QoS Adaption aware Algorithm for Grid Service Selection

Jingya Zhou, Junzhou Luo, Zhiang Wu
School of Computer Science and Engineering, Southeast University, Nanjing, P.R.China
Key Laboratory of Computer Network and Information Integration Ministry of Education, Nanjing, P.R.China
jyz@seu.edu.cn; jluo@seu.edu.cn; zawu@seu.edu.cn

Abstract

Grid service composition has been recognized as a flexible way for resource sharing and application integration since appearance of service-oriented architecture. Approaches are needed to select service candidates with various Quality of Service (QoS) levels according to use's performance requirements. In this paper We model this problem as the Multi-Constrained Optimal Path Selection Problem (MCOP), due to the dynamic property of grid service, adaptive mechanism is introduced to ensure the whole QoS when some service candidates fail. An algorithm QAGSS is proposed. Simulation results show that QAGSS has greater success rate and lower cost than previous algorithms.

Keywords: Grid, QoS adaptation, Composite service, DAG, MCOP.

1. Introduction

Open Grid Service Architecture (OGSA) [1] as a Service Oriented Architecture (SOA) provides a combination framework, in which physical resources are virtualized to the user in the form of grid services. Grid services use standard interfaces for invoking. Using standard interfaces individual services can be combined into a composite service, by which complicated application can be completed. Since many services have the same functional properties while different non-functional properties, such as QoS, cost etc. and service-oriented grid requires deliver seamless QoS, so algorithms that can rapidly and efficiently select service candidates to form a composite service for applications is needed.

Candidates of composite service and their relationship can be represented by directed acyclic graph (DAG), in which QoS and cost can be seen as weight. Hence the problem above can be modeled as MCOP, which is to select a suitable path satisfying multi-constraints meanwhile minimizing the global cost. Algorithms for solving MCOP are introduced to web services selection problem in many previous. Different from web service, the state of grid service varies dramatically. Existing services may change QoS level, fail or even withdraw at any moment, while new services may join. In this case, adaptive mechanism is worth being considered for QoS level assurance. In this paper, we model the problem as MCOP and an effective algorithm QAGSS is proposed, which takes QoS adaption into account.

The rest of this paper is organized as follows. In Section 2 we will present a brief overview of related work about QoS and service selection. We introduce an approach for QoS parameters standardization in Section 3. The details about algorithm will be narrated and discussed in Section 4. In Section 5 experiment results are presented, and we make a brief analysis. Finally, the paper is concluded in Section 6 and future work is also discussed here.

2. Related work

Foster etc. propose GARA in [2]. GARA supports reservation and adaptation, which support the management of end-to-end QoS in service-oriented grid environment. This is the initial work on grid architecture to support QoS. Rashid Al-Ali proposed a Grid-QoS management framework (G-QoSM) [3], in this framework Rashid etc. classify services into three types based on different QoS levels: guaranteed, controlled load and best effort. Adaption strategies are used to support resource capacity sharing [4].

QoS parameters are classified into five categories and a hierarchical structure of grid QoS is proposed in [5]. The heuristic algorithm based on the structure is confirmed to be effectively by experiment results, but algorithm considered no composite service as well as the cost it takes.

T. Yu etc. study the end-to-end QoS issues of composite service by using a QoS broker in [6]. The problem is modeled in two ways: the combinatorial model defines the problem as a Multi-dimension Multi-choice Knapsack Problem (MMKP) and the graph model defines the problem as a Multi-Constrained Optimal Path Problem(MCOP) and novel algorithms are designed to meet the global QoS constraints while maximize the user defined utility function. These

978 -1-4244-1651-6/08/$25.00 © 2008 IEEE

algorithms can solve the problem by finding near optimal solutions in polynomial time, and it is proved to be suitable for web services, but due to the dynamic nature of grid service, whether it is adapt to grid service is still need to be confirmed.

Composite service selection is modeled as MCOP Problem, for one dimension of QoS, the MCOP Problem is known as NP-complete [10]. To cope with this problem, many pseudo-polynomial-time algorithms such as jaffe's algorithm [8] are proposed, but their complexities depend both on the actual values of the edge and scale of the problem. An efficient heuristic algorithm is introduced in [7] to minimize the nonlinear cost function for finding a feasible path while also incorporating the cost optimization of the selected feasible path, however, the complexity of computation will increase as μ increases, and here μ is an adjustable parameter used to calculate the cost function in algorithm. In this paper we improve this algorithm by introducing upper bound of μ to reduce the computational complexity. It is proved that the time complexity of QAGSS proposed in this paper is the same as that of Dijkstra's algorithm [9].

3. QoS parameter standardization

QoS describes a service's capability to meet consumer's demands. There are many properties to describe QoS, such as concurrent processing capabilities, duration, throughput, reliability, availability, accuracy, security, and so on. The performance of service can be reflected by these parameters from different perspectives, which can be roughly classified into additive and non-additive. For the additive parameters such as duration, throughput it is the sum of the additive parameter value from end to end. In contrast, value with respect to a non-additive parameter, such as bandwidth is determined by the value of that constraint at the bottleneck part. For constraints associated with non-additive parameters, we can simply remove services that do not satisfy these constraints. So in this paper we will mainly discuss additive QoS parameters. The user's QoS requirements may be different with parameters, for example, user may demand delay less than 5ms, while throughput no less than 100, also different service classes may have different quantification standards for the same QoS parameter. Hence, we present an approach to standardize QoS parameters. We assume that *m* is the number of service classes and *n* is the dimension of QoS, $q_k(i)$ represent the *i*th dimension QoS parameter of service class S_i, all QoS parameters are positive, so we get a *n*m* matrix as follow:

$$\begin{pmatrix} q_1(1) & \cdots & q_1(m) \\ \vdots & \ddots & \vdots \\ q_n(1) & \cdots & q_n(m) \end{pmatrix}$$

We roughly put parameters into positive criterion and negative criterion. Positive criterion donates the higher value the higher quality such as throughput, while negative criterion donates the lower value the higher quality such as duration. In this paper two approaches are provided respectively to both criteria.

Translation approach for parameters belongs to positive criterion is shown in equation (1):

$$Q_k(i) = \begin{cases} q_k(i) / \sqrt{\sum_{i=1}^{m} q_k{}^2(i)}, & if \sqrt{\sum_{i=1}^{m} q_k{}^2(i)} \neq 0 \\ 0, & if \sqrt{\sum_{i=1}^{m} q_k{}^2(i)} = 0 \end{cases} \quad (1)$$

Translation approach for parameters belongs to negative criterion is shown in equation (2):

$$Q_k(i) = \begin{cases} \sqrt{\sum_{i=1}^{m} q_k{}^2(i)} / q_k(i), & if \sqrt{\sum_{i=1}^{m} q_k{}^2(i)} \neq 0 \\ 0, & if \sqrt{\sum_{i=1}^{m} q_k{}^2(i)} = 0 \end{cases} \quad (2)$$

We can simply define QoS constraint relationship *r(q, u)*, it is a binary relationship, the first one of the pair indicates service QoS parameter, the target service value user expected is represented by the other. There is only one element in the set of constraint relationship: $R=\{\succeq\}$.

4. QoS Adaption aware Algorithm

4.1. Service composition model

The services we discussed can be divided into various service class based on functionality, a service class is a set of services with common functionality but different non-functional properties such as QoS levels and costs etc. The services requested by a user or application can be classified into two categories — individual service and composite service, the first one means request can be accomplished by a single service class while the other implies request should be completed by a set of service classes. DAG composite service shown in Figure 1 is a general kind of composite service described by DAG, in which node represents service class and directed edge represents collaborative relationship among service classes. DAG composite service has several execution paths, pipeline composite service is one of them, so it can be considered as special case of DAG composite service, and in this paper we discuss DAG composite service for general purpose. Each service class may have many candidate services, and each service has its own QoS level, but users do not care about it, usually they put forward end-to-end QoS requirements. Consequently, the problem need to be solved by service selection is how to select a feasible and optimal path from a mass of candidate ones in Figure 2 to achieve the user's QoS requirements meanwhile minimize costs.

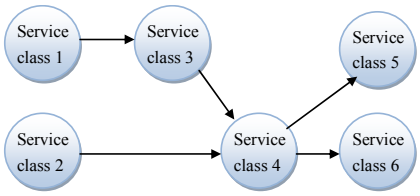

Figure 1. DAG Composite Service

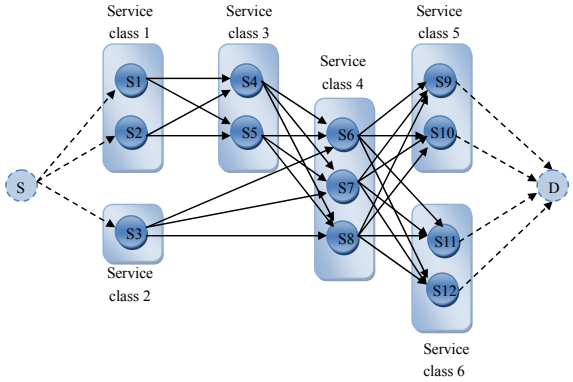

Figure 2. Service Candidate Graph

In Figure 2 we add two nodes—source node and destination node, from source node S we draw edges to all nodes that do not have predecessors, in the same way from all nodes that do not have subsequences we draw edges to destination node D. Each candidate represented by a node has its own QoS parameter values and cost, here we give an approach to move QoS parameter values and costs from nodes to edges, and definition is shown as follows:

Definition 1 Consider a composite service that is represented by DAG graph *G=(V, E)*, where *V* is the set of nodes and *E* is the set of edges. *e(i, j)* donates the directed edge *(si, sj)*, each edge is associated with *n* additive QoS parameters $e_k(i, j)$ and cost *c(i, j)*, *k=1,2,…,n.*

$$e_k(i,j)=\begin{cases}(Q_k(i)+Q_k(j))/2, if\ (si\neq S, sj\neq D)\\ Q_k(i)+Q_k(j)/2, if\ (si=S, sj\neq D)\\ Q_k(i)/2+Q_k(j), if\ (si\neq S, sj=D)\end{cases} \quad (3)$$

$$c(i,j)=\begin{cases}(c(i)+c(j))/2, if\ (si\neq S, sj\neq D)\\ c(i)+c(j)/2, if\ (si=S, sj\neq D)\\ c(i)/2+c(j), if\ (si\neq S, sj=D)\end{cases} \quad (4)$$

in Figure 2 the both nodes S and D are added without QoS parameters and costs, so we can set *n* parameter values and costs of both nodes are zeros. Hence, we model service selection problem as MCOP using $e_k(i, j)$ as QoS parameters *c(i, j)* as costs respectively instead of $Q_k(i)$ and *c(i)*, the definition is given below:

Definition 2 MCOP: Consider a composite service that is represented by DAG graph *G= (V, E)*, where *V* is the set of nodes and *E* is the set of edges. Each edge is associated with *n* additive QoS parameters $e_k(i, j)$ and *c(i, j)*, *k=1,2,…,n.* Given n constraints u_k, *k=1,2,…,n,* the problem is to find a path *p* from source node S to destination node D subject to:

(i) $e_k(p)=\sum_{e(i,j)\in p} e_k(i,j)\geq u_k$ for *k = 1,2,…,n,* and

(ii) $c(p)=\sum_{e(i,j)\in p} c(i,j)$ is minimized over all feasible paths satisfying (i).

For a path *p* the sum of values of nodes that belong to *p* equals to that of edges belong to *p,* and the values include QoS parameter values and costs.

$$c(p)=\sum_{e(i,j)\in p} c(i,j)=\sum_{si\in p} c(i)$$

$$e_k(p)=\sum_{e(i,j)\in p} e_k(i,j)=\sum_{si\in p} Q_k(i)$$

4.2. Introduction to the QAGSS Algorithm

From above definitions, we now present the following cost function $C_\mu(p)$ for the service selection problem:

$$C_\mu(p)=(\frac{u_1}{e_1(p)})^\mu+(\frac{u_2}{e_2(p)})^\mu+\ldots+(\frac{u_n}{e_n(p)})^\mu \quad (5)$$

where $\mu \geq 1$. Then we conclude the following conclusions on the performance of algorithm that return a path *p* by minimizing the cost function (5) for a given $\mu \geq 1$.

Theorem 1 Suppose that there is at least one feasible path exists, and *p* is a path that minimizes the cost function $C_\mu(p)$ for a given $\mu \geq 1$. Then

(i) $u_k \leq e_k(p)$ for at least one *k*

(ii) $e_k(p)\geq \sqrt[\mu]{n}u_k$ for all other *k*'s

(iii) The likelihood of finding a feasible path increases as μ increases.

Proof of Theorem 1in detail can be found in [7]. Now we present our heuristic algorithm QAGSS, which is composed of three parts, one is feasible part, second is optimal part and last is adaptive part. For the feasible part, QAGSS tries to minimize the objective function C_μ for $\mu \geq 1$. It first finds optimal path from each node x to D using function Reverse_Dijkstra [9] with some modifications to relaxation process, then it starts from S and discovers each node based on the minimization of $C_\mu(p)$, where *p* is a complete path passing through node x. This path is determined heuristically at node x by connecting the already traveled segment from S to x and the remaining segment from x to D, and this can be done by calling function Look_Ahead_Dijkstra [9] also with some modifications to relaxation process in [7]. The main QAGSS algorithm is described as follows, and Table 1 shows the description of variables algorithm used.

Main QAGSS Algorithm
Input: G=(V, E), S, D, u
Output: A suitable composite service
1 **Reverse_Dijkstra (G, e, D)**

2 **If** (t[S]>n) **then**
3 **return** failure
4 **End If**
5 $\mu \leftarrow$ MAX_NUM
6 **Look_Ahead_Dijkstra (G, e, c, S)**
7 **If** ($H_k[x] \geq u_k$ for $k \in [1,n]$) **then**
8 **return** suitable composite service S_{sui}
9 **Ada_Set (S_{sui})**
10 **End If**
11 **return** failure

Table 1. Description of variables

Variable	Description
t[x]	The cost of the shortest path from x to D under the cost function $C_\mu(p)$
$T_k[x]$	The individually accumulated edge values along the path
$P_t[x]$	Predecessor of x on optimal path that from x to D
C[x]	The cost of a foreseen complete path via node x based on the cost function $C_\mu(p)$
$H_k[x]$	The individually accumulated edge values along the already traveled segment of the path from S to x
$P_C[x]$	Predecessor of x on the path from S to x
c[x]	The cost along the already traveled segment of the path from S to x
u	The set of u_k, $k \in [1,n]$

There are two directions in the algorithm, backward from S to D and forward from D to S. The backward direction is depicted by lines 1-4 in QAGSS, which estimate the cost of the remaining segment using $\mu = 1$. Reverse_Dijkstra returns a path p from S to D. Before moving to forward direction, QAGSS checks whether t[S]>n or not, if true it means no feasible path exist, which is based on Theorem 1. If not true, path p may be a feasible path. If path p is feasible, we use Look_Ahead_Dijkstra to find a path q with condition $c(q) \leq c(p)$, if not we use it to find a path q with condition $C_\mu(q) \leq C_\mu(p)$, in both cases, we set μ as MAX_NUM that is a constant we set in the algorithm, which could be changed to alter the possibility of finding a feasible path. This is based on Theorem 2 described as follows:

Theorem 2 Suppose Reverse_Dijkstra returns path p, and Look_Ahead_Dijkstra returns path q, then

(i) if p is feasible, q is feasible too and $c(q) \leq c(p)$

(ii) if p is not feasible, $C_\mu(q) \leq C_\mu(p)$.

Proof in detail can also be found in [7], and the following is relaxation process of Reverse_Dijkstra:

Reverse_Dijstra_Relax
Input: Two nodes x, y$\in$V
Output: Predecessor of y on the path
1 **If** ($t[x] > \sum_{k=1}^{n}\left(\frac{u_k}{T_k[y]} + \frac{u_k}{e_k(x,y)}\right)$) **then**
2 $t[x] \leftarrow \sum_{k=1}^{n}\left(\frac{u_k}{T_k[y]} + \frac{u_k}{e_k(x,y)}\right)$
3 $T_k[x] \leftarrow T_k[y] + e_k(x, y)$ for $k \in [1,n]$
4 $P_t[y] \leftarrow x$
5 **End If**

In forward direction, QAGSS invokes Look_Ahead_Dijkstra to identify if there is another path q which possibly improves the performance over path p.

Look_Ahead_Dijkstra_Relax
Input: Two nodes x, y$\in$V
Output: Predecessor of y on the path and accumulated cost
1 Set t as a temporary node
2 c[t] $\leftarrow$ c[x] + c (x, y)
3 $C[t] \leftarrow \sum_{k=1}^{n}\left(\frac{u_k}{H_k[x]} + \frac{u_k}{e_k(x,y)} + \frac{u_k}{T_k[y]}\right)^{\mu}$
4 $H_k[t] \leftarrow H_k[x] + e_k(x, y)$ for $k \in [1,n]$
5 $T_k[t] \leftarrow T_k[y]$ for $k \in [1,n]$
6 **If** (**Select_best (t, y)** = t) **then**
7 c[y] $\leftarrow$ c[t]
8 C[y] $\leftarrow$ C[t]
9 $H_k[y] \leftarrow H_k[t]$ for $k \in [1,n]$
10 $P_C[y] \leftarrow x$
11 **End If**

The above function firstly judge the value of μ, if μ=1 it is no need to compute cost function C_μ, otherwise use C_μ with μ = MAX_NUM to select feasible path. Then we use function Select_best to choose the next node for performance improvement. It selects one of input nodes such that the selected one should minimize cost if foreseen complete path passing through these nodes are feasible, otherwise, it selects one that minimizes the cost function C_μ.

Ada_Set
Input: Suitable composite service S_{sui}
Output: Adaptive service set
SC_i : the ith service class in S_{sui}
S_j : the jth one in l candidates
S_g : the selected candidate in a service class
A_i : the backup service set of SC_i
1 **For** each SC_i in S_{sui}
2 **If** ($S_g \in A_i$) **then**
3 remove S_g from A_i
4 **End If**
5 **If** ($Q_k(g) \geq Q_k(A_i)$ for $k \in [1,n]$) **then**
6 find l candidates$\in SC_i$ subject to $Q_k(j) \geq Q_k(g)$ for $j \in [1,l]$ $k \in [1,n]$ and cost are low
7 **For** each S_j
8 **If** $S_j \neq$ NULL **then**
9 Add S_j to A_i
10 **End If**

11 **End For**
12 **End If**
13 **End For**

For the enhancement of service adaptive capacity, function Ada_Set is designed. It reserves a backup service set for every service class on the path. Whenever a suitable path is found, Ada_Set is called to find l candidates subject to $Q_k(j) \geq Q_k(g)$ for $j \in [1,l]$ $k \in [1,n]$ and cost are low, and add these to A_i. The intention of setting up A_i is to compensate for changing or failure of selected services. When QAGSS find suitable path, it can select candidates from backup service set, after a suitable path is found, Ada_Set check whether a backup service is on the path, if true remove it.

The QAGSS algorithm determines whether there is possibility of path existence by invoking function Reverse_Dijkstra. If true continues to invoke function Look_Ahead_Dijkstra to find a more suitable path. Ada_Set is introduced for assurance of end-to-end QoS in grid environment. Since the modified Dijkstra's algorithm is executed two times at most, the QAGSS algorithm has the same time and space complexities with Dijkstra.

5. Experiment

5.1. Configuration

In this paper we conduct the simulation experiment to estimate the performance of QAGSS algorithm we proposed, and compare it with jaffe's algorithm [8] which is used to solve the MCP problem. MCP problem is a slightly different version of MCOP problem, aims only at finding feasible path that satisfies multi-constraints. For more than two dimensions MCP problem is known to be NP-complete. In [8] the author considers 2-dimension MCP problem, and uses Dijkstra's shortest path algorithm with two adjustable parameters α, β to minimize objective function. Simulation arrangement is described below.

In the simulation, we use a Request_generator to generate user's requests, which is composed of service classes included and corresponding QoS constraints. The number of service classes in each request is classified into two groups [1, 5] and [5, 10]. For brevity, we define that every service class has 10 candidates, QoS dimension is 2 and capacity of backup set is 1. So the number of candidate nodes in DAG also has two ranges [10, 50] and [50, 100]. Each node has 2-dimension QoS parameters and cost information, we use Node_inf_generator to generate these values respectively. Without loss of generality, every dimension QoS parameters are generated by different distributions, and mean of each dimension QoS parameters are calculated. We use these mean instead of edge mean to evaluate the average value of a path, as equation (6) shows: $A_k = (Nsc/2)*m_k$ (6)
A_k represents the kth dimension average value of a path, Nsc is the average of service classes included in a request, so the average length of a path can be represented by expression Nsc/2 and m_k describes the kth dimension QoS parameters mean. The QoS levels of requests have significant effect on the performance. For comparison of performance under different situations, we divided the QoS requests into three groups:

(i) 75% QoS request c_k smaller than average value A_k for $k \in [1,n]$
(ii) 50% QoS request c_k smaller than average value A_k for $k \in [1,n]$
(iii) 25% QoS request c_k smaller than average value A_k for $k \in [1,n]$

In each group, we divide requests into two groups, one with range from 1 to 5 service classes and the other with range from 5 to 10. For each group the simulation is repeated 100 times respectively and the results reported in the subsequent sections are average values.

5.2. Results and Analysis

In Figure 3 the cost of composite service selected by QAGSS is lower than Jaffe's, since QAGSS algorithm requires at most two iterations of Dijkstra's algorithm while jaffe's requires one. The cost of path returned by second iteration is no larger than that of first. Success rate in equation (7) is introduced to make contrast between different algorithms under different situations.

Success rate = number of requests satisfied / total number of requests (7)

From Figure 4 we conclude that success rate improves as MAX_NUM increases, especially among the range from 1 to 10, which is consistent with theorem 1. When MAX_NUM exceeds 10, there is little improvement in success rate. It is due to the nature of the heuristic algorithm, one can expect few anomalies in the general trend. We also observe the smaller the number of service classes included in a request the greater success rate is. Contrast between Jaffe's algorithm and QAGSS algorithm is described in Figure 5. It is obviously that success rate of ours is larger than that of jaffe's. QAGSS execute in two directions, when path returned from backward direction is not feasible, it

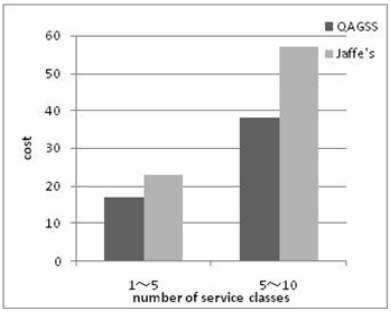

Figure 3. Cost returned by QAGSS and Jaffe's

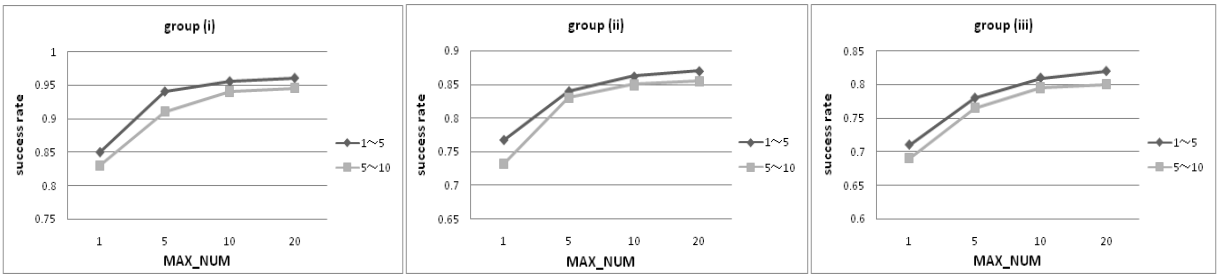

Figure 4. The success rate performance of QAGSS with MAX_NUM

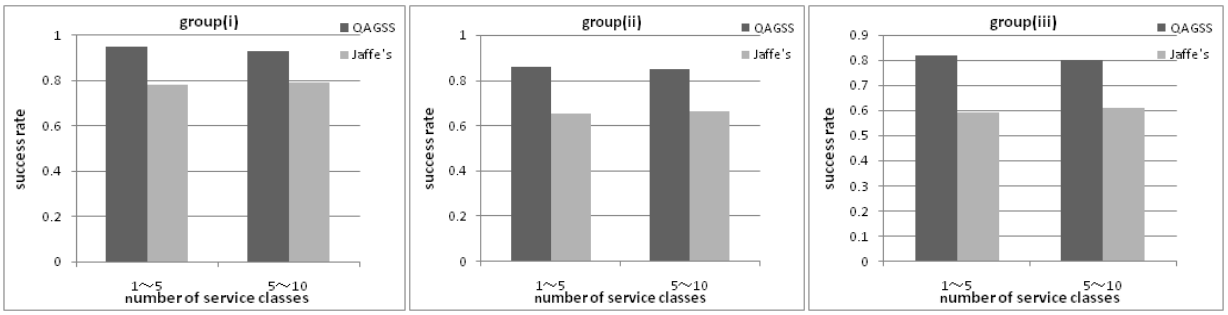

Figure 5. Success rate performance of QAGSS and Jaffe's

will continue to move to forward direction if there exist possibility. According to theorem 2(ii) cost function may be minimized, then increases success possibility.

6. Conclusion and Future work

This paper studies composite service selection problem in grid environment. QoS classification is discussed briefly and approach for standardization is also provided. We model the problem as MCOP, and then QAGSS algorithm is proposed to select least cost composite service while satisfying end-to-end QoS requirements. Adaptive mechanism is considered in QAGSS to adapt to the dynamic characteristic of grid service. In addition, QAGSS has the same time complexity with Dijkstra's, which is suitable for making runtime decisions. Experiment results show that QAGSS performs better among similar algorithms in success rate and cost.

In future, we plan to investigate performances of QAGSS in large-scale distributed grid environment. We will also make some improvements on adaptive mechanism to achieve higher performances and to be more suitable for grid services selection.

Acknowledgement

This work is supported by National Natural Science Foundation of China under Grants No. 90604004 and 60773103, Jiangsu Provincial Natural Science Foundation of China under Grants No. BK2007708 and Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201.

References

[1] I. Foster, C. Kesselman JM Nick, S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. 2002.

[2] I. Foster, A. Roy, and V. Sander. A quality of service architecture that combines resource reservation and application adaptation. *In International Workshop on Quality of Service*, 2000, pp.181–188.

[3] R. Al-Ali, O. Rana, D. Walker, S. Jha, and S. Sohail. G-QoSM: Grid service discovery using QoS properties Computing and Informatics Journal, *Special Issue on Grid Computing*, 21(4), 2002, pp.363–382.

[4] R. Al-Ali, A. ShaikhAli, O. Rana, D. Walker, QoS adaptation in service-oriented girds. *In Proceedings of the 1st International Workshop on Middleware for Grid Computing (MGC2003).*

[5] Zhiang Wu, Junzhou Luo and Aibo Song. QoS-Based Grid Resource Management. *Journal of Software,* Vol.17, No.11, November 2006, pp.2264-2276.

[6] T. Yu, Y. Zhang and K. J. Lin, Effective Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web (TWEB),* Vol. 1, May. 2007.

[7] T. Korkmaz, M. Krunz, Multi-Constrained Optimal Path Selection. *In Proc. of 20th Joint Conf. IEEEComputer & Communications Societies (INFOCOM)*, 2001, pp.834-843.

[8] J. M. Jaffe, Algorithms for finding paths with multiple constraints, *Networks,* vol. 14, 1984, pp. 95–116.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, *The MIT press and McGraw-Hill book company,* sixteenth edition, 1996.

[10] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, *Prentice Hall*, Inc., 1993.