QoS Aware Service Composition with Multiple Quality Constraints*

Bixin Liu, Quanyuan Wu, Yan Jia, and Bin Zhou

National University of Defense Technology, Changsha, China
{bxliu, qywu, yanjia, binzhou}@nudt.edu.cn

Abstract. Service composition has been recognized as a flexible way for resource sharing and application integration. Quality of service (QoS) is an important issue for composite services. In this paper, we address the issue of component services selection to ensures their composition satisfy given QoS constraints. We propose the concept of reduction tree as a general scheme to aggregate multi-dimensional quality. And then a heuristic algorithm MCSC_HEU is presented to find execution plans satisfying multiple QoS constrains, with the main idea of evaluating partial plans by a heuristic function during the course of reduction. The time complexity of MCSC_HEU is of polynomial level. Extensive evaluations show that MCSC_HEU succeeds in finding feasible plans with very high probability but demands much less time than exhausting search. So it is an efficient solution for QoS aware service selection with multiple constrains.

1 Introduction

Recently web services have been recognized as the next generation framework for building agile distributed applications over the Internet. Applications are provided as web services which can be discovered and composed into more coarse-grained services, called *composite service* [1]. Composite service is usually modeled as a business process build upon it component service. Instead of pre-established relationship between component services in the composition, service oriented computing advocates discovering and binding to services dynamically according to users' requirements on functional aspects as well as non-functional aspects, especially the quality of service (QoS).

Quality of service (QoS) of a web service typically includes a combination of several qualities or properties [2], such as service time, service cost, success rate and etc. Since QoS of a composite service is determined by the QoS of its underlying component services [3], the dynamic nature of composite services offers a good new chance to provide quality guarantee and service level agreement by selecting proper component services according to preferences and quality requirements set by the users. It has a good reason to believe that the quality aware service selection

* This work is supported by the National High-Tech Research and Development Plan of China under Grant No. 2004AA112020, No. 2003AA115210, 2003AA115410 and the National Natural Science Foundation of China under Grant No.90104020.

H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 123 – 131, 2005.

mechanism will be an indispensable part for the QoS management framework for composite services. However, to decide which candidates should be chosen so as to satisfy the global constraints over the composite services is not an easy job, especially when multiple QoS dimensions are considered.

Issues of quality of a business process have been addressed in some earlier work on workflow, among which the METEOR system [5] has given a major contribution. A stochastic workflow reduction algorithm to compute multi-dimensional QoS of workflows has been proposed in METEOR and later extended to web service processes [6]. But service selection problems is absent from their research. Some recent work on web services composition has addressed the issue of QoS aware service selection. An extensible quality model has been proposed and a preference oriented service ranking approach has been presented in [4]. But they only concern selecting the best-qualified service for an activity. Such a local strategy can not handle the global constraints and preferences for the composite services. Limited work has addressed service selection issue in the global or end-to-end sense. A global planning approach based on multi choice decision making and integer programming has been studied in [7], with the objective to maximize the user preference. No domain specific efficient algorithms have been investigated. Similarly, [8] has proposed a utility based approach for service selection to ensure end-to-end response time constrain while maximizing the system benefit and minimizing the overall cost. It is solved by modeling the problem as a multiple choice knapsack problem. A simulated annealing approach for optimizing the performance cost ratio of composite services has been discussed in [9] with the background of grid computing. However the simulated annealing approach is usually not time-efficient.

In this paper, we investigate the issue of quality driven service selection for composite services to ensure multiple global QoS constraints. Compared with other work, the contribution of our research is as follows:

1. We have defined the generic quality-driven service selection problem as multi-constrained service composition (MCSC) problem, which is proved to be an NPC problem.
2. To aggregate the multi-dimensional QoS of composite services, we have explored a reduction based approach and proposed the *reduction tree* as a general QoS aggregation scheme for processes-based application. This concept can be easily extended to various quality metrics and process structures.
3. Based on the concept of reduction tree, we have proposed a heuristic which utilized a non-linear heuristic function to approximate the feasibility of execution plans. Time complexity of the heuristic algorithm MCSC_HEU is polynomial. Evaluations show that MCSC_HEU performs well both in its effectiveness and efficiency.

The rest of this paper is organized as follows. We present firstly the premises of our study and define the MCSC problem in section 2. Then in section 3, we introduce the concept of reduction tree and present the heuristic algorithm MCSC_HEU with its principle and complexity analysis. Extensive evaluations are presented in section 4. At the end, section 5 concludes the paper.

2 Premises and Problem Statement

Firstly, we assume that the composite service model is structured. That is to say the composite service model can be decomposed into substructures recursively according to predefined composition pattern, such as sequence, and-branch and or-branch, until all the substructures are atomic activities. Major composite service modeling languages provide building blocks for structured modeling, such as WSBPEL.

Secondly, general quality metrics are discussed. We consider *n* independent quality metrics $q^{(1)}, q^{(2)} \cdots q^{(n)}$. So the QoS of every candidate service s is represented as a quality vector $q(s) =< q^{(1)}(s), q^{(2)}(s) \cdots q^{(n)}(s) >$. We notice that some QoS metrics could be *negative* (the higher the value, the lower the quality), such as response time and cost, and others could be *positive*, such as success rate. The positive criteria can be converted to equivalent negative one or vice versa by using the reciprocal of its original value. So we assume all the QoS metrics are negative in the following discussion.

Thirdly, it is rational suppose that the quality of a structure in a composite service can be computed by aggregating the quality of its low-level substructures. The aggregation manner usually depends on both quality metrics and composition patterns, which has been discussed in [7,8]. We will not repeat to study specific aggregation rules, however, aggregation functions $f^{(x)}(pat, q_1^{(x)}, q_2^{(x)}, \cdots)$ for metric x are utilized to abstract the aggregation manners, where *pat* is flag of composition pattern , $q_1^{(x)}, q_2^{(x)}$ are the quality in dimension x for two low-level substructures, and $\cdots$ represents other potential parameters such as the execution probability of substructures . We demand that every $f^{(x)}$ is monotonous for all composition patterns: $q_1^{(x)} \geq q_1'^{(x)}$ implies $f^{(x)}(*, q_1^{(x)}, q_2^{(x)}, \cdots) \geq f^{(x)}(*, q_1'^{(x)}, q_2^{(x)}, \cdots)$ and $q_2^{(x)} \geq q_2'^{(x)}$ implies $f^{(x)}(*, q_1^{(x)}, q_2^{(x)}, \cdots) \geq f^{(x)}(*, q_1^{(x)}, q_2'^{(x)}, \cdots)$.

Now we define the problem formally.

Definition 1. Consider a composite service with activity set $A = \{a_1, a_2, ..., a_N\}$ and their corresponding candidate sets $\{S_1, S_2, .., S_N\}$, its *partial execution plan p* is a partial function from *A* to $\bigcup_1^N S_i$ satisfying $p(a_i) \in S_i$ $(i=1..N)$. If $Dom(p)=A$, we say *p* is an *execution plan*.

Because the QoS of composite service is related to specific execution plan, we denote the quality vector of a composite service with respect to (partial) execution plan *p* as $< q^{(1)}(p), q^{(2)}(p) \cdots q^{(n)}(p) >$.

Definition 2. Given a constrain vector $cons =< c^{(1)}, c^{(2)} \cdots c^{(n)} >$, where $c^{(i)} (i = 1..n)$ is a real number, execution plan *p* is said to *satisfy cons* (or *cons* is satisfied by *p*) if for all $i=1..n$ $q^{(i)}(p) \leq c^{(i)}$.

Definition 3. Given a composite service with activity set $A=\{a_1, a_2, .., a_N\}$ and corresponding candidate sets $\{S_1, S_2, .., S_N\}$, the *multi-constrained service composition problem* (MCSC problem) is to find an execution plan *p* that satisfies the given constrain vector *cons*.

Theorem 1. The MCSC problem is NP-Complete.

Theorem 1 can be proved by converting a special case of MCSC problem which concerns sequentially connected activities to a typical scenario considered in the literature of QoS routing. Thus the problem becomes to find a path in a network of *M*N* nodes that satisfies the given constraints, namely Multi-Constrain Path (MCP) problem [10], which has been proved to be NP-complete. So MCSC problem is NP-Complete too. Details are omitted for space limit.

3 Service Selection Algorithm for QoS Aware Service Composition

Before the service selection algorithms is presented, we introduce the concept reduction tree at first which acts as the QoS aggregation scheme for composite service and basic data structure in the algorithms.

3.1 Reduction Tree

Reduction tree is developed based on the reduction concept introduced in [5]. For composite services with well-structured model, once all the substructures' qualities in a structure are known, the quality of their composition can be calculated according to simple pre-defined rules. This procedure is called reduction. Reduction can be carried out repeatedly on two substructures in the reverse direction to decomposition and ends when the top structure, the whole process, is reached. Reduction tree captures the reduction procedure of a composite service.

Definition 4. The *reduction tree (R-Tree)* of a composite service is a binary tree in which the degree of each node is either 0 (leaf node) or 2 (non-leaf node). Leaf nodes represent activities of the process, and non-leaf nodes, annotated with workflow pattern flags denotes a substructure in the process. Every node in the R-Tree is weighted by a real number $prob_u$ which is the probability that the substructure denoted by u is visited if its parent is entered.

Reduction tree can be derived from the composite service model and the execution history of the service. Given the R-Tree of a composite service, we can compute its QoS w.r.t an execution plan by attaching a quality vector $q_u = (q^{(1)}{}_u, q^{(2)}{}_u \cdots q^{(n)}{}_u)$ to every node in the post order. Firstly, vectors of leaf nodes are set respectively to the quality vector of its assignment in the execution plan. Then vectors of those non-leaf nodes are determined by aggregating the vectors of their children according to pre-defined reduction rules. After the procedure is executed at the root, quality vector of the root is the quality of the process.

The left part of Fig.1 depicts an example R-Tree. Flags of "$|$", "$\wedge$"and "$\vee$" in the non-leaf nodes represents sequence, AND-split/join and OR-split/join respectively.

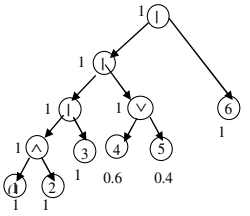

Fig. 1. Example of reduction tree

3.2 Heuristic Algorithm for Service Selection

Based on the concept of R-Tree, an exhausting search algorithm can be developed that every possible pair of partial plans are merged at every non-leaf nodes so that all the possible plans can be generated at the root. This simple algorithm is effective but not efficient enough because of its exponential complexity.

So we propose a heuristic algorithm MCSC_HEU that can solve the problem in polynomial time. The main idea of MCSC_HEU is to evaluate partial plans at non-leaf nodes by a heuristic function instead of comparing every pair of possible partial plans to decide which are the most promising combinations leading to the feasible solutions. The heuristic function to evaluate the favorableness of partial plan p on non-leaf node is defined as follows:

$$h(p) = \prod_{i=1}^{n} q^{(i)}(p) / c^{(i)}$$

Explanation of *h(p)* is given in Fig. 2 which shows a simple case with only two dimensions. The square area ($\mathcal{F}$) represents the feasible region in the 2D space and the black dot represents a partial plan with normalized weight on each dimension. Every partial plan p determines a rectangle ($\mathcal{D}$) with the origin and two axes, filled with bias as shown in the Fig.2. In 2D space, the area of $\mathcal{D}$ indicates the cost of a partial plan with respect to *h(p)*. For n-dimensional cases, $\mathcal{D}$ is a n-dimensional hypercube and *h(p)* represents the volume of the hypercube. So it is reasonable to suppose that the smaller $\mathcal{D}$ is, the better.

Because *g(p)* is an approximation of the likelihood that a partial plan will be extended to a feasible plan, it is inevitable that the heuristic may fail in some cases. To improve performance MCSC_HEU search for the best k partial plans with respect

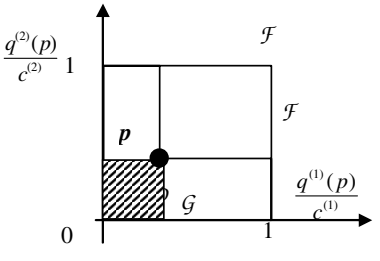

Fig. 2. Explanation of the cost function

MCSC_HEU (*Candidates[1..N],cons*)
FOR *all leaf node i in R-Tree*
FOR *all* s_{ij} *in Candidates[i]*
$P^i.insert(\{<i, s_{ij}>\}, q_{ij})$
FOR *all non-leaf node j in R-Tree*
$P^j = null$
REDUCE (*RTree.root*);
IF *p in* P^{root} *and p<cons* RETURN *p*.

REDUCE (*u*)
IF *u is a leaf node* RETURN
REDUCE (*u.leftchild*);
REDUCE (*u.rightchild*);
FOR *all* p_m *in* P^l *of u.leftchild*
FOR *all* p_n *plan in* P^r *of u.rightchild*
$p_u = p_m.plan \cup p_n.plan$
FOR *every metric x* $q_u^{(x)} = f^{(x)}(flag, q^{(x)}(p_m), q^{(x)}(p_n), \cdots)$;
P^u.insert(p_u)
sort entries in P^u *with increasing order w.r.t g(p)*
keep the first k *entries in* P^u *and delete others*

Fig. 3. Algorithm MCSC_HEU

to the heuristic function instead of the best one. When partial plans are merged at non-leaf nodes in the reduction procedure, *k* plans with minimum *g* are preserved and then contribute to their parents. Others are considered to be not good enough and discarded. The algorithm MCSC_HEU is presented in Fig. 3.

In MCSC_HEU k^2 partial plans are generated at each non-leaf node and the best *k* plans are selected by sorting technique. Because the time complexity for the best sorting algorithm is $o(n\log_2 n)$ where n is the number of items to be sorted, the time complexity of MCSC _HEU is $o(Nk^2\log_2 k)$. Intuitively the efficiency of MCSC_HEU is related to *k*. The bigger *k* is, the less partial plans that may lead to feasible solutions will be dropped. But it is a contradiction that big *k* will increase the complexity. Fortunately, our evaluation experiments in section 4 show that a small *k* (*k*=4 for instance) is enough for MCSC_HEU to find the right solution with very high probability.

4 Evaluations

The goal of experiments is to evaluate how well MCSC_HEU algorithm performs regarding to various conditions. We compare the performance of MCSC_HEU with

the exhausting search algorithm and define the *comparative ratio* as the performance metric which is the ratio that the feasible plan is found by the heuristic and exhausting search.

4.1 Experiments Setup

We study a special case of the 3-dimensional MCSC problem which considers three generic quality metrics: service time, service cost and success rate. Semantics of these quality metrics and their reduction rules can be referred to [5,7].

Experiments are conducted on composite service process templates generated at random. The QoS parameters for candidate services are generated stochastically too. Values of these quality parameters are uniformly distributed in [1,100], [101,200] and [0.1,1.0] respectively. Constraints $(c^{(t)}, c^{(c)}, c^{(r)})$ are randomly generated as follows: $c^{(t)} = coef * \max(q^{(t)}(p_c), q^{(t)}(p_r))$, $c^{(c)} = coef * \max(q^{(c)}(p_t), q^{(c)}(p_r))$, $c^{(r)} = coef * \max(q^{(r)}(p_t), q^{(r)}(p_c))$, where p_t , p_c and p_r are respectively the best plans w.r.t response time, cost and success rate, and $coef \in [0.5,1,5]$ is the adjustable relaxation coefficient that determines the feasible region. It can be understood that the smaller $coef$ is, the less possibly that the satisfying plan exists. The constraints selection scheme guarantees that at least one feasible plan exists when *coef* is over 1.0.

4.2 Effectiveness and Scalability of MCSC_HU

The first experiment investigates effectiveness of the heuristic by changing the scale coefficient *coef* and *k*. We construct process with 10 activities and 5 candidates for each activity. In the experiments *coef* is set initially to be 0.5 and increased in step of 0.05. For every *coef*, the experiment results are collected by running two algorithms 1000 times with random reduction tree topologies, candidates' QoS parameters and corresponding constraints. The experiment are repeated several times with k= 1 to 128. Curves for comparative ratio are depicted in Fig. 4.

An approximate increasing trend with increasing *coef* can be discovered in Fig.4. Due to the heuristic nature of MCSC_HEU, one can expect few anomalies in the general trend. Take the case that k is 4 for example. The comparative ratio is 0.925 when *coef* is 1.0. So we can claim in the case that each non-leaf node maintains the best 4 plans, if only there is a plan satisfying given constraints, the probability that MCSC_HEU can find the solution is about 92%. It increases as the constraints are relaxed. We get nearly 100% comparative ratio when *coef* is 1.5 which tells the fact that the heuristic performs almost as well as the exhausting search if the constraints are relaxed to a moderate level. Experiments with other *k* discover similar trends for comparative ratio in spite of some differences in detailed data.

Fig.4 also shows that the comparative ratio is closely relevant to the value of *k*. Generally speaking the bigger *k* is adopted, the better performance we get. For example, the comparative ratio is 87.8% when k =1 and *coef* =1.0 while it goes up to 98.6% when k =128 and *coef* =1.0. However we noticed that the performance improvement achieved by increasing *k* is distinct when *k* is below 8. It becomes unattractive if *k* increases to 16 or more although we can expect that the curve will

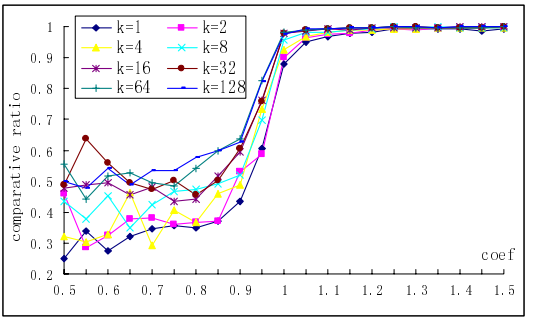

Fig. 4. Performance of MCSC_HEU

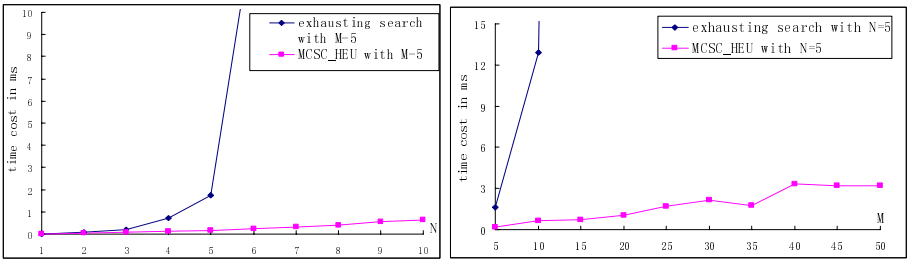

Fig. 5. Time cost comparison of exhausting search and MCSC_HEU

overlap that for the exhausting search algorithm if is k is set to be infinite. This inspires us that we should take a moderate table size in application to get satisfying performance and, at the same time, limit the complexity to a moderate level.

To test the time cost of MCSC_HEU regarding the scale of the problem, we set *coef* =1.2 and change the values of N and M. Fig. 5 shows the result. The curve for the exhausting search approach exhibits obvious exponential increase in its time cost while the curve for the heuristic algorithm indicates approximate linear increase with increasing N. Curves for different M also shows that MCSC_HEU demands much less time than the exhausting search does. When the problem scales, the advantage of MCSC_HEU in the execution time becomes more notable.

5 Conclusions

The emergence of web services has created unanticipated opportunities for establishing agile distributed applications by composing services dynamically to provide new functionality. We have addressed the issue of QoS aware service composition in this paper and raised the MCSC problem aiming at selecting proper component services to ensure their composition satisfy specified QoS constraints. Based on the concept of reduction tree, a heuristic service selection algorithm MCSC_HEU has been developed to solve the MCSC problem with polynomial time complexity. Experiments show that the heuristic performs well both in its

effectiveness and efficiency. Furthermore, the approach presented is general and can be easily extended to deal with much wider scenarios.

References

1. B. Benatallah, M. Dumas, M.-C. Fauvet, F.A. Rabhi, Quan Z. Sheng. Overview of Some Patterns for Architecting and Managing Composite Web Services. ACM SIGecom Exchanges, Vol. 3, No. 3, (August 2002), Pages 9-16
2. Daniel A.Menascé, QoS Issues in Web Services, IEEE INTERNET COMPUTIN, NOVEMBER • DECEMBER 2002, Published by the IEEE Computer Society
3. Daniel A.Menascé ,Composing Web Services: A QoS View, IEEE INTERNET COMPUTIN, NOVEMBER • DECEMBE,R 2004, Published by the IEEE Computer Society.
4. Yutu Liu,Anne H.H. Ngu,Liangzhao Zeng，QoS Computation and Policing in Dynamic Web Service Selection, WWW2004, New York, New York, USA.
5. Cardoso, J., A. Sheth and J. Miller. Workflow Quality of Service. International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02), Valencia, Spain, Kluwer Publishers.
6. Jorge Cardoso, Amit Sheth, John Miller, Jonathan Arnold, and Krys Kochut，Quality of Service for Workflows and Web Service Processes，Journal of Web Semantics, 2004
7. Liangzhao Zeng, Boualem Benatallah,Anne H.H. Ngu, et. al, QoS-Aware Middleware for Web Services Composition, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 5, MAY 2004
8. Tao Yu, Kwei-Jay Lin, Service Selection Algorithms for Web Services with End-to-end QoS Constaints, in Proc. of the IEEE International Conference on E-Commerce Technology, 2004
9. H. Jin, H.H. Cheng, Z.P. Lu,X.M. Ning.Qos Optimizing Model and Solving for Composite Service in CGSP Job Mananger. Chinese Journal of Computers, Apr.2005, Vol 28. No.4. P578-588
10. Yuan, X., Liu, X. Heuristic algorithms for multi-constrained quality of service routing, In Proceedings of the IEEE INFOCOM 2001. Piscataway, NJ: IEEE Communication Society, 2001. 844~853.