling algorithm based ACS is depicted as follows (The flowchart of the ACS algorithm is shown in Figure 3).

Step 1 Initialization of service flow sequence. A group of M ants are used in the algorithm. Each ant set out from initial note (such as d_1 in Figure 2) to traverse the DAG of *SF* and builds a sequence of service requirements.

Step 2 Initialization of algorithm. In initial cloud service set (such as cloud service set 1 in Figure 4), randomly select a

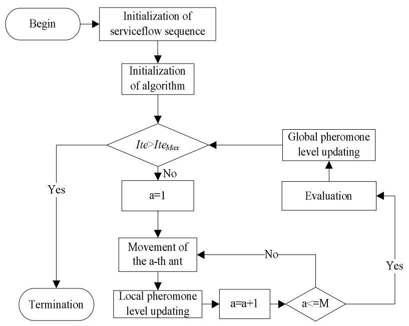

Begin
Initialization of serviceflow sequence
Initialization of algorithm
$Ite > Ite_{Max}$
Global pheromone level updating
Yes
No
a=1
Evaluation
Movement of the a-th ant
No
Yes
Termination
Local pheromone level updating
a=a+1
a<=M

Figure 3. The flowchart of ACS

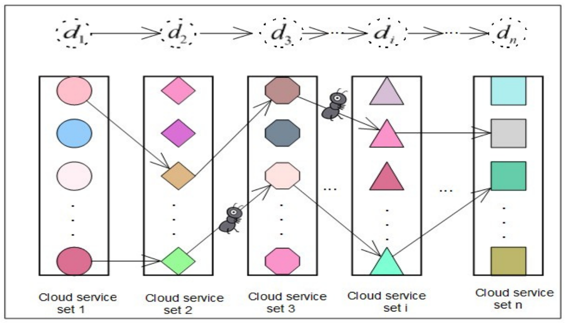


Figure 4. Procedure of ants selecting candidate cloud services （where d_i represents the i-th service requirement. Graphic elements that are in the same shape represent some candidate cloud services with special functional properties, whereas different colors in the graphic elements with the same shape indicate different required QoS properties for the candidate cloud services）

candidate cloud service for each ant. The cycle number Ite is set to be 0. The initial pheromone level between any two candidate cloud services is set to be τ_0, τ_0 is a small positive constant.

Step 3 Movement. According to the service flow sequence that ant has build, each ant moves from one candidate cloud service S_j^{i-1} to another candidate cloud service S_k^i (as shown in Figure 4) according to the following rule.

$$S_k^i = \begin{cases} \arg\max_{1\le k\le m_i}\{\tau_{jk}\eta_{jk}^{\beta}\} & if\ q \le q_0 \\ k & else \end{cases} \tag{16}$$

k is obtained from the following equation.

$$p_{jk} = \frac{\tau_{jk}\eta_{jk}^{\beta}}{\sum_{1\le k\le m_i}\{\tau_{jk}\eta_{ik}^{\beta}\}} \tag{17}$$

Where β is a parameter that determines the relative importance of heuristic information and pheromone level. q_0 is a parameter and $0 \le q_0 \le 1$. q is a random number in interval [0,1]. $\eta_{jk} = \eta_{csp} + \eta_R \times \varphi_1 + \eta_t \times \varphi_2 + \eta_c \times \varphi_3 + \eta_s \times \varphi_4$. In order to guarantee the quality of services, the weight of η_{csp} is set to be 1. φ_1, φ_2, φ_3 and φ_4 are parameters which represent the weight of reliability, response time, cost, and security, respectively. The weight of a property reveals the extent a user concerns about this property. φ_1 , φ_2 , φ_3 and φ_4 are numbers that in interval [0,1] and they satisfy $\varphi_1 + \varphi_2 + \varphi_3 + \varphi_4 = 1$.

Step 4 Local pheromone level updating. Update the pheromone level between two candidate cloud services for each ant as

$$\tau_{jk} = (1-\rho)\tau_{jk} + \rho * \sum_{a=1}^{M} \Delta\tau_{jk}^{a} \qquad (18)$$

$$\Delta\tau_{jk}^{a} = \begin{cases} \tau_0 & \text{if the } a-th \text{ ant pass away } (S_j^{i-1}, S_k^{i}) \\ 0 & \text{else} \end{cases} \qquad (19)$$

Local updating rule will increase the diversity of ACS. In equation (18), ρ is a pheromone decay parameter and $0 < \rho < 1$. For each ant, repeat *Step 3* and *Step 4* until all the ants build their schedule.

Step 5 Evaluation. After all the ants have built their schedule in that iteration, the quality of a schedule can be evaluated by the following equation (20).

$$F(score) = 4 + \left(\begin{aligned} &\varphi_1 * \frac{SF_r - MinReliability}{MinReliability} + \varphi_2 * \frac{Max\,Responsetime - SF_t}{Max\,Responsetime} \\ &+ \varphi_3 * \frac{Budget - SF_c}{Budget} + \varphi_4 * \frac{SF_s - MinSecurity}{MinSecurity} \end{aligned} \right) \qquad (20)$$

According to (20), the function of $F(score)$ is composed of two parts. In the second part, $\varphi_1 * \frac{SF_r - MinReliability}{MinReliability}$, $\varphi_2 * \frac{Max\,Responsetime - SF_t}{Max\,Responsetime}$, $\varphi_3 * \frac{Budget - SF_c}{Budget}$, and $\varphi_4 * \frac{SF_s - MinSecurity}{MinSecurity}$ are formulas that is to evaluate the reliability, response time, cost, and security of service flow, respectively. If the reliability of service flow doesn't satisfy reliability constraint, the value of $\varphi_1 * \frac{SF_r - MinReliability}{MinReliability}$ will be a negative number, and vice versa. In order to normalizing the value of ($SF_r - MinReliability$) to the interval [0,1] , we divide ($SF_r - MinReliability$) by $MinReliability$ .The reason of adding 4 in equation (20) is to avert the negative value of $F(score)$. We can see from (20) that the higher the value of $F(score)$ is, the better the schedule is.

Step 6 Global pheromone level updating. When the best schedule has been selected， update the pheromone level between two candidate cloud services that belongs to the best schedule as

$$\tau_{jk} = (1-\rho)\tau_{ik} + \rho * (|F(score)| + 1) \qquad (21)$$

where S_j^{i-1} and S_k^{i} belongs to the best schedule.

Step 7 Termination. $Ite = Ite + 1$. Move ants to the initial cloud service set and randomly select a candidate cloud service for each ant, then continue *Steps 3-7* until $Ite \le Ite_{\max}$ or all the ants have built the same schedule.

TABLE I. Users' QoS requirements for SF 1 and SF 2

Service flow	QoS			
	MinReliability	*MaxResponsetime* (s)	*Budget* (RMB/month)	*MinSecurity*
SF1	0.85	2.5s	400	0.8
SF2	0.85	2.5s	800	0.8

VI. EXPERIMENTS

We test our approach at a PC. The PC's configuration is Intel® Core™2 Duo CPU E7500 @ 2.93GHz with 4GB RAM. The operating system of the PC is Windows 7. The ant colony system algorithm program was developed by JAVA.

We adopt two service flow (SF1 is shown in Fig.2 (a) and SF2 is shown in Fig.2 (b)) in our experiment. The QoS requirement of service flow is shown in table 1. Our experiments are done using 10 cloud service sets. Each cloud service set contains 10~50 candidate cloud services. The reliability, response time, cost, and security of candidate cloud services are randomly generated. The parameters of ACS are set as: q_0=0.7 , M=50 , $\beta = 2$,and $Ite_{\max} = 50$. We set φ_1=0.7 , φ_2=0.1 , φ_3=0.1 , φ_4=0.1 if user prefers to optimize the reliability of service flow, φ_1=0.1 , φ_2=0.7 , φ_3=0.1 , φ_4=0.1 if user prefers to optimize the response time of service flow, φ_1=0.1 , φ_2=0.1 , φ_3=0.7 , φ_4=0.1 if user prefers to optimize the cost of service flow, and set φ_1=0.1 , φ_2=0.1 , φ_3=0.1 , φ_4=0.7 if user prefers to optimize the security of service flow.

In Fig.5, We can see from (a) that ACS find a schedule that meets all constraints from the 5th iteration and the reliability of service flow is optimized gradually with the increase of iteration time. (c) and (d) are performance of ACS in the case of SF1 and SF2 under cost optimization, respectively. The experimental results show that the ACS performs well in cost optimization. (b) and (e) are performance of ACS under response time and security optimization, respectively. We can see from the two figures that in the ACS the optimum global solution can be gained and the convergence speed is fast. Figure (f) shows that when the number of cloud service sets is maintained constant and the number of candidate cloud services increases, the iteration time increased sharply. That's because the amount of calculation of each ant is increasing when the number of candidate cloud services increases.

VII. CONCLUSION

We propose a service flow scheduling model in cloud computing. In our model, ACO algorithm is designed to schedule service flow with various QoS requirements. We consider different QoS properties of cloud services, including reliability, response time, cost, and security. Then four optimization objectives are designed according to users' preferences. Users are allowed to define QoS thresholds in SLA. In order to ensure the quality of service, we use default

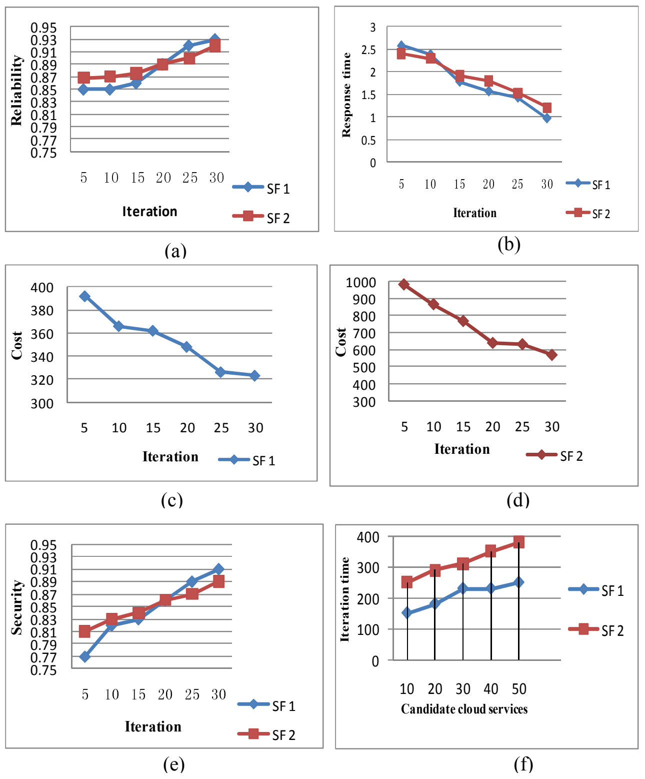


Figure 5. Performance under different optimization objectives (Where (a) is the performance of ACS in the case of SF1 and SF2 under reliability optimization, (b) is the performance of ACS in the case of SF1 and SF2 under response time optimization, (c) is the performance of ACS in the case of SF1 under cost optimization, (d) is the performance of ACS in the case of SF2 under cost optimization, (e) is the performance of ACS in the case of SF1 and SF2 under security optimization, (f) is the relationship of iteration and the number of candidate cloud services in cloud Service Set)

rate to represent the ratio that cloud service provider violate SLA and use SLA monitoring module to monitor the running condition of cloud services. Finally, we tested our algorithm in experiments. The experiments results demonstrate that the approach proposed in this paper is effective.

REFERENCES

[1] Michael Armbrust,Armando Fox,Rean Griffith,Anthony D.Joseph,Randy Katz Andy Konwinski,Gunho Lee,David Patterson,Ariel Rabkin,Ion Stoica,Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley Reliable Adaptive Distributed Systems Laboratory.2009.

[2] Ian Foster,Yong Zhao, Ioan Raicu, Shiyong Lu. Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop. 2008. GCE '0812-16 Nov. 2008. pp.1-10.

[3] Jeremy Geelan. Twenty one experts define cloud computing.Virtualization. August 2008. Electronic Magazine, article available at http://virtualization.sys-con.com/node/612375.

[4] http://developers.sun.com.cn/blog/functionalca/resource/sun_353cloudcomputing_chinese.pdf.

[5] Frank E.Gillett, Eric G.Brown, James Staten, Christina Lee. The New Tech Ecosystems of Cloud, Cloud Services, And Cloud Computing. Forrester Research Report, 2008.

[6] Ian Foster,Yong Zhao, Ioan Raicu, Shiyong Lu. Cloud Computing and Grid Computing 360-Degree Compared . Proc. IEEE Grid Computing Environments Workshop (GCE '08). 2008,pp: 1-10.

[7] D. B. Tracy, J. S.Howard, and B. Noah. Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, vol. 61, no. 6 , 2001, pp. 810 - 837.

[8] J. Yu, R. Buyya. Workflow Scheduling Algorithms for Grid Computing. Metaheuristics for Scheduling in Distributed Computing Environments, F. Xhafa and A. Abraham (eds), ISBN:978-3-540-69260-7, Springer, Berlin, Germany, 2008.

[9] Suraj Pandey1, LinlinWu1, Siddeswara Guru2, Rajkumar Buyya1. A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. Technical Report,CLOUDS-TR-2009-11,Cloud Computing and Distributed Systems laboratory, The University of Melbourne Australia, October,2009.

[10] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos. Scheduling Workflows with Budget Constraints. CoreGRID Workshop on Integrated research in Grid Computing. Technical Report TR-05-22, University of Pisa, Dipartimento Di Informatica, Pisa, Italy, November 2005,pp: 347-357.

[11] J. Yu, R. Buyya, and C. K. Tham. A Cost-based Scheduling of Scientific Workflow Applications on Utility Grids. Proc. of the 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, December 2005, pp: 140-147.

[12] KhaledTalukde, MichaelKirley, Rajkumar Buyya. Multiobjective Differential Evolution for Scheduling Workflow Applications on Global Grids. Concurrency and Computation: Practice and Experience. Wiley Press, New York, USA.21 (13), pp: 1742-1756, 2009.

[13] Wei-Neng Chen, Jun Zhang. An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem with Various QoS Requirements. 2009 IEEE Transactions on systems, Man, And Cyberetics.

[14] J. Yu and R. Buyya. Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms. Scientific Programming Journal, IOS Press, 2006, 14(3-4), pp: 217-230.

[15] Kim S, Weissman JB. A genetic algorithm based approach for scheduling decomposable data Grid applications. ICPP.IEEE Computer Society: Silver Spring, MD, 2004, pp: 406–413.

[16] Ye G, Rao R, Li M. A multiobjective resource scheduling approach based on genetic algorithms in Grid environment. Fifth International Conference on Grid and Cooperative Workshops, Hunan, China, 2006; 504–509.

[17] Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi. A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing. 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications.2009, pp: 629-634.

[18] Zhangjun Wu, Xiao Liu, Zhiwei Ni, Dong Yuan, Yun Yang. A Market-Oriented Hierarchical Scheduling Strategy in Cloud Workflow Systems.Journal of Supercomputing, Special issue on Advances in Network&Parallel Comptg, to be appeared, 2010.

[19] Mohammed Alhamad, Tharam Dillon, Elizabeth Chang.Conceptual SLA framework for Cloud Computing.4th IEEE International Conference on Digital Ecosystems and Technologies. 2010, pp: 606-610.

[20] A. Colorni, M. Dorigo, and V.Maniezzo. Distributed optimization by ant colonies.F.Varela and P.Bourgine (Eds.). Proceedings of the First European Conference on Artificial Life. Paris: Elsevier Publishing,1991: 134-142.

[21] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents . IEEE Transactions on Systems, Man, and Cybernetics-Part B. 1996, 26(1):29-41.

[22] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to TSP. IEEE Trans. Evol. Comput. vol. 1, no. 1, pp. 53–66, Apr. 1997.

[23] M. Dorigo and T. St¨ utzle, Ant Colony Optimization. Cambridge, MA: MIT Press, 2004.