QoS-Based Services Selecting and Optimizing Algorithms on Grid

Qing Zhu, Shan Wang, Guorong Li, Guangqiang Liu, and Xiaoyong Du

School of Information, Renmin University of China, Beijing 100872, P.R. China
Key Laboratory of Data Engineering and Knowledge Engineering
(Renmin University of China), Beijing 100872, P.R. China
zq@ruc.edu.cn

Abstract. QoS-Based Services Selecting and Optimizing Composition between the peers play an increasingly important role to ensure interoperability on Grid environment. However, the prohibitive cost of selecting, matching, mapping and composing algorithm has now become a key bottleneck hindering the deployment of a wide variety of Grid services. In this paper, we present QoS-Based Services Selecting and Optimizing Composition on Grid. First, it checks requesters' semantic in order to form candidate service graph. Second, it designs service selecting and mapping algorithms for optimizing the model. Third, it creates an executed plan of optimum composition on Grid. We conducted experiments to simulate and evaluate our approach.

Keywords: Information Grid, SOA, Service Composition, Selecting Algorithm.

1 Introduction

Grid has been developed to support for solving large-scale problems not only in science, but also in engineering and commerce. Grid supports for the sharing and coordinated use of diverse resources in dynamic, we call distributed virtual organizations.

The complexity of service composition in semantic Grid [3] includes three main factors: (1) different disciplines have different problems, each dependent on different aspects of domain-specific knowledge; (2) new services can be flexibly composed from available service components based on the user's function requirement and quality-of-service (QoS), and (3) both the underlying computing resources and the information input for the process are dynamic. So it's important how to understand precisely meaning of requirement and how to solve a QoS engineering problem since the service selection must select the best services to compose an efficient complex service with QoS assurance.

However QoS-based service composition presents significant challenges and requires addressing a number of critical issues such as discovering and identifying relevant services, formulating semantics, selecting algorithm and creating composition plans using current context, goals, constraints and costs, binding to and invoking composition instances and checking their validity.

In this paper, a key contribution is a dynamic composition model based on semantics[4], graph theory and service selecting algorithm. Candidate Service Generator is a key component of the dynamic composition framework and selects an optimum composition by considering requesters' Quality of Services (QoS). The design of QoS-Based Services Selecting and Optimizing Algorithms is a key improvement to existing Grid computing and runtime services to support the execution of applications.

The rest of the paper is organized as follows. Section 2 describes an overview of system architecture and presents the semantic interpreting according to the user's requirement and QoS-based criterion. Section 3 presents key technology of selecting and mapping algorithms. Section 4 presents simulation and evaluation. Section 5 presents related work. Finally, the paper summarizes and concludes in Section 6.

2 System Architecture

To achieve both flexibility and simplicity, we propose architecture for QoS-Based Services Selecting and Optimizing Composition in Grid Environment. The system is called GridSC (Grid Service Composition System), which can be seen as a middle control layer in an active grid environment that offers a generic service abstraction and automates mapping of processing resources to grid services.

The architecture of GridSC system consists of four components running in two different phases that are semantic Interpreter phase and composition plan generate phase. Grid Service Composition system includes other four key components: Semantic Interpreter, Candidate Service Generator, Prediction Combiner and Plan Execution as shown in Fig 1. Candidate Service Generator, Prediction Combiner and Plan Execution belong to composition plan generate phase.

Semantic Interpreter receives client's request of services, and formally describe service composition specification according to the service interface. In the meantime, semantic interpreter understands and translates the semantic meaning of client's request, then extracts semantics information.

Candidate Service Generator identifies the location and capabilities of processing resources to build a candidate service resource graph that describes the physical network topology. It translates the service specification onto the physical resource graph while taking into account all service-specific constraints and the lists of peers' resource. Finally, it provides common, reusable service candidates to prediction combiner.

Prediction Combiner includes prediction evaluating and service selecting algorithms which needs to select and map a specific service and service level along an optimal path in the execution plan. The selection is based on a user's QoS requirements and Multiple QoS constraints. It provides a new service plan that involves several basic service components through QoS-based services selecting and optimizing algorithms.

Plan execution reserves and allocates appropriate physical resources as determined by the service execution plan and resource lists on globe peers. Once the

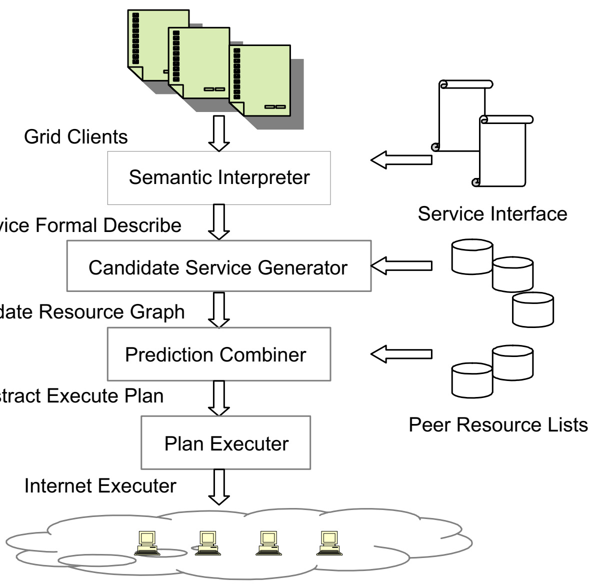


Fig. 1. Service Composition System Architecture

service has been deployed, Plan execution is the final task that maps the execution plan and combines all required components to provide an operational service composition for users. If the available resources are changed, plan execution adopts a proactive approach (e.g. failure recovery, service plan reconstruction) to maintain the quality of composed service during runtime.

From the view of the point, service specification is the foundation of service composition which can describe semantic explanation of user requisition and create requiting goal. Each service owner retains control over the services that they make available to others. They determine how the service is realized and set the policy for accessing the service. It is the key problem that consists of a new service composition QoS-based by using available web services on Grid environment. This paper focuses on the service semantic interpreter and services selecting and optimizing composition algorithms part.

2.1 Semantic Interpreter

The semantic interpreter is critical aspect with respect to service composition which is supported semantic translation, service probe, service selection, composition and monitoring. Automating these steps in Grid service usage life cycle is the aim of semantic Grid services composition. However, the semantic interpreter is the first step of understanding the meaning of service requirement on Grid.

In Grid services domain, semantics represented by the semantic metadata can be classified into the following types, namely, Functional Semantics, Data

Semantics, QoS Semantics and Execution Semantics. These different types of semantics can be used to represent the capabilities, requirements, effects and execution pattern of a Grid service. The semantic interpreter focuses to formalization expression as focused on the functional semantics. Research on Grid services composition on the other hand is based on the findings and results from the semantic Grid research to apply for services that perform some action producing an effect.

The Semantic Interpreter defines the components required by a service and describes how components are interrelated and constrained after user requesting. Lexical and syntactic information is the easiest to represent and process in a parser. But it is not sufficient to resolve all the ambiguities in understanding language of machine; and by its nature, it cannot determine what a sentence means. Therefore, we use Semantic Interpreter that can extract semantics information from Client's Request or interface.

Semantic information allows rich descriptions of Grid services and processes that can be used by computers for automatic processing in various applications. At the meanwhile, the deployment of ontology help understand a well-defined set of common data elements or vocabulary that can support communication across multiple channels, accelerate the flow of information, and meet customer needs.

When user provide m key Requirement Services, semantic interpreter abstracts m keywords, R_1, R_2, ..., R_m and the Requirement function Ψ (R_1, R_2, ..., R_m). Semantic Requirement Function Ψ (R_1, R_2, ..., R_m) is a Boolean function with parameters of key services. The two legal Boolean operations are defined as below: R_i AND R_j: Both keywords R_i and R_j are contained in the answer; R_i OR R_j: Either keyword R_i or R_j is contained in the answer. To handle any of these cases, the semantic interpreter must generate a Composition Candidate Graph that states a logical proposition. Now, let us introduce the formational notions associated with Semantic Requirement Services.

Definition 1. *Semantic Requirement is represented:*
$\Psi(R_1, R_2, ..., R_m) = \{V_t, V_n, Rule, Select\text{-}Sentence\}$,
There are four important components in a formalization description. Where:
1. Terminal symbols $V_t = \{AND, OR, R_i\}$; R_i *: the description of i-th Service Requirement;*
2. Nonterminals $V_n = \{Keyservice, Select\text{-}Sentence, Segment\}$;
3. Start Symbol: Select-Sentence;
4. Productions or Rule consisted of :
Select-Sentence → Segment,
Segment → Keyservice AND Segment,
Segment → Keyservice OR Segment,
Segment → Keyservice,
Keyservice → $R_1|R_2|R_3|\cdots|R_i|\cdots$;

First, the Semantic Interpreter parses user's service request to a set of select keywords and computes the final states once a keyword selects. Just as above described, it receives m (m>0) keywords R_1, R_2, ..., R_m and a Boolean function Ψ (R_1, R_2, ..., R_m) as one whole select command at the start of the user Requirement Services session. Second, Semantic Interpreter translates them into a sequence of effective keywords and does group some or all of them according to the AND/OR semantics included in Boolean function. The groups of keywords are further constructed to Parser Trees. The semantic interpreter uses syntactic rules to generate parse trees and translates those trees into composition candidate service graphs.

2.2 Candidate Service Graph

Grid computing can provide ubiquitous resource and service availability. Grid is defined to be a dynamic and open environment where the availability and state of these services and resources are constantly changing. Therefore, Grid applications are similarly complex, dynamic and heterogeneous. The primary focus of the service composition model presented in this paper is to evaluate and select a specific service to find an optimal path in the execution plan. Prediction Combiner produces Composition execution plans from the pool of available services to satisfy QoS-based defined composition objectives, policies and constraints.

In GridSC system, the user's functional requirements are given in the form of a composition function graph by the semantic interpreter. The function graph consists of required service functions $F_1, \cdots, F_k$ that are connected by dependency links and commutation links. The dependency link indicates that the output of one function is used as the input by its successor.

Definition 2. *Composition Function Graph is defined as a directed weighted graph G= (F, E, C), where, the user's functional requirements F= $\{F_1, F_2, \cdots,$ Fm$\}$ represents the set of $|F|$ nodes, E= $\{e_1, e_2, \cdots, e_n\}$, represents the set of $|E|$ edges, a weight function C= $\{C_1, C_2, \cdots, C_t\}$ is defined represents the set of QoS-based constraints condition, e_i=(F_i, F_j, C) $\in$ E , represents the function relation that function F_j is a direct successor of state F_i by constraints C_m.*

In the model, composition objectives, and composition policies and constraints ($\{C_i\}$) can be dynamically defined as simple semantic information statements. The available service pool is represented as a candidate service graph CSGraph (S, E), where the nodes represents services, S= $\{s_i\}$, in the pool and the links, E= $\{s_i, s_j\}$, can be modeled as possible interaction. GridSC system can discover and locate the service resources to produced a candidate service graph is present in Figure 2.

Definition 3. *Candidate Service Graph is a state graph CSGraph = (S, E), where S represents the set of $|S|$ peers, denoted by composite services S_i, $1 \leq j \leq |S|$, and E represents the set of $|E|$ overlay links, denoted by e_j , $1 \leq j \leq |E|$.*

Definition 4. *Composite Service $S_i = \{s_{i1}, s_{i2}, \cdots, s_{im}\}$, represents the set of $|S_i|$ compound state of service S= $\{S_1, S_2, \cdots, S_k\}$; denoted by s_{ij} ; s_{ij}: represents basic service; S_i is matching with the service function set F_i in Candidate Service Graph CSGraph.*

Definition 5. *Overlay Links* $E=\{e_1, e_2, \cdots, e_t\}$ *represents the set of* $|E|$ *edges;* $e_i=< s_{ij}, s_{km}, C >$*: if valid (service* s_{ij} *and (condition C)) then activation (*s_{km}*).*

Let's give an example. The four viable service compositions are selected from Candidate Service Graph on Fig 3.

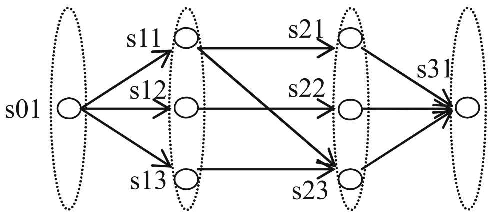
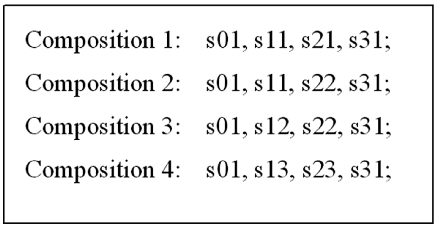


Fig. 2. Candidate Service Graph



Composition 1: s01, s11, s21, s31;
Composition 2: s01, s11, s22, s31;
Composition 3: s01, s12, s22, s31;
Composition 4: s01, s13, s23, s31;

Fig. 3. Service request compositions

Service descriptions are augmented with semantic information in the form of keywords and context information. This semantic information along with QoS-based polices and constraints are used to select applicable services (s_0) and interactions (e_{ij} where Valid (e_{ij}, C_k) = True). Candidate composition plans can be represented as paths in this graph G (S, E). Execution plans may be evaluated and ranked based on different QoS-based cost factors. In the candidate service graph, G(S, E), the available services are vertices and interaction are edges. The edges are created at runtime using a relational join operation. Service Composition can be defined as finding a path from initial to final in G_0 (S, E).

2.3 Execution Plan of Composition

Execution Plan of Service Composition is presented while the QoS-based Services Selecting and Optimizing Algorithms are discussed. We make precise some of the informal arguments and descriptions that we'll meet the terms and conceptions followed in this paper.

Definition 6. *Service Consistency Relation (*$s_i>s_j$*). Give two service components* S_i *,* $S_j \in S$ *in Candidate Service Graph CSGraph (S, E), if Composite Service* S_i*,* S_j *are each matching with Service function* F_i*,* $F_j \in F$*, in Composition Function graph G(F, E), and* $\exists e \in E$ *,* $e=(F_i,F_j,C)$*,and* $In(F_j)=1$ *Valid (Service* S_i *and* S_j*) = True, and Output(*s_i*)*$\supseteq$ *Input(*s_j*), then the relation of service* s_i *and* s_j *is called Service Consistency Relation,denoted by* $s_i>s_j$*.*

Definition 7. *Service Mapping is defined as Composition Function Model G= (F, E) is mapping into Candidate Service Graph CSGraph = (S, E), and the set of Candidate Service* S_i *is matching with Function* F_i *in order to satisfy user's requirements.*

Definition 8. *Execution Planning is defined as the path, denoted as* $P=\{p_1,p_2,\cdots,p_m\}$*, in the Candidate Service Graph CSGraph (S, E),which are selected the basic services to produce Execution Planning of service composition.*

We give the example of service composition, from Function Model graph of services mapping into Candidate Service Graph, and service Execution Planning $P_0,P_1,\cdots,P_n$. Fig 4.

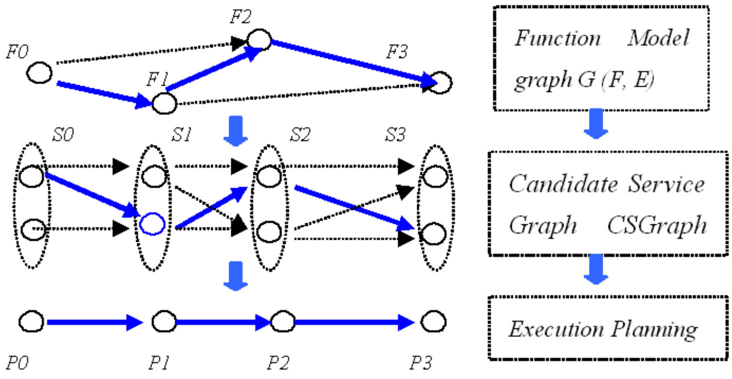

F0
F1
F2
F3
S0
S1
S2
S3
P0
P1
P2
P3
Function Model graph G (F, E)
Candidate Service Graph CSGraph
Execution Planning

Fig. 4. Composition Mapping

The Semantic Interpreter specifies a composition request as a set of constraints, keywords, input service and output services. Candidate Service Generator discovers the participating services, S_i, generate the set of associated interactions E_i, and the composition graph G_i on Candidate Service according to the keyword set and constraint set. In the service selection step, the services in the current service pool are parsed to generate service set S by Prediction Combiner. A relational join operation is then used to construct the set of ad-hoc interactions, E, by matching interfaces, and to create service graph G(S, E). Cost associated with each C_{ij} is calculated and evaluated by Prediction Combiner. Candidate composition Execution plans can now be generated as paths in G between initial and final using graph path algorithms. The composition Execution plans can be ranked based on costs. These costs could reflect QoS-based factors, operational environments and/or user defined factors. Constraints can belong to different categories and can control aspects of both services and compositions. New services can be flexibly composed from available service components based on the user's function and quality-of-service (QoS) requirements.

2.4 QoS Criteria for Composite Services

In GridSC system, a QoS-based approach to service composition is the challenge, which maximizes the QoS of composite service executions by taking into account the constraints and preferences of the users. Traditionally, QoS[2] problem has been studied within different domains such as network layers and multimedia systems. However, we are faced with new challenges in QoS-based service composition because it requires an integrated solution considering multi-dimensional requirements (i.e., function, resource and QoS requirements) at the same time. On the other hand, the QoS of the resulting composite service executions is a determinant factor to ensure customer satisfaction, and different users may have different requirements and preferences regarding QoS. Moreover, the candidate service graph could be a state graph instead of a linear path in order to accommodate parallel execution of services.

In the following, we describe key tasks involved in generic quality criteria for basic services: (1) Execution price (2) Execution duration (3) Availability (4) Successful execution rate (5) Reputation Each basic service may provide different service levels; each level is associated with a QoS vector parameters QoS = ($Q_1,\cdots,Q_n$), for example: Q_1=Q.price, Q_2= Q.duration, Q_3=Q.availability, Q_4=Q.succeed-rate, Q_5=Q.reputation. The quality criteria defined above in the context of basic Grid services, are also used to evaluate the QoS of composite services. If service composition satisfies all the user constraints for that task, it has the maximal score. If there are several services with maximal score, one of them is selected randomly. If no service satisfies the user constraints for a given task, an execution exception will be raised and the system will propose the user to relax these constraints.

3 Selection Algorithm of Service Composition

In this section, we present the service selection algorithms used by the Prediction Combiner for service composition with two or more QoS constraints. We use two algorithms: the SA algorithm and the Heuristic algorithm to solve the problem.

First, we present the QoS[9] quality criteria in the context of basic services, indicate its granularity and provide rules to compute its value for a given service. Second, we assume that the same service definition is used by all basic service candidates for a specific service component on Candidate Service Graph. So we are concerned about the compatibility issue among services and focus on the QoS service selection problem.

3.1 Advanced Simulated Annealing (Advanced-SA) Algorithms

Currently, we have implemented the following more general optimization algorithms in our prototype.

Simulated Annealing (SA): The simulated annealing heuristic is based on the physical process of "annealing". We use the temperature reduction ratio R as a parameter to control the cost/optimality trade-off.

SA is a search technique based on physical process of annealing, which is the thermal process of obtaining low-energy crystalline states of a solid. The temperature is increased to melt solid. If the temperature is slowly decreased, particles of the melted solid arrange themselves locally, in a stable "ground" state of a solid. SA theory states that if temperature is slowed sufficiently slowly, the solid will reach thermal equilibrium, which is an optimal state. By analog, the thermal equilibrium is an optimal task-machine mapping (optimization goal), the temperature is the total completion time of a mapping (cost function), and the change of temperature is the process of mapping change. If the next temperature is higher, which means a worse selecting and mapping, the next state is accepted with certain exponential probability. The acceptance of "worse" state provides a way to escape local optimality which occurs often in service selecting.

3.2 Max-Min Exhaustive Algorithm

Max-min Exhaustive Algorithm (Max-min): This algorithm always yields the actual optimal configuration, but the optimization cost grows rapidly with the problem size. The Max-min heuristic selects a "best" (with minimum completion time) machine for each task. Then, from all tasks, send the one with minimum completion time for execution. The idea is to send a task to the machine which is available earliest and executes the task fastest, but send the task with maximum completion time for execution. This strategy is useful in a situation where completion time for tasks varies significantly.

3.3 Heuristic Greedy Algorithm (HG)

A greedy algorithm means if it builds up a solution in small steps, choosing a decision at each step myopically to optimize some underlying criterion. There are many different greedy algorithms for the different problems. In this paper, we designed Heuristic Greedy Algorithm (HG) to optimize the cost of execution time.

Currently, we have implemented the more general optimization algorithms in our prototype. The algorithms discussed above are suitable under different circumstances. Therefore, for each physical mapping problem, the most appropriate algorithm needs to be selected. Therefore, we propose that the Prediction Combiner should be able to choose the best optimization technique to solve the problems.

4 Simulation and Evaluation

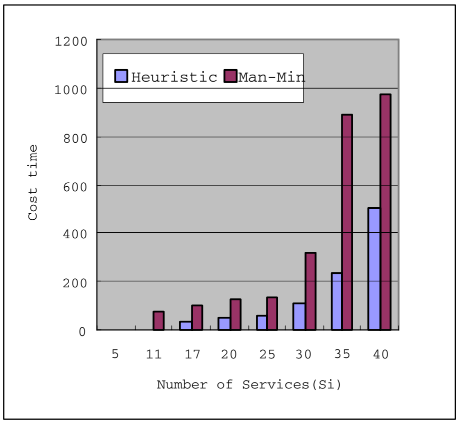
In this section, we will evaluate performance of different parameters on Max-min Exhaustive Algorithm (Max-min), Heuristic Greedy algorithm and Advanced Simulated Annealing (Advanced-SA) Algorithms. Service composition processing time includes three parts (1) Create candidate service graph time: according to user's requirement. (2) Selecting time: executing selecting algorithm to better service. (3) Execution plan time: realizing physical mapping. Our experiments mainly evaluate (2) selecting time, which actually is the major part of service composition processing time.

```
void HeuristicAlgorithm::Select()
Let S be the set of selected service nodes
      For each Si∈S, we store a Qos-based number Q(Si)
Initially S={s} and Q(s) =0
find best Qos-based service from first set of top graph
while S<>V    //For each of no-selecting service set do
      Select a service node v ∉ S with at least one edge from S for which
      CountCost(); min $_{e=(u,v),u\in S}$ Q(u) is as small as possible
      Add v to S and define
      If select[i] success then break;
      If select set finished then exit(0)
endwhile
```

Fig. 5. Heuristic Greedy algorithm

In the following experiments, we assume an equal-degree random graph topology for the 1-8 services of service composition candidate graph. For simplicity, we only consider one process plan with the two service composition algorithms. We produced random numbers of QoS vector parameters, QoS = (Q_1,···,Q_n), for example price, duration, availability, succeed-rate, and reputation. The number of service class and candidates in each service class involved in the process plan range from 5 to 40.

We run our system experiments using the simulation Grid environment on several Pentium(R)4 CPU 2.4GHZ PC with 1GB of RAM. We implement simulation experiment in C++ and both use other development tool.



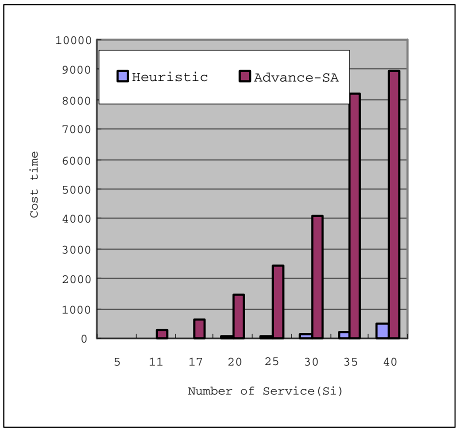


Fig. 6. Cost time of Max-min & Heuristic **Fig. 7.** Cost time of SA & Heuristic

Fig. 6 shows the selecting time with the number of service nodes increasing from 5 to 40. This experiment runs two algorithms: Heuristic Greedy algorithm and Max-min Exhaustive Algorithm. Fig.7 shows the selecting time with the number of service nodes increasing from 5 to 40. This experiment runs two algorithms: Heuristic Greedy algorithm and Advanced Simulated Annealing (Advanced-SA) Algorithms. This experiment shows Heuristic Greedy algorithm is efficient QoS-Based Services Selecting and Optimizing Algorithms about this service composition on Grid.

5 Related Work

Many projects have studied the service composition problem. SpiderNet[1] is the service composition middleware by both user's need for advanced application services and newly emerging computing environments such as smart rooms and peer-to-peer networks. SpiderNet only researched QoS-Assured Service Composition in Managed Service Overlay Networks and no semantic issue has been addressed. The SWORD project [12] and eFlow project [11] proposed a developer toolkit for the web service composition. It uses a rule-based expert system to check whether a composite service can be realized by existing services and generate the execution plan given the functional requirements for the composed application. SWORD only addressed the on-line and no QoS issue has been addressed. Our main contribution is to use semantic knowledge to interpreter user requirement of service composition, to take QoS-driven composition goal into account to find best quality composition by using selecting algorithms on Grid.

6 Conclusion

In this paper, we study the problem of QoS-Based Services Selecting and Optimizing Algorithms on Grid. Two problem Algorithms are proposed: Advanced Simulated Annealing (Advanced-SA) Algorithms, and Heuristic Greedy Algorithm (HG) Algorithms. Semantic Interpreter, Candidate Service Generator, Prediction Combiner and Plan Execution, four components consist GridSC (Grid Service Composition System) system on the Grid environment that offers service composition middle control layer. In the paper we discussed client QoS requirement and QoS constraint. We have presented two algorithms, both optimal and heuristic, to compose and select services QoS-based constraints as well as to achieve the maximum utility.

Acknowledgements

This work is supported by National Natural Science Foundation of China under Grant No. 60473069, Science Foundation of Renmin University of China (No. 30206102.202.307), and the project of China Grid (No. CNGI-04-15-7A).

References

1. Gu, X., Nahrstedt, K., Chang, R.N., Ward, C.: QoS-Assured Service Composition in Managed Service Overlay Networks, In: Proc. of The IEEE 23rd International Conference on Distributed Computing Systems (ICDCS 2003), Providence, Rhode Island, May, pp. 19-22 (2003)
2. Zeng, L., Benatallah, B., Dumas, M.: Quality Driven Web Services Composition[A]. In: Proceedings of the 12th International Conference on World Wide Web (WWW) [C], Budapest,Hungary, pp. 411–421. ACM Press, New York (2003)
3. Zhuge, H.: Semantic Grid. Scientific Issues, Infrastructure, and Methodology, Communications of the ACM 48(4), 117–119 (2005)
4. Berardi, D., Calvanese, D., De Giacomo, G., Hull, R., Mecella, M.: Automatic Composition of Transition-based Semantic Web Services with Messaging. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB, Trondheim, Norway, 2005, pp. 613-624 (2005)
5. Arpinar, I.B., Zhang, R., Aleman, B., Maduko, A.: Ontology-Driven Web Services Composition. IEEE E-Commerce Technology, July 6-9, San Diego, CA (2004)
6. Majithia, S., Walker, D.W., Gray, W.A.: A framework for automated service composition in service-oriented architecture, in 1st European Semantic Web Symposium (2004)
7. Berardi, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M.: Automatic composition of e-services that export their behavior. In: Proc. 1st Int. Conf. on Service Oriented Computing (ICSOC), LNCS, vol. 2910, pp. 43-58 (2003)
8. Carman, M., Serafini, L., Traverso, P.: Web service composition as planning, in proceedings of ICAPS03 International Conference on Automated Planning and Scheduling, Trento, Italy, June 9-13 (2003)
9. Chen, H., Jin, H., Ning, X.: Q-SAC: Toward QoS Optimized Service Automatic Composition. In: Proceedings of 5th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid05), May, pp. 623-630 (2005)
10. Benatullah, B., Dumas, M., Shang, Q.Z., et al.: Declarative composition and peer-to-peer provisioning of dynamic web services[A]. Proceedings of the 18th International Conference on Data Engineering(C).Washington: IEEE, pp. 297-308 (2002)
11. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and dynamic service composition in e-flow. Technical Report, HPL-200039, Software Technology Laboratory, Palo Alto, CA, (March 2000)
12. Ponnekanti, S.R., Fox, A.: Sword: A developer toolkit for Web service composition. In: 11th World Wide Web Conference (Engineering Track), Honolulu, Hawaii, (May 2002)