Non-Cooperative Game Based QoS-Aware Web Services Composition Approach for Concurrent Tasks

Haifeng Li, IEEE member
School of Geosciences and Info-Physics, School of Civil Engineering, Central South University, Changsha, China
lihaifeng@csu.edu.cn

Qing Zhu
State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China
Zhuq66@263.net

Yiqiang Ouyang
Department of Urban and Regional Planning College of Design, Construction and Planning, University of Florida,USA
yqouyang@gmail.com

***Abstract*—Web services make tools which used to be merely accessible to the specialist available to all, and permitting previous manual data processing and analysis tasks to be automated. One of key problem is Web services composition in terms of Quality of Service (QoS). There are many task concurrencies, such as remote sensing image processing, in computation-intensive scientific applications. However, existing Web service optimal combination approaches are mainly focused on single tasks by using "selfish" behavior to pursue optimal solutions. This causes conflicts because many concurrent tasks are competing for limited optimal resources, and the reducing of service quality in services. Based on the best reply function of quantified task conflicts and game theory, this paper establishes a mathematical model to depict the competitive relationship between multitasks and Web service under QoS constraints and it guarantees that every task can obtain optimal utility services considering other task combination strategies. Moreover, an iterative algorithm to reach the Nash equilibrium is also proposed. Theory and experimental analysis show the approach has a fine convergence property, and can considerably enhance the actual utility of all tasks when compared with existing Web services combinatorial methods. The proposed approach provides a new path for QoS-aware Web service with optimal combinations for concurrent tasks.**

Keywords; Web service combination; QoS; Non-Cooperative Game; Nash equilibrium

I. INTRODUCTION

Web services "have a transformative effect on scientific communities, making tools which used to be merely accessible to the specialist available to all, and permitting previous manual data processing and analysis tasks to be automated" [1, 2]. Web services optimization composition in terms of Quality of Service (QoS) is a key problem[3-5].

Concurrency of a large number of tasks often exists in Web services-based applications, especially in crisis-orientated management. However, when every task "selfishly" seeks for the optimum solution without considering the performance of the entire service system, such methods will result in conflicts between tasks because numerous concurrent tasks will compete for limited optimal resources. This means that many tasks will be assigned to the same optimal service at the same time, which results in the degeneration of processing service capability, and cause service quality decline in all service chains[6, 7]. Each service must deal with different tasks under concurrency; thus, these tasks form a waiting queue, and response time is not only influenced by the process ability of the process service itself, but also by the task load. Moreover, the complicated construction of service chain control flow makes the calculation of QoS aggregation value of service chain, particularly in terms of response time, much harder.

Unfortunately, existing QoS-aware service composition methods based on optimization theory [3] and pursuing performance optimization (e.g., time, price, stability, and so on) under user QoS constraint [8] are unable to solve the problems mentioned above. For instance, the local QoS optimal composition methods [9, 10] select an optimal service via "greedy" means; the global QoS optimal composition method [11-14], which comprehensively considers the QoS model, takes the QoS-aware service composition problem to be a mixed integer linear programming; the re-planning methods consider the dynamic change of QoS and guarantee the optimal performance of the service chain execution under the dynamic environment through re-planning mechanisms [12, 15-17].

Aimed at solving the conflict of optimal resources resulting from the competition of concurrent tasks in time sensitive applications, a non-cooperative game method is proposed in this paper. Our method assigns tasks to each service node in a balanced manner, decrease the conflict caused by competition for optimal resources, and allow each task to receive the highest performance. The contributions of this paper are:

First, we modelled the competition for optimal Web services as a non-cooperative game in which each task composites concrete services to obtain the best utility according to the service composition strategies of the other tasks. To our best knowledge, this is the first model that considers the Web service composition problem from the view of the entire system (all tasks) instead of a tradition single task approach to address the issue of competition amongst the best Web services using game theory under a simultaneous task situation.

Then, the control flow characteristics of Web combination service in terms of the features of multitask concurrency are analyzed, and the QoS aggregation model of multitask concurrency and the evaluation model of task integration utility are advanced.

Lastly, the optimal service combination issue under competition is analyzed by using non-cooperative game theory. Moreover, a mathematical model which depicts the

978-0-7695-4463-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ICWS.2011.45

competitive relationship between multitasks and Web services under the QoS constraints is established through using game theory and the best reply function of quantified task conflicts as bases. Likewise, the study puts forward an iterative algorithm which reaches the Nash equilibrium on such basis.

Further content organization is represented as follows: the second section discusses the non-cooperative game theory of multitask web service optimal combination; the third section describes iterative algorithm of best reply; the fourth section provides experimental results and analysis; the fifth section illustrates relevant work, and the sixth section summarizes this paper, and forecasts the further work.

II. NON-COOPERATIVE GAME THEORY OF OPTIMAL WEB SERVICE COMBINATION FOR CONCURRENT TASKS

The primary consideration here is a phenomenon in which many concurrent tasks require a service chain of the same function. For example, in a huge disaster, many different departments have to implement disaster assessment through a service chain with identical functions (Figure 1). If every task takes the "selfish" optimal strategy, and maximizes its own utility without regard for selections of other tasks, then each task will select the optimal resource (heavy line, Figure 1). This will result in the performance degradation problem of optimal resources, which takes away from the service chain performance of all tasks.

Optimal Web service combination of concurrent tasks can be seen as a non-cooperative game process in which all tasks are dynamically adjusted according to the strategy of other tasks. Each task continuously readjusts its strategy according to the combination strategy of other tasks, and finally reaches the Nash equilibrium[18]. Under the equilibrium state, each task can attain the widest utility with considering the combination strategy of other tasks. This also guarantees the optimization of task performance. As a consequence, we propose the optimal multi-task Web service combination method based on the non-cooperative game theory.

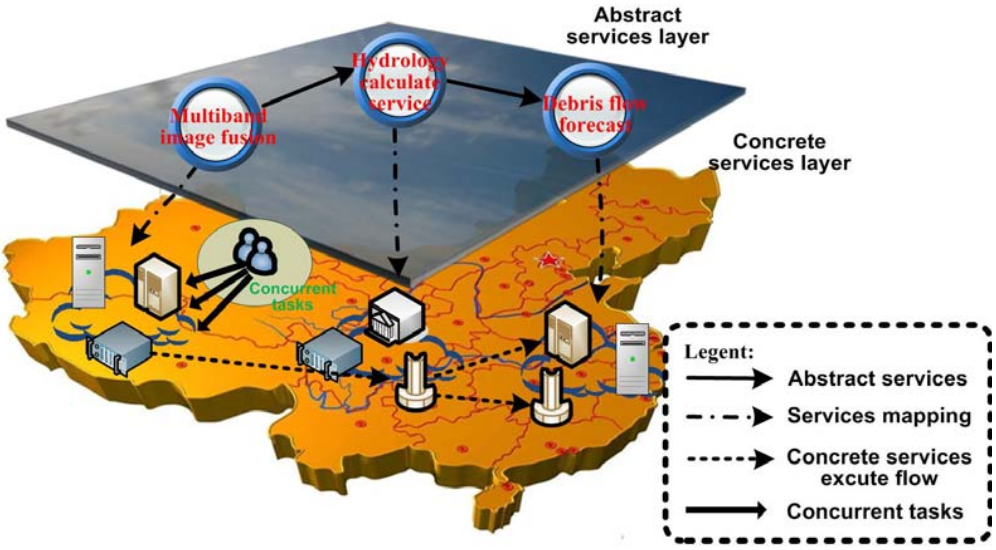

Figure 1. Sample Web service chain of concurrent tasks.

Definition 1 the non-cooperative game model of optimal service combination: is assembled by players and formed by each player's strategy space and utility function, where:

(1) Players: I types of continuous tasks exist and every type of task has the same QoS constraints such as cost and response time. Moreover, the arrival ratio of every task i follows the Poisson distribution with rate $\lambda_i, i = 1, \cdots, I$. A services chain is composed of the abstract service collection $\mathcal{A} = \{a_1, \cdots a_L\}$[17], and every abstract service a_l is comprised of J concrete services with the same functions, but different QoS, written as $\mathcal{C}_l = \{c_1^l, \cdots, c_j^l\}$ [17].. Suppose the processing time of each concrete service follows the exponential distribution with the even speed μ_j; every concrete service in the service chain can be described as $M/M/1$ queuing system[19].

(2) Strategy: Let $s_{i,j}^l$ represent the ratio of task i allocated to GI service j in step l, which is the serial numbers of service in service chain. Vector $s_i^l = (s_{i,1}^l, \cdots, s_{i,J}^l)$ represents the service selection strategy of task i in step l, and based on this, the service selection strategy of task i is $s_i = (s_i^1, \cdots, s_i^l)$; $s = (s_1, \cdots, s_I)$, the compositional strategy of all tasks, is called the combination strategy of optimal service combination game.

(3) Utility function: $\mathcal{U}_i(s)$ stands for the expected performance executed by every task on service selection. In this paper, we define task i to select s rather than s', if and only if $\mathcal{U}_i(s) \prec \mathcal{U}_i\left(s'\right)$.

The essence of the model is that the combination strategy of each task is the best reply to that of other tasks. As a

consequence, we can use the best reply to define the competitive relationship between concurrent tasks: task i's best reply to strategy combination s_{-i} is $s_i^* \in s$, which means task i's other strategy$s_i$will not be more than s_i^* in utility[20].

$$\mathcal{U}_i\left(s_i^*, s_{-i}^*\right) \prec \mathcal{U}_i\left(s_i, s_{-i}^*\right) \quad (1)$$

$s_{-i} = (s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_n)$ represents the strategy combination vector of all players, excluding player i.

Next, we need to define the computing method of every task$i$'s utility function under multitask circumstances.

A. *Service chain structure model and its QoS computing method under concurrent tasks*

Under multitask concurrency condition, tasks may have to wait in the service execution queue. Thus, the computing method of the response time is also different. The queue model of $M/M/1$ (Figure 2) is applied to estimate the response time of processing services. The cost and availability can be regarded as uninfluenced by the continual concurrent tasks; accordingly, we can utilize the same aggregation algorithm as in single task conditions[12].In the $M/M/1$ queue model, each task i's arrival time follows the exponential distribution with the speed of λ_i, and each processed service time follows the exponential distribution with the parameter of μ. Therefore, each processing service time is $W = 1/(\mu - \lambda_i)$[19].

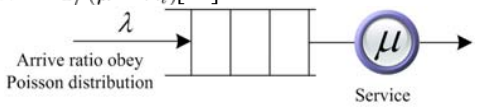

Figure 2. $M/M/1$ queue model.

To calculate the aggregation response time of the service chain, analysis should be carried out in sequential, parallel, branch, and loop structures:

Sequential structure represents the L abstract services implemented by order;

Parallel structure: the M branched abstract services must be simultaneously carried;

Branch structure: there are N branches, among which every branch n is selected according to the possibility b_n.

Loop structure: the abstract services will be re-implemented for K times;

(1) Sequential structure. In the queue network, according to Burke's theorem[21], for the $M/M/1$ queue with arriving rate of λ, its output is also a Poisson process with the rate of λ. That is to say, for all the processing services of the sequential structure, the arrival and departure processes follow the Poisson distribution. As a result, the computing method of the response time of the sequential structure is $W = \sum_{l=1}^{L} 1/(\mu^l - \lambda)$, where L represents the total length of the sequential structure, and l indicates the index of steps in the sequential structure.

(2) Parallel structure. In the parallel structure, every branch needs to be implemented, Therefore, the arrival rate of every branch task is λ based on Burke's theorem[21]. The departure process of each branch also follows the Poisson distribution with the rate of λ. Furthermore, the total response time is determined by the longest parallel branch, i.e., the key path in parallel structure. As a result, to solve the response time of a parallel structure, the key path should first be solved, then the parallel structures serialized. Hence, the computing method of the response time of the parallel structure is $W = \max_M \sum_{l=1}^{L} 1/(\mu_m^l - \lambda)$, where k_p represents the number of parallel branches, and m indicates the index of parallel structure branches.

(3) Branch structure. Branch structure describes the possibility of execution route being selected, if there are N branches, and every branch n's possibility of being chosen is b_n, the sum of which satisfies $\sum_{n=1}^{N} b_n = 1$. Accordingly, the arrival rate of every branch task is $b_n\lambda$. Based on Burke's theorem, the departure of every branch follows the Poisson distribution with the rate of $b_n\lambda$. Tasks are allocated to different branches with different possibilities in the branch structure; thus, we can still use the serialization method to calculate the response time which can be computed as $W = \sum_{n=1}^{N} \sum_{l=1}^{L} 1/(\mu_n^l - b_n\lambda)$.

(4) Loop structure. Different from the loop peeling[6] and unfolding[10] methods, we consider the loop structure as the execution feedback to the queue model. Even if it does not obey Poisson distribution inside the loop structure, the behaviors of internal processing services can still be independent as $M/M/1$ because of the feedback. Thus the loop structure is still a part of the $M/M/1$ queue network. In accordance with Jackson theorem, presuming the internal arrival rate of the loop structure is r, the feedback possibility is $1 - p$, consequently:

$r = \lambda + (1 - p)\, r$, so $r = \lambda/p$

The response time in the loop structure is $W = \sum_{l=1}^{L} 1/\mu_l - \lambda/p = \sum_{l=1}^{L} p/(p\mu_l - \lambda)$.

Under the condition of multitask concurrency, every processing service needs to simultaneously deal with different tasks. Thus, for every processing, the arrival quantity of actual tasks is $\sum_{m=1}^{I} s_{m,j}^l \lambda_m$, and the entire expected response time of task i is the aggregation formulas of every QoS dimension. When it comes to cost and availability, the computing methods are the same with [12].

Sequential structure: $\sum_{l \in S} \sum_{j=1}^{J} \frac{s_{i,j}^l}{\left(\mu_j^l - \sum_{m=1}^{I} s_{m,j}^l \lambda_m\right)}$

Parallel structure: $\max_{cp \in P} \sum_{l \in cp} \sum_{j=1}^{J} \frac{s_{i,j}^l}{\left(\mu_j^l - \sum_{m=1}^{I} s_{m,j}^l \lambda_m\right)}$

Branch structure: $\sum_{k_n=1}^{k_n} \sum_{l \in C_{k_n}} \sum_{j=1}^{J} \frac{s_{i,j}^l}{\left(\mu_j^l - b_{k_n} \sum_{m=1}^{I} s_{m,j}^l \lambda_m\right)}$

Loop structure: $\sum_{l \in C} \sum_{j=1}^{J} \frac{s_{i,j}^l}{\left(p\mu_j^l - \sum_{m=1}^{I} s_{m,j}^l \lambda_m\right)}$

B. *The computing model of task utility*

In this paper, the cost, availability, and response time are included in the QoS model, which is a representative dimension denoting accumulation, multiplication, and extreme value aggregate theorem[22, 23]. Without loss of generality, the paper primarily considers the abovementioned

three representative dimensions as the QoS computing indicators, written as q_1, q_2, and q_3.

QoS indicators have differences in value range and dimension, thus, we first need to normalize QoS indicators.

Different tasks have different preferences over the QoS dimensions; hence, the different aggregate QoS utility values with the simple additive weighting method. As a result, for any task i, the utility function can be expressed as $\mathcal{U}_i(s) = \sum_{h=1}^{3} w_h \mathcal{U}_i^h$, where w_h represents different QoS dimensions with different weights; and $\mathcal{U}_i^h$ states the QoS aggregation utility of the h dimension in the i task

$$v_i^1 = \frac{\mathcal{U}_i^1 - \min q_i^1}{NF_i^1}, v_i^2 = \frac{\max q_i^2 - \mathcal{U}_i^2}{NF_i^2}, v_i^3 = \frac{\mathcal{U}_i^3 - \min q_i^3}{NF_i^3} \quad (2)$$

Where $\mathcal{U}_i^1 = F_i(\mathcal{L})$ indicates the aggregation function of normal cost, $\mathcal{U}_i^2 = AV_i(\mathcal{L})$ represents the aggregation function of normal availability, and $\mathcal{U}_i^3 = R_i(\mathcal{L})$ states the aggregation function of normal response time. $h = \{1,2,3\}$, representing the QoS dimensional index; for example, $h = 1$ indicates the cost; $\max q_i^h$ and $\min q^h$ represents the maximum aggregation value and the minimum aggregation value of the h QoS dimension in the i task, respectively. If $\max q_i^h$ and $\min q^h$ are equal to each other, then it represents the QoS dimension values of the services equal to each other inside the service chain, thus being assigned as 1. $\max q_i^h - \min q_i^h$ is the normalized factor written as NF_i^h. To decide every QoS, the maximum and minimum values of the dimension do not need to be gone over specific services in every abstract service, but must be selected only the maximum or minimal value inside. As a result, the normalization process can be achieved in the polynomial time[12].

III. ITERATION ALGORITHMS WITH BEST REPLY

Based on the non-cooperative game with optimal service combination, the mathematical model of the best reply of a single task is first established, and then the best reply of multitask iteration algorithm is used to solve the non-cooperative game model of service combination.

A. The best reply of a single task

First, we establish the quantitative model for the competitive relationship between concurrent tasks by maintaining the computing ability of the services.

Definition 2: The retained computing ability (RCA)[24] is the available processing ability in the j concrete service in the a^l abstract service of task i:

$$\mu_{i,-j}^l = \mu_j^l - \sum_{m=1,m\neq i}^{I} \kappa s_{m,j}^l \lambda_m \quad (3)$$

Obviously, tasks will impact each other by using the process ability of Web services. The RCA illustrates how many abilities of the Web service remain when a task is assigned to this service. It is thus natural to question how the task chooses the optimal strategy under this condition.

Definition 3: Considering the RCA, the optimal problem $\mathcal{OP}_i$ of every task is

$$\min \mathcal{U}_i(s) = \sum_{h=1}^{3} w_i^h v_i^h \quad (4)$$

$$s.t\, s_{i,j}^l \geqslant 0, \forall l \in L, i = 1,\cdots,I, j = 1,\cdots,J \quad (5)$$

$$\sum_{j=1}^{J} s_{i,j}^l = 1, \forall l \in L,\ \sum_{i=1}^{I} s_{i,j}^l \lambda_i \leqslant \mu_j, \forall l \in L \quad (6)$$

$$\frac{(-1)^{h+1}\mathcal{U}_i^h}{\lambda_i} \leqslant (-1)^{h+1} D_h\ \forall h = 1,2,3 \quad (7)$$

(5) describes non-negative conditions. It represents every processing server is allocated with non-negative tasks; (6) describes conservation conditions. It represents every task i is allocated to processing services and states stable conditions, and shows that any arrival rate is smaller than the biggest service rate to guarantee that the system does not "explode" because of queuing; (7) explains the constraint conditions between task and cost, as well as availability and response time, where D_h, $h = 1,2,3$ separately represents, task i's average cost, average availability, and average response time constraint conditions.

Definition 4: Best reply (BR) strategy of every task, is the solution of $\mathcal{OP}_i$.

Objective and constraint functions are second-order derivable, which constantly have second-order derivatives greater than or equal to 0. Therefore, $\mathcal{OP}_i$ is a convex programming problem. First-order Karush-Kuhn-Tucker's (KKT) condition is necessary and a sufficient condition for the $\mathcal{OP}_i$ solution exists. The LaGrange function is

$$\begin{aligned} &L\left(s_{i,1}^1,\cdots,s_{i,J}^1,\cdots,s_{i,J}^L,\alpha^1,\cdots\alpha^L,\beta_1^1,\cdots,\beta_J^1,\cdots,\beta_J^L,\gamma^1,\cdots,\gamma^h\right) \\ &= \sum_{h=1}^{3} w_i^h v_i^h + \sum_{l=1}^{L} \alpha^l \left(\sum_{j=1}^{J} s_{i,j}^l - 1\right) + \sum_{l=1}^{L}\sum_{j=1}^{J} \beta_j^l s_{i,j}^l + \sum_{h=1}^{3} \gamma^h \left(\frac{\mathcal{U}_i^h}{\lambda_i} - D_h\right) \end{aligned} \quad (8)$$

According to KKT condition, $\alpha^l, \gamma^h \geqslant 0, \beta_j^l \geqslant 0$ for any $l = 1\cdots L$, $j = 1\cdots J$, $h = 1,2,3$, the necessary and sufficient condition between $s_{i,j}^l$ and the solution of $\mathcal{OP}_i$is:

$$\frac{\partial L}{\partial s_{i,j}^l} = 0, \frac{\partial L}{\partial \alpha^l} = 0, \beta_j^l s_{i,j}^l = 0 \quad (9)$$

$$\gamma^h \left(\frac{(-1)^{h+1}\mathcal{U}_i^h}{\lambda_i} - (-1)^{h+1} D_h\right) = 0 \quad (10)$$

Where constraint condition (6) is the assumed condition of the entire system stability; it can be considered as always true; therefore

$$\mathcal{O}_j^l + \mathcal{P}_j^l + \mathcal{Q}_j^l = 0 \quad (11)$$

$$\sum_{j=1}^{J} s_{i,j}^l = 1,\ \beta_j^l s_{i,j}^l = 0 \quad (12)$$

$$\gamma^h \left(\frac{(-1)^{h+1}\mathcal{U}_i^h}{\lambda_i} - (-1)^{h+1} D_h\right) = 0 \quad (13)$$

$$l = 1 \cdots L, j = 1 \cdots J, h = 1, 2, 3 \quad (14)$$

Where:

$$\mathcal{O}_j^l = \frac{w_i^1 \cdot \delta 1_{i,j}^l}{NF_i^1} - \frac{w_i^2 \cdot \delta 2_{i,j}^l}{NF_i^2} + \frac{w_i^3}{NF_i^3} \frac{\mu_{i,-j}^l}{\left(\mu_{i,-j}^l - \delta 3_{i,j}^l s_{i,j}^l\right)^2} \quad (15)$$

$$\mathcal{P}_j^l = \alpha^l + \beta_{i,j}^l \quad (16)$$

$$\mathcal{Q}_j^l = \gamma^1 \delta 1_{i,j}^l - \gamma^2 \delta 2_{i,j}^l + \frac{\gamma^3 \mu_{i,-j}^l}{\left(\mu_{i,-j}^l - \delta 3_{i,j}^l s_{i,j}^l\right)^2} \quad (17)$$

$$\delta 1_{i,j}^l = \kappa f_j^l \lambda_i, \delta 2_{i,j}^l = \kappa \lambda_i \ln\left(a v_j^l\right), \delta 3_{i,j}^l = \kappa \lambda_i \quad (18)$$

The abovementioned equations form nonlinear equations, and by solving the equations, we can obtain the best reply combination strategy $\bar{s}_i^{(x)}$ in any iterations of task i.

B. Best reply iteration algorithm of concurrent tasks

The core concept behind using a best reply-based algorithm to solve the Nash equilibrium is that every task resets its own combination strategy according to other strategies. This process is iterated until it converges to Nash equilibrium. For instance, the first task uses the original value to get the combination $\bar{s}_1^{(1)}$, where the superscript 1 is the iteration number. Then, the second task obtains the optimal combination strategy $\bar{s}_2^{(1)}$ in accordance with the combination strategy of the previous task, that is, the best reply of task 2 compared with that of other tasks. The process is continued until the last task gains its corresponding composition strategy $\bar{s}_I^{(1)}$ according to the previous $I-1$ task's combination conditions. Subsequently, the second round of iteration is implemented; this time, the first task will update its own combination strategy based on the combination strategies of all other tasks. The steps can be called a basic progressive system. Based on the abovementioned best reply of a single task, we design the ESS-based best reply iteration algorithm of the multitasks[25] (Figure 3).

$\bar{s}_i^{(x)}$ // composition strategy of task i at the xth iteration
$\mathcal{U}_i^{(x)}$ // utility value of task i at the xth iteration
x // the number of iteration
$norm$ // L^1 norm at the xth iteration
$function\ Iterative\,(task\ i)$
    $x \leftarrow 0, norm \leftarrow 1$
    $\bar{s}_i^{(0)} \leftarrow 0, \mathcal{U}_i^{(0)} \leftarrow 0$
    $while\,(true)$
        // get the global value of x and sum
        $getGlobeInformation\,(x, sum)$
        $norm \leftarrow sum$
        $if\,(norm < \varepsilon)$
            $exit\ while$
        $\mathcal{BR}Iterative\,()$
        $x \leftarrow x + 1$
$end\ function$
$function\ \mathcal{BR}Iterative\,()$
    $sum \leftarrow 0$
    $for\,(each\ task\ i\ in\ task\ set\ \mathcal{T})\ (i = 1, \cdots, I)$
        $for\,(each\ step\ l\ in\ services\ chain)$
            $for\,(each\ concrete\ service\ j\ in\ step)$
            // calculate residual services capability $\mu_{i,-j}^l$ for each services
            $\mu_{i,-j}^l = \mu_j^l - \sum_{m=1,m\neq i}^{I} \kappa s_{m,j}^l \lambda_m$
        // calculate composition strategy of best reply for each task at the xth iteration
        $\bar{s}_i^{(x)} \leftarrow \mathcal{BR}$
        $sum \leftarrow sum + \left|\mathcal{U}_i^{(x-1)} - \mathcal{U}_i^{(x)}\right|$
    $updateGlobeInformation\,(x, sum)$
$end\ function$

Figure 3. Multitask best reply's iteration algorithm, in which $norm = \sum_{i=1}^{I} \left|\mathcal{U}_i^{(x-1)} - \mathcal{U}_i^{(x)}\right|$, and ε is system acceptance tolerance.

IV. EXPERIMENT AND ANALYSIS EVALUATION

A. Simulation environment

To test the efficiency of optimal Web service combination non-cooperative game method, a simulation experiment was conducted. First, the abstract service chain (Figure 1) was simulated, in which every abstract service includes 10 specific services. Specific services are modeled as M/M/1 processing queue system, and every queue implements services based on a "first come, first served" principle. Afterwards, every QoS indicator value of a service was randomly created, and all of the values conform to normal distribution. Based on such conditions, the non-cooperative game method of optimal Web service combination was evaluated.

B. Performance Evaluation

In a bid to evaluate the non-cooperative game method performance of optimal Web service combination, two classic methods were compared:

Proportional scheme (PS) [26] is a classic distribution method employing task combination strategy, in which servers with higher processing ability acquire more tasks according to the processing ability of servers and based on the ratio distribution task equilibrium algorithm. In accordance with the comprehensive QoS indicator of every service, tasks are distributed, and the weighed utility value of every QoS indicator is assigned as the combination standard. Services with smaller comprehensive utility value are allocated with more tasks.

Mixed integer linear programming-based (MILP) [8, 12] method is a typical single-task service combination algorithm with the basic characteristics of optimizing the combination strategy of every task without regard for the combination strategy of other task. This method considers the normalized weight utility of QoS indicator as the optimal objective function to satisfy user's global optimal solutions of QoS demand constraint conditions.

The performance evaluation of the non-cooperative game method of optimal Web service combination's was

conducted in terms of three aspects: algorithm convergence, task utility, and fairness of task distribution.

*1) **Algorithm convergence.*** One basic issue of the best reply iteration algorithm is whether it can reach the Nash equilibrium; i.e., whether the algorithm can achieve convergence. The combination strategy of each task was intially set as 0. The combination strategy of each task is redesigned according to the orderly combination strategy of other tasks. The algorithm regards the combination strategy of 0 as the initial point which can be considered as any point because any given point can be easily converted to point 0. The experiment demonstrates that the best reply iteration algorithm has good convergence without dependence on the chosen value of the initial point. The iteration algorithm can converge to the Nash equilibrium from any point.

In the iteration process, by adding the number of tasks and changing the tolerable error norm to change the system state, we test the effect of these factors on the convergence of the algorithm. The method is the number of tasks ranges from 10 to 35, and the average arrival rate of every kind is $\lambda_i = 10$, and the tolerable error norm is from 0.1 reduced to 1.0×10^{-5}. As the tolerable error norm increases, the iteration number of algorithm increases accordingly when the quantity of tasks is constant. Similarly, the iteration number of the algorithm increases as the number of tasks increases when the tolerable error norm changes.

During the process of algorithm convergence, every task constantly adjusted its own combination strategy according to strategies of the other tasks; the load condition of every server also fluctuated regularly with changes in the combination strategies of the tasks and finally converged to a relatively fixed value. Figure 4 shows the load change of a concrete service in the iterative convergence process. For instance, a service may be presented few tasks to choose from because of its relatively poor initial performance (wave trough in Figure 4). In the second round of iteration, however, its relative performance becomes improved because no task was allocated; the service may cause load concentration (wave crest in Figure 4). The fluctuation of the service load became exceedingly smaller with each iteration process and ultimately converged at a fixed value.

Figure 4. Service task distribution convergence trajectory.

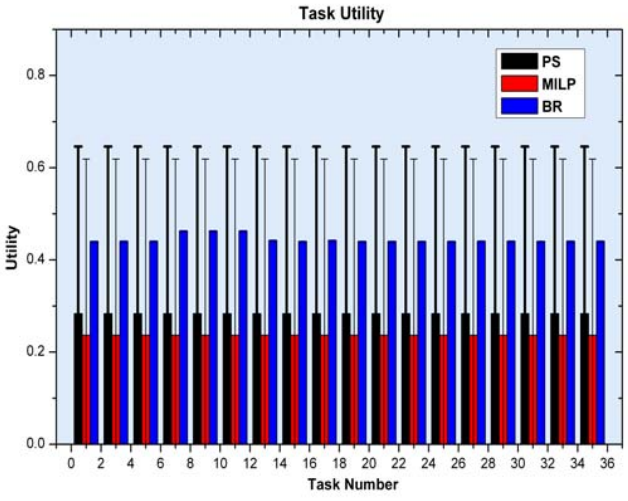

Figure 5. Comparison between expected and actual task utilities.

Task utility. The best reply iteration algorithm in this paper reduces the conflicts of multitask optimal resource selection under the multitask environment, attains the goal of collaborative optimization, and enhances the utility optimization of all tasks. The task expectancy utility plans the stage for the service chain, and the tasks do not consider the allocations and service optimal combination strategies of other tasks. The task utility is gained by the task in the operations. Utility error is the difference between the actual and expected task utilities. Compared with PS and MILP (Figure 5), BP method considers the conflicts between the combination strategies of other tasks. Therefore, the actual utility and utility error are all minimal. BP method stipulates that every task constantly changes and finally converges to the Nash equilibrium according to other task strategies.

Figure 6 shows that as the system load increases, the actual task utility increases as well. With the increase of system load, the number of specific service tasks allocated also increases, and the response time of the service increases

too. Then, the actual task utility increases accordingly and the performance is consequently reduced. The actual task utility of PS and MILP methods rises as the system load increases because the methods do not consider the combination strategies of other users; when the system load increases to a certain quantity, the task utility of the methods remarkably increased. BR method states that every task has to consider the combination strategies of other tasks; thus, the utility of each task has a nearly linear growth, and the method does not explode because of the growth of the system load.

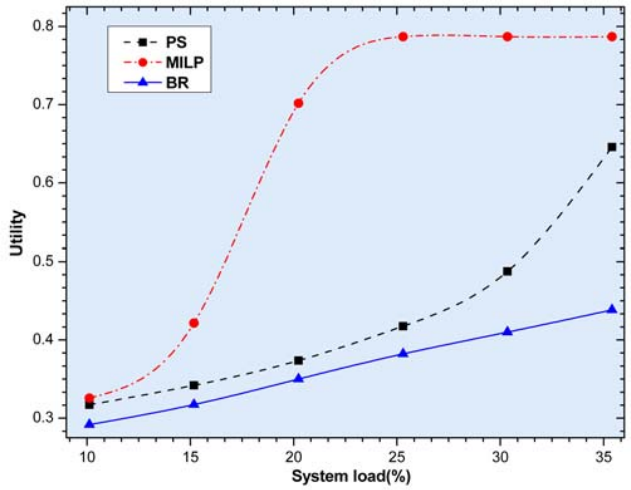

Figure 6. Relationship between system load and actual task utility

V. RELEVANT STUDY

In a Web service system, many services exist; these can satisfy the same function demands with different QoS parameters (such as execution time, cost, availability, etc.). A key problem is how to use the nonfunctional (QoS) properties of the service as bases to automatically select, optimize the combination, and execution the Web service chain [27]. The wide range of studies on QoS-aware service selection and combination method in the service-oriented computing area can be divided into two classes according to whether the running time can be adjusted:

1) Static environment

These methods do not consider dynamic environment changes and model the QoS-aware service chain combination as a constraint optimization problem.

According to the differences in computing the optimal strategy in the static environment, the QoS-aware method can be divided into two types: local optimization and global optimization-based methods[12, 28]. Local optimization method chooses the best services via greedy search. This method always has best performance but the solution of this method is not a globe optimal, but a local optimal. Global optimization-based method overcomes the disadvantages of local optimization, and can find the global optimization solution under QoS constraint[11]. Based on its five dimensions (cost, response time, reputation, success rate, and availability), the method analyzes work flow control structure, establishes the QoS-aware integer linear programming optimization model, and utilizes the general solver of integer linear programming. However, the method has the following disadvantages:

2) Dynamic environment

The QoS-aware service chain creation method under dynamic environments[29-31] is achieved mainly by supervising service execution and replan or rebind to guarantee that the service chain can smoothly executed with optimal performance[32].

Based on the replan [15, 33], the references used genetic algorithm to solve the QoS combination problems, considered property changes in executing services, and put forward a replan mechanism which can be triggered as quickly as possible when abnormal service conditions appear. A threshold beforehand is set in the strategy so that when any QoS property changes more than the threshold does, the program will replan. However, threshold was not properly defined, and the specific re-programming strategy was not provided.

Based on the segmentation optimization concept[34], the references used a two-stage re-programming method: in the first stage, the mixed integer programming is slackened into a linear programming, and allow for the decision variables to choose values in [0,1], to use the simplex method to quickly solve the creation conditions; the second stage applies back-track algorithm to build integer programming based on slack to obtain a feasible solution.

From the above, existing studies on QoS-aware optimal service combination method essentially are focused on single tasks to pursue the optimal solutions based on optimization theory. The competition and cooperation of resources among concurrent tasks are very common in the lifecycle of service optimal combination. Nevertheless, the competition and cooperation is the basis for maintaining orderliness, high efficiency, and coordinated services. As a result, studies on multitask concurrency with QoS-aware space information optimal service combination method are beneficial in analyzing and solving optimal combination problems of high-efficiency service resources in the entire system (all tasks). In a comprehensive utility perspective, all tasks have the best comprehensive utility of space information service to guarantee that in the complicated space service system, the highly efficient allocation and dispatch of service resources can be realized.

VI. SUMMARY AND OUTLOOK

Non-cooperative game based QoS-aware Web service combination method solves the performance bottleneck caused by optimal service competition in the Web service chain in multitask conditions, in addition to realizing the coordinated optimization of all service chains. Theories and experiment results have demonstrated that the method has a good convergence and all expected average task utility could be maximized.

But there still some issues are not considered in this paper. The main problem is that the best reply based

algorithm will increase the computational complexity for the iteration process. Actually, the performance degeneration is little with a small ε given, e.g. equal 10^{-3}, the iteration number is no more than 16. In this paper, we mainly consider the response time calculating method and best services compositing caused by confliction of concurrent tasks, and the performance of algorithm will be considered in future work.

Further work will include two parts: theory and experiment. Theory analysis will include the evolutionary game model with dynamic environmental changes, system optimization, Pareto optimization, as well as a relationship analysis featuring the Nash equilibrium. Experiment analysis will include more experiments in the PlanetLab.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (NSFC, 41001220), and Postdoctoral Science Foundation in Central South University.

REFERENCES

[1] Foster, I.: 'Services for Science', in Editor (Ed.)^(Eds.): 'Book Services for Science' (Springer-Verlag, 2008, edn.), pp. 3

[2] Foster, I.: 'Service-Oriented Science', Science, 2005, 308, (6), pp. 814-817

[3] Menascé, D.A.: 'QoS Issues in Web Services', IEEE Internet Computing, 2002, 6, (6), pp. 72-75

[4] Menascé, D.A.: 'Composing Web Services: A QoS View', IEEE Internet Computing, 2004, 8, (6), pp. 88-90

[5] Wang, S., Sun, Q., and Yang, F.: 'Towards Web Service selection based on QoS estimation', International Journal of Web and Grid Services, 2010, 6, (4), pp. 424-443

[6] Alameh, N.: 'Chaining geographic information Web services', IEEE Internet Computing, 2003, 7, (5), pp. 22-29

[7] Alameh, N.: 'Service Chaining of Interoperable Geographic Information Web Services', IEEE Internet Computing, 2002, 7, (5), pp. 22-29

[8] Ardagna, D., and Pernici, B.: 'Adaptive Service Composition in Flexible Processes', IEEE Transactions on Software Engineering, 2007, 33, (6), pp. 369-384

[9] Benatallah, B., Dumas, M., Sheng, Q.Z., and Ngu, A.H.H.: 'Declarative composition and Peer-to-Peer provisioning of dynamic Web services', in Editor (Ed.)^(Eds.): 'Book Declarative composition and Peer-to-Peer provisioning of dynamic Web services' (IEEE Computer Society, 2002, edn.), pp. 297-308

[10] 10 Casati, F., Ilnicki, S., Jin, L.-J., Krishnamoorthy, V., and Shan, M.-C.: 'eFlow: A Platform for Developing and Managing Composite e-Services', in Editor (Ed.)^(Eds.): 'Book eFlow: A Platform for Developing and Managing Composite e-Services' (IEEE Computer Society, 2000, edn.), pp. 341-349

[11] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q.Z.: 'Quality driven web services composition', in Editor (Ed.)^(Eds.): 'Book Quality driven web services composition' (ACM, 2003, edn.), pp. 411-421

[12] Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H.: 'QoS-Aware Middleware for Web Services Composition', IEEE Transactions on Software Engineering, 2004, 30, (5), pp. 311-327

[13] Ko, J.M., Kim, C.O., and Kwon, I.-H.: 'Quality-of-service oriented web service composition algorithm and planning architecture', The Journal of Systems and Software, 2008, 81, (1), pp. 2079-2090

[14] Shen, Y.-h., and Yang, X.-h.: 'A self-optimizing QoS-aware service composition approach in a context sensitive environment', JOURNAL OF ZHEJIANG UNIVERSITY-SCIENCE C-COMPUTERS & ELECTRONICS, 2011, 12, (3), pp. 221-238

[15] Canfora, G., Penta, M.D., Esposito, R., and Villani, M.L.: 'QoS-Aware Replanning of Composite Web Services', in Editor (Ed.)^(Eds.): 'Book QoS-Aware Replanning of Composite Web Services' (IEEE Computer Society, 2005, edn.), pp. 121-129

[16] Claro, D.B., Albers, P., and Hao, J.-K.: 'Selecting Web Services for Optimal Composition', in Editor (Ed.)^(Eds.): 'Book Selecting Web Services for Optimal Composition' (2005, edn.), pp.

[17] Canfora, G., Penta, M.D., Esposito, R., and Villani, M.L.: 'A framework for QoS-aware binding and re-binding of composite web services ', J. Syst. Softw., 2008, 81, (10), pp. 1754-1769

[18] Fudenberg, D., and Tirole, J.: 'Game theory' (MIT Press, 1991. 1991)

[19] Gross, D., and Harris, C.M.: 'Fundaments of Queueing Theory' (John Wiley & Sons, 1985, 2nd edn. 1985)

[20] Weibull, J.W.: 'Evolutionary Game Theory' (MIT Press, 1997. 1997)

[21] Bacon, D.F., Graham, S.L., and Sharp, O.J.: 'Compiler Transformations for High-Performance Computing', ACM Computing Surveys, 1994, 26, (4), pp. 345-420

[22] Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K.: 'Quality of Service for Workflows and Web Service Processes', Journal of Web Semantics, 2004, 1, (3), pp. 281-308

[23] Marchetti, C., Pernici, B., and Plebani, P.: 'A quality model for multichannel adaptive information', in Editor (Ed.)^(Eds.): 'Book A quality model for multichannel adaptive information' (ACM, 2004, edn.), pp. 48-54

[24] Grosu, D., and Chronopoulos, A.T.: 'Noncooperative Load Balancing in Distributed Systems', Journal of Parallel and Distributed Computing, 2005, 65, (9), pp. 1022-1034

[25] Boulogne, T., Altman, E., and Pourtallier, O.: 'On the Convergence to Nash Equilibrium in Problems of Distributed Computing', Annals of Operations Research, 2002, 109, (1-4), pp. 279-291

[26] Chow, Y.-C., and Kohler, W.H.: 'Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System', IEEE Transactions on Computers, 1979, 28, (5), pp. 354-361

[27] Hoffmann, J., Bertoli, P., Helmert, M., and Pistore, M.: 'Message-Based Web Service Composition, Integrity Constraints, and Planning under Uncertainty: A New Connection', Journal of Artificial Intelligence Research, 2009, 35, pp. 49-117

[28] Ardagna, D., and Pernici, B.: 'Global and Local QoS Constraints Guarantee in Web Service Selection', in Editor (Ed.)^(Eds.): 'Book Global and Local QoS Constraints Guarantee in Web Service Selection' (2005, edn.), pp. 805-806

[29] Chiu, D., Deshpande, S., Agrawal, G., and Li, R.X.: 'A Dynamic Approach toward QoS-Aware Service Workflow Composition' (2009. 2009)

[30] Besson, J., and Caplinskas, A.: 'QOS-Aware Composition of Enterprise System's Components: Constraint Logic Programming Approach', Informatica, 2010, 21, (4), pp. 487-504

[31] Zeng, L., Lei, H., and Chang, H.: 'Monitoring the QoS for Web Services', in Editor (Ed.)^(Eds.): 'Book Monitoring the QoS for Web Services' (Springer-Verlag Berlin, Heidelberg, 2007, edn.), pp. 132-144

[32] Zeng, L., Ngu, A.H., Benatallah, B., Podorozhny, R., and Lei, H.: 'Dynamic composition and optimization of Web services', Distributed and Parallel Databases, 2008, 24, (1-3), pp. 45-72

[33] Cardoso, J.: 'Quality of Service and Semantic Composition of Workflows', University of Georgia, 2002

[34] Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R.: 'Heuristics for QoS-aware Web Service Composition', in Editor (Ed.)^(Eds.): 'Book Heuristics for QoS-aware Web Service Composition' (IEEE Computer Society, 2006, edn.), pp. 72-82