APPLYING GENETIC ALGORITHM TO SELECT WEB SERVICES BASED ON WORKFLOW QUALITY OF SERVICE

Shang-Chia Liu
Department of Business Administration,
Fu-Jen Catholic University
510, Zhongzheng Rd., Xinzhuang Dist.,
New Taipei City, 24205, Taiwan
scliu@mail.fju.edu.tw

Sung-Shun Weng
Graduate Institute of Information and Logistics Management,
National Taipei University of Technology
1, Sec. 3, Chunghsiao E. Rd.,
Taipei City, 10608, Taiwan
wengss@ntut.edu.tw

ABSTRACT

Due to the rapid development of Web technologies, Internet applications increasingly use different programming languages and platforms. Web services technologies were introduced to ease the integration of applications on heterogeneous platforms. The quality of Web services has received much attention as it relates to the service discovery process. However, less work has been done on issues related to the quality of composite services. This study uses the selection model along with the concept of workflow quality of service (QoS) in order to improve the quality of service performance of current Web services in the discovery process. It also uses a selection model as the foundation for selecting Web services, conducting simulations to measure the overall workflow QoS performance when implemented in sequence. However, optimal solutions to service composition selection require exponential time in the number of services. We therefore apply genetic algorithm to quickly find the best-fitting service composition. Finally, we score and sort each service composition based on the service requesters' preferences towards QoS. The results of the experiment show that considering workflow QoS in selecting service composition improves the actual QoS performance. At the same time, using genetic algorithm to optimize the service composition provides an improvement in the solution time.

Keywords: Web services, Workflow, Quality of service, Genetic algorithm, Service selection

1. Introduction

As each generation of Web technologies emerges, different types of service providers arise in response to these new technologies. To allow applications or services to execute across heterogeneous operating systems, several Web services specifications were developed. These specifications include XML, WSDL, SOAP, and the UDDI data exchange standard. Of the problems relating to Web services, the one that has received the most attention is the one of finding Web services that fits one's needs. The more web services are available, the more difficult it becomes to find the most appropriate service for a specific application [Tao et al. 2012]. Although Web services provide a universal data exchange platform and offer methods that represent the service retrieval interfaces, a service requester may not be able to find suitable Web services simply by using search keywords. Even if a service is found that conforms to the functional requirements, it is unclear if the quality performance of the service can satisfy the requester's needs [Gluhovsky 2009].

Although the present UDDI structure does not yet consider the quality of service (QoS) when searching for services, there is much interest in QoS. However, previous studies have largely focused on the quality of a single service. Much work has already been done in the field of Web services quality [Ran 2003a]. However, real-world applications frequently need to integrate services into a workflow. This study seeks to determine a way for service requesters to find the service composition that delivers the expected QoS.

When a service is composed of several sub-tasks, each sub-task will affect the overall performance of the overall service. One common example is when a task must be completed before another one can complete. In this

kind of situation, overall performance is not improved when each sub-task individually seeks the best QoS. On the contrary, because QoS optimization results in an additional service, we incur additional delay and cost. If service A needs to wait for service B to finish, then service A will be acceptable as long as it conforms to the quality required by the overall workflow. It is unnecessary to seek a service that has the shortest possible response time, and this unnecessary search cost can be avoided by the service requester. The same holds true for more complex interrelationships between sub-tasks. Individual sub-tasks do not need to optimize individual quality; rather, we want to find the best service composition for the overall required QoS.

Each service requester may have different demands on QoS, so the right service composition may differ depending on the needs of each service requester. Wang &Wang [2009] indicated that a process that uses a large number of services may take a long time to solve for optimal service composition. Thus, it is necessary to develop a method that can improve on solution speed while not sacrificing accuracy of the result [Zhang et al. 2010]. This study attempts to use the concept of workflow QoS to achieve the following objectives:

(1) Use QoS management of workflows to find composite services that satisfy certain quality requirements.
(2) Tailor the service composition to the differing quality needs of individual service requesters.
(3) Develop a method to speed up the selection of service composition, as well as a method to find the best-fit service composition.

2. Literature review

2.1. Web services

Web services have grown in popularity because of their ability to integrate different platforms and enable business-to-business software communication. Web services can also interpret the content of Web services, and convert them into a convenient format to allow the computer to request services itself, rather than relying on human beings [Luarn & Lin, 2003]. In addition, the Publish-Find-Bind model of Web services can promote the software reuse and online collaboration [Hsu et al. 2011]. The W3C Web Services Glossary defined "Web service" is a software system designed to support interoperable machine to machine interaction over a network. Other systems can interact with access methods designated by Web services by passing XML messages that follow the SOAP standard. Chen et al. [2003] declared The Web services can be considered as a set of callable interfaces to software programs or components, regardless of their implementations. Services can thus be discovered and called by other agent programs, achieving the objective of system integration. Web services can easily provide interoperable software functions over the Intranet and the Internet. From a distributed computing architecture viewpoint, Web services describe a service-oriented and component-based application architecture [Chen and Meixell 2003].

Enterprises Web services typically involve the integration of heterogeneous systems [Cao et al. 2010]. Heterogeneity means that the operation platforms and databases of sub-systems as well as programming languages may be different. Technologies that are applied to system integration, such as DCOM, CORBA, RMI and other management technologies for distributed components, are also used as communication tools among heterogeneous systems. However, these methods require heterogeneous systems to use similar distributed technologies. If services using different distributed technologies need to communicate with each other, complicated middleware must be written to translate. Because of the many different methods for constructing system platforms in a distributed environment, integrating systems that are built from different platforms will surely be time-consuming. The SOAP technology of Web services can solve the above problems.

A Web service is an open and distributed software component. Its foundation is built on HTTP, XML, SOAP, WSDL, UDDI and other standards. Users can use development tools for any programming language and operating system to describe and write Web services.

Zeng et al. [2003] noted that component services should be selected during the execution of a composite service rather than at design-time. Rao and Su [2004] introduced a method for automatically composing semantic Web services using linear logic theorem proving. Their method used a semantic Web service language (DAML-S) for external presentation of Web services, and internally represented services by extra-logical axioms and proofs in linear logic. In the study, they proposed an architecture where the DAML-S parser, linear logic theorem prover and semantic reasoner can work together. Kuter et al. [2005] presented a hierarchical test network planning algorithm designed for planning domains in which information about the initial state of the world may not be complete, but is discoverable through plan-time information-gathering queries. They showed that their system is sound and complete, and derived several mathematical relationships between the amount of available information, the likelihood of the planner finding a plan, and the quality of the plan found. They also performed experimental tests that confirmed their theoretical results and demonstrated how the system can be used for Web service composition. Agarwal et al. [2005] formulated the Web service composition problem and described the first integrated system for composing Web services end to end, i.e., from specification to deployment. The proposed solution is based on a

novel two-staged composition approach that addresses the information modeling aspects of Web services, provides support for contextual information while composing services, employs efficient decoupling of functional and non-functional requirements, and leads to improved scalability and failure handling. They also presented Synthy, a prototype of the service composition system, and demonstrated its effectiveness on an application scenario from the telecom domain. Zhang [2005] discussed the issue of trustworthy Web services and encouraged researchers to work towards standardizing on a precise and comprehensive definition of Web services trustworthiness. Park & Park [2008] provided an A* heuristic search algorithm to find an optimal invocation plan for a given set of Web services and also presented a greedy method of generating an efficient solution in a short time. The experimental results showed that their proposed greedy method can find a close-to-optimal solution efficiently and had good scalability for a complex call hierarchy of Web services.

2.2. Quality of service (QoS)

The study of quality of service (QoS) is currently an active field of research [Kasap et al. 2007]. QoS can be used to express the non-functional requirements of network community research [Salamatian & Fdida 2001]. Some studies have defined QoS in distributed systems: how one can express the desired QoS for a system as well as how requests can be delivered to a resource manager to satisfy QoS requirements [Qi et al. 2011]. QoS originated as a research topic in the fields of networking, real time computing, and system middleware. At present, related research on the quality of Web services has contributed greatly to improving the quality of Web services.

Although the broker mechanism can use the service's basic description to easily obtain the required services, this only gives access to information recorded during service registration and does not provide any indication of the condition or quality of the service itself. Hence, in actual implementation of the service, situations sometimes arise where the service is unable to function or the quality fails to satisfy users' requirements. [Ran 2003a, Ran 2003b] mentioned several QoS problems that remain in Web services, such as how to determine whether services conform to the performance requirements, and whether they can provide a high degree of reliability for constructing key tasks of a system. [Ran 2003a, Ran 2003b] then focused his discussion on QoS issues and proposed a new Web service discovery framework and a discovery model.

Sirin et al. [2004] developed a goal-oriented, interactive composition approach that used matchmaking algorithms to help users filter and select services while building the composition. They implemented these ideas in a prototype system that can compose Web services and provide filtering capabilities in cases where many similar services are available. Zeng et al. [2004] presented a middleware platform which addresses the issue of selecting Web services for composition in a way that maximizes user satisfaction (utility functions over QoS attributes) while satisfying the constraints set by the user and by the structure of the composite service. Two selection approaches were described and compared in their work: one based on local (task-level) selection of services and the other based on global allocation of tasks to services using integer programming. Ko et al. [2009] proposed a QoS-oriented web service composition planning architecture. The main modules of the architecture were composition broker and execution plan optimizer. With the aid of the UDDI server, the composition broker discovered candidate outsourced web services for each atomic process of the selected schema and gathered QoS information on the web services. After that, the execution plan optimizer ran the web service composition algorithm in order to generate a QoS-oriented composition plan. The simulated results have shown that the algorithm was very efficient.

2.3. Workflow and quality of service

At present, the service-oriented architecture promoted by Internet standard organizations only handles the registration, query and linkage of a single service. Because the functions provided by a single service are limited, E-commerce applications usually integrate or link several Web services to satisfy requirements. In the future, Web services will likely move toward composite Web services, integrating several sub-services to collaboratively complete a task [Chandrashekhar & Bhasker 2011]. Future business systems will require seamless integration of business processes, business applications, business intelligence and Web services. Many sub-processes of business processes may also be formed by Web services [Qi et al. 2011]. When services are integrated into business processes, not only should the quality of the service itself be considered, but the effect of quality on the overall process should be also taken into account. The CrossFlow approach considered not only time, but also the cost that would be incurred in the implementation of the workflow [Grefen et al. 2000; Klingemann et al. 1999].

Another issue in composite services is the workflow problem. Workflow management has been used in business processes for many years. Studies related to workflow have been discussed in some research into cross-organization integration. Cardoso et al. [2002] proposed a workflow QoS model, a method to compute the QoS of a workflow. These QoS metrics for workflow include workflow time, workflow cost and the degree of workflow reliability. Cardoso et al. [2004] presented a predictive QoS model that made it possible to compute the quality of service for workflows automatically based on atomic task QoS attributes. They also presented the implementation of their QoS model for the METEOR workflow system. They described the components that have been changed or added, and

discussed how they interact to enable the management of QoS. Jaeger et al. [2004] introduced a mechanism that determined the QoS of a Web service composition by aggregating the QoS dimensions of the individual services. This allowed them to verify whether a set of services selected for composition satisfies the QoS requirements for the whole composition. Their aggregation built upon abstract composition patterns which represented basic structural elements of a composition: sequence, loop, and parallel execution. Jaeger et al. [2005] extended existing composition patterns with the ability to consider dependencies between services. They also studied how to use pattern-based aggregation in the run-time monitoring process and explained how the data derived from the monitoring process can be used to calculate a more accurate aggregation of QoS for the composition. Maximilien & Singh [2004] addressed dynamic service selection via an agent framework coupled with a QoS ontology. With the approach, participants can collaborate to determine each other's service quality and trustworthiness. Xia et al. [2006] proposed a stochastic model to evaluate QoS (make-span, reliability and cost) of workflow systems based on QWF-net, which extended traditional WF-net by associating tasks with firing-rate, failure-rate and cost-coefficient. They also presented a sensitivity analysis technique to identify QoS bottleneck.

2.4. Genetic algorithm

The genetic algorithm (GA) is a search technique developed by Holland [1992] and used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Control parameters of genetic algorithms include population size, crossover rate, chromosome coding, fitness function, and termination condition.

Genetic algorithms can be used in the search for optimal composition. However, although binary genetic algorithms can find the optimal solution, too much time is used in complex encoding and decoding operations [Nazif & Lee 2012]. When there are too many parameters and a large number of bits are required to represent them, encoding and decoding will seriously slow down the searching speed of the system. On the other hand, real-valued genetic algorithms do not have this problem. In addition, most real world optimization problems have real-valued parameters. Using real-valued genetic algorithms to solve problems not only makes it more convenient to manage parameters, but also eliminates the complicated computation of encoding and decoding, and further enhances the accuracy of the system. This study applies real-valued encoding schemes to the dynamic encoding problems in various workflow situations.

Su et al. [2007] proposed an improved genetic algorithm to select optimal web services composition plans from a lot of composite plans on the basis of global QoS constraints. A special fitness function and a mutation policy were proposed in the work. The simulation results have shown that the improved genetic algorithm can gain effectively the composite service plan that satisfied the global QoS requirements, and that the convergence of genetic algorithm was improved. Ma and Zhang [2008] presented a quickly convergent population diversity handling genetic algorithm for web service selection with global QoS constraints. In their study, an enhanced initial population policy and an evolution policy were proposed based on population diversity and a relation matrix coding scheme. The simulation results have shown that convergence and stability of genetic algorithm were improved greatly.

3. Research methodology

When Web service requesters need to search for services from UDDI, multiple services may have to combine into a composite service in order to complete the task. Sub-services in the composite service may possess sequential processes, parallel processes, conditional processes, and processes with recursive and network relationships. Figure 1 gives an example of a workflow. For the service requester, selecting a composite service that possesses optimal quality is a very complicated problem. The workflow sub-tasks in figure 1 contain the ordered workflow sequences. For example, task 1 (T1) produces some output on execution, which then becomes the input of task 2 (T2); similarly, task 3 (T3), task 4 (T4), and task 5 (T5) all proceed simultaneously and send their results to task 6 (T6). We can expect different quality from different combinations of workflows and services. In addition, different service requesters will also weigh quality differently.

The main objective of this study is to solve the problem of insufficiency of current service brokers. This insufficiency arises because existing brokers only provide the discovery of service names and function names to service requesters. Even if the discovered service names and functions conform to the search conditions, the service may not satisfy the quality requirements of the service requester. Although several papers in the literature propose to use the QoS filtering mechanisms to select for QoS itself, existing studies have largely focused on filtering the quality of a single service. When Web services need to be integrated with other applications involved in workflow processes, workflow problems will be involved. This study is designed with the following three main points:

(1) Because there are many registered Web services, obtaining services with specific functions requires a filtering mechanism.

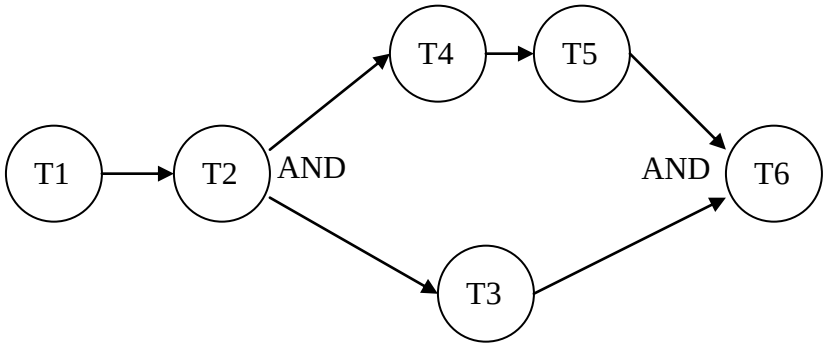

Figure 1: Workflow example 1

(2) Although filtering can greatly decrease the number of Web services under consideration, composite Web services that involve workflow still have the problem of selecting the optimal service composition. Hence, the quality selection mechanism must be used to find the optimal service combination.
(3) When the number of Web services greatly increases, using brute force to find the optimal combination will be too slow. We must develop a method that will find the best-fit combination in the shortest time.

3.1. System architecture of Web services selection

The architecture of this study's selection system is shown in figure 2.

(1) Web services are registered in each UDDI by the service provider. The number of unfiltered Web services may be enormous. Hence, filtering must be done to list the services that conform to service requester requirements.
(2) Candidate services are those remaining after the filtering mechanism has selected on functional descriptions. Services are provided by different service providers, with varying quality.
(3) Recommendation of composite services is done by selecting among combinations of services based on the performance quality weights set by users. The combinations are then sorted to help user select the appropriate service combination.
(4) The selection model is divided into two parts in this study: "filtering mechanism" and "selection mechanism based on workflow QoS". These parts will be discussed in more detail later.
(5) The workflow management engine calculates the QoS of the overall workflow based on the workflow description, as well as the implementation order of each sub-task and information about input resources and output destination.
(6) The QoS database stores the quality index values of the provided Web services in UDDI.

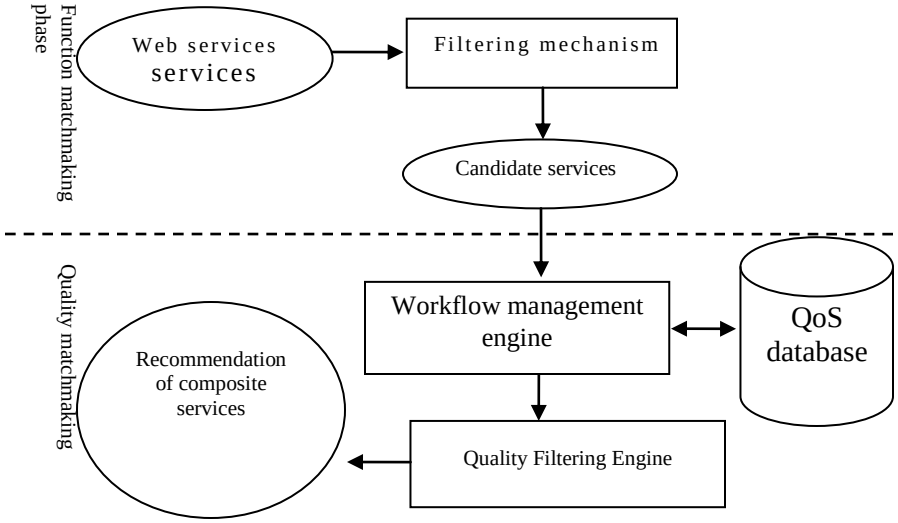

Figure 2: System architecture of Web services selection

3.2. Selection model of Web services

This study uses a two-part selection model of Web services. The first part is the filtering mechanism which decreases the number of candidate services in order to avoid unnecessary searching in the selection stage. The second part is the selection mechanism which uses workflow QoS to deduce the quality performance of the composite services. Figure 3 is a diagram of the selection mechanism. The respective components in the mechanism are described below.

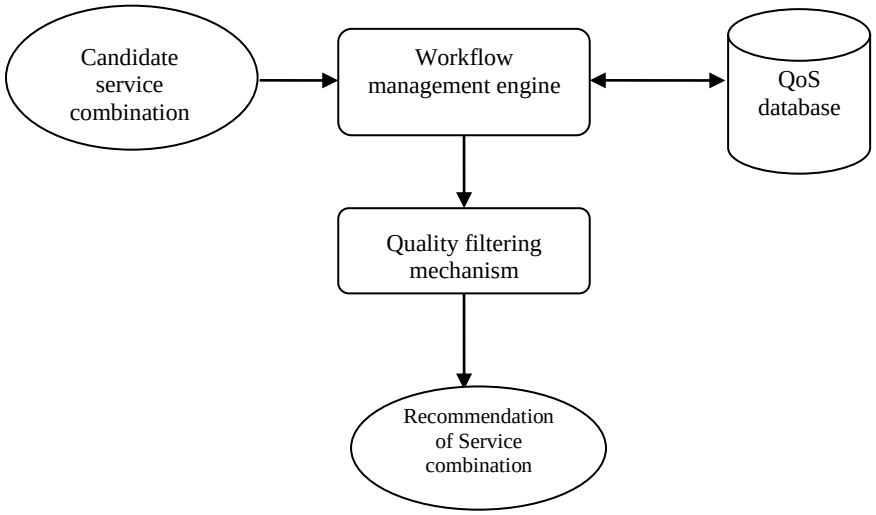

Figure 3: Selection mechanism based on workflow QoS

(1) Workflow management engine

This study's workflow management engine adopts the SWR algorithm of [Cardoso et al. 2002] as the deduction tool for workflow QoS. Processes can be classified into sequential processes, parallel processes, conditional processes, and processes with recursive relationships. We will examine the sequential and parallel process in some detail here.

The objective of the workflow management engine is to simulate and deduce the QoS performance of the workflow in actual execution. There are three categories of evaluation indices for measuring QoS:

(a) Response time (RT): the total time spent for a workflow process.

a. The sequential process is shown in figure 4. The time calculation can be divided into three steps:

Step 1: Find the terminal task ($task_k$) first.

Step 2: Add the time of the terminal task ($task_k$) to the time of the higher level parent task ($task_j$) and then delete the original terminal task ($task_k$). $task_j$ then replaces $task_k$ to become the new terminal task.

Step 3: If $task_j$ is the root node, then its time is the total time of the whole process. Otherwise, repeat step 2 until the root node is found.

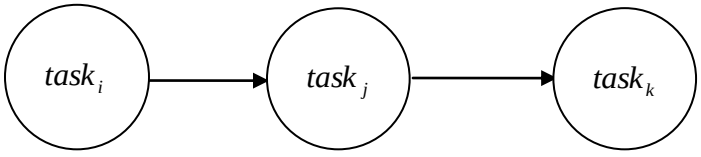

Figure 4: Sequential process

Based on the above computation steps, we can compute the total time (t-taski) spent by all the sub-tasks in the process. The total time (T) spent can then be expressed by formula 1:

$$\mathbf{T} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{t} - \mathbf{task_i} \qquad (1)$$

b. For parallel processes, the overall workflow QoS is affected not by total time, but by the worst-performing of all the simultaneously-running processes, as indicated in formula 2:

$$\mathbf{T = Max(t\text{-}task_i),\ i \in n} \quad (2)$$

(b) The overall service cost (C) is simply the sum of all individual task costs, as shown in formula 3.

$$\mathbf{C} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{c - task_i} \quad (3)$$

(c) Reliability (R): the probability that the sub-services involved in the overall process can perform correctly. There are two steps to compute this value.

Step 1: Compute the individual service reliability ($r-task_i$) from the service QoS records. The formula is given in formula 4, where N_C is the number of successful implementations of the sub-task, and N_S is the total successful call frequency.

$$r-task_i = \frac{N_C}{N_S} \quad (4)$$

Step 2: Estimate its overall reliability based on the reliability of the sub-services and the type of the service process. For example, if the workflow is a sequential process, the total service reliability is the product of the sub-tasks reliabilities, as given in formula 5.

$$R = \prod_{i=1}^{n} r-task_i \quad (5)$$

The above formulas only calculate the workflow QoS value for a specific scenario. In different situations, the workflows involved would also be different.

(2) Quality filtering mechanism

The fitness function of this study is based on the most common attributes of QoS found in the literature: response time (T), cost (C), and reliability (R). Before carrying out quality filtering, we first normalize the quality parameters, then compute the overall score based on the weighted value of each index.

(a) Normalization

Step 1: Normalize to a value between 0 and 1, according to formula 6.

$$S_i = \frac{Q_i - Q_{min}}{Q_{Max} - Q_{min}} \quad (6)$$

Step 2: If higher scores mean lower quality, then invert the scale using formula 7. If higher scores mean higher quality, then pass them along unchanged as in formula 8.

$$NS_i = 1 - NQ_i \quad (7)$$

$$NS_i = NQ_i \quad (8)$$

(b) Setting the QoS weights

Different service requesters will have different QoS weights, based on each requester's preferences for quality. However, the weights must add up to 100%.

(c) Overall score

Formula 9 gives the overall score of a particular service composition as the weighted sum of each QoS metric, where W_T , W_C , and W_R representing the weights for response time, cost, and reliability, respectively. T, C, and R are the QoS indices as calculated above. T_{Max} and C_{Max} are the maximum values for response time and cost among all service compositions, while T_{min} and C_{min} are the minimum values for response time and cost among all service compositions.

$$F = W_T \times \left(1 - \frac{T - T_{\min}}{T_{Max} - T_{\min}}\right) + W_C \times \left(1 - \frac{C - C_{\min}}{C_{Max} - C_{\min}}\right) + W_R \times R \quad (9)$$

3.3. Optimal selection system

We simulate several different scenarios to experimentally examine the validity of the proposed methods. By considering workflow QoS early in the selection process, the service composition will more closely match the service requester's preferences, for all settings of weights. Thus, we consider the workflow QoS in the quality filtering stage.

However, with a large number (n) of sub-tasks in the workflow, and m candidate services, this method must consider an exponential number (m^n) of possible workflows. Although the service composition with the highest

QoS is found, the solution time is too long. This study uses genetic algorithm (GA) in the selection mechanism to select the optimal candidate composition more quickly. Parallel processing can also be applied to shorten the search.

The revised selection system randomly produces a population. Through selection, crossover, copy, and mutation, and with natural selection operating to preserve good genes, we can quickly find a set of the best-fit solutions. However, conventional genetic algorithms use a binary encoding scheme which cannot adapt to varying numbers of sub-task services. We avoid this problem by using a real-valued encoding scheme.

(1) The architecture of the optimal selection system

Our optimal selection system is based on the previously-discussed selection system, and is shown in figure 5. The filtering mechanism is unchanged from before, while genetic algorithms are introduced into the optimal selection mechanism.

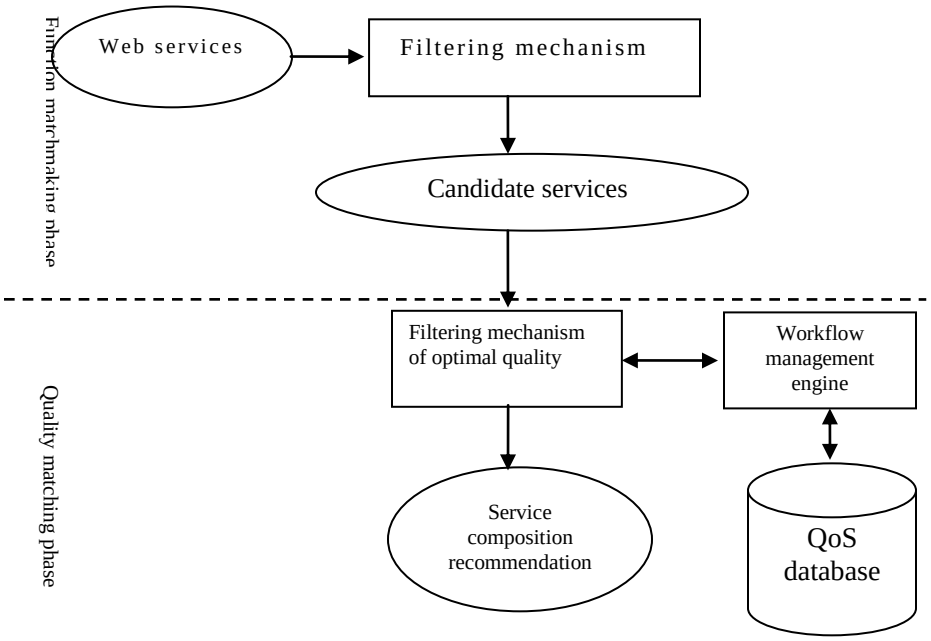

Figure 5: Architecture of optimal selection system

(2) The workflow-based optimal selection mechanism

Figure 6 shows the architecture of the workflow-based optimal selection mechanism.

(a) Gene encoding: Each bit holds a random integer number.

(b) Mutation: Select a random position i to mutate, then set it randomly to an integer value from 1 to N_i.

(c) Fitness function: Calculate fitness from formula 9. Because we do not know beforehand the maximal and minimal values of all the workflow QoS scores, we derive them based on the maximal and minimal values of the time and cost indices.

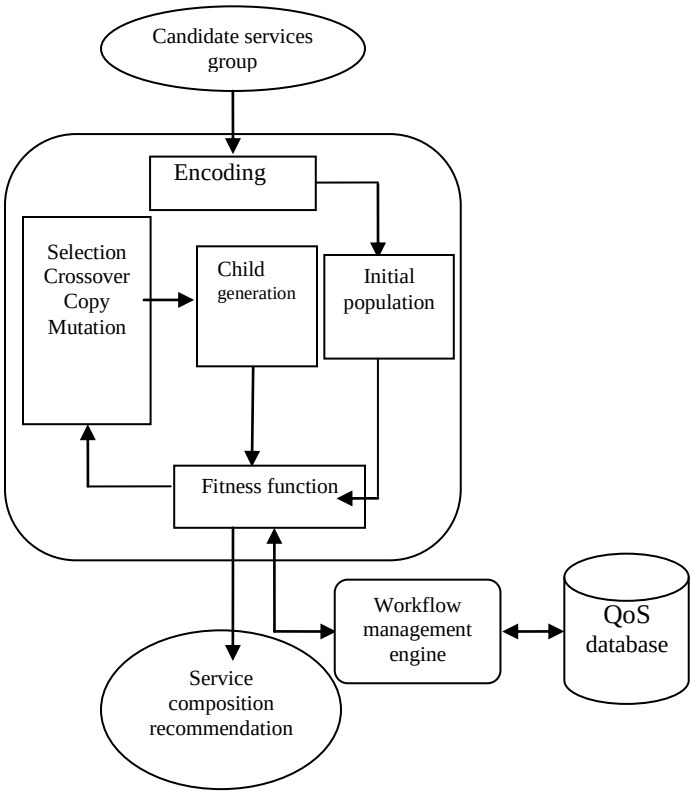

Figure 6: Architecture of the optimal selection mechanism

4. Analysis and discussion of experimental results

4.1. Different scenarios and experiments

Suppose that the service requester needs to create a system and the system is composed of six sub-tasks. The system workflow is shown in figure 7. Several service providers each operate multiple services in the service brokers' UDDI registration system (Appendix A). Information from actual service operation is recorded in the QoS database. In the experiments, all services make it as candidate services after passing through the functional requirement filtering. The experiments compare several selection strategies to demonstrate the feasibility of this study's selection strategy.

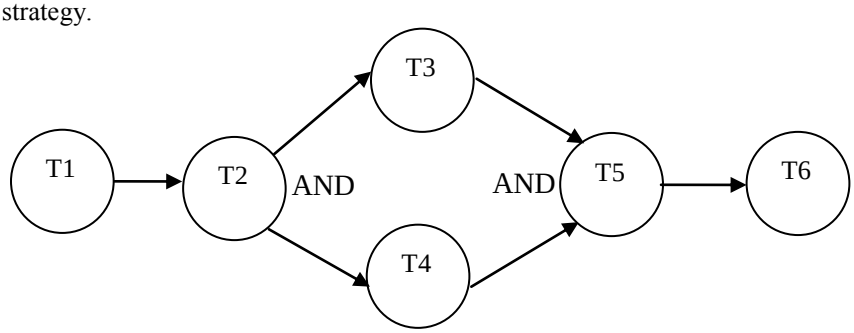

Figure 7: Workflow example 2

(1) Case 1

In this case, we use the Web service searching method to find the services shown in Appendix B. Each task has several services which conform to the required conditions. The requester is then left with the difficult task of selecting among the services.

(2) Case 2

In this case, we add QoS as a factor in the selection of services, and set different weights for quality indices based on the preferences of different service requesters. For example, response time (T) may be set to 30%, cost

(C) to 45% and reliability (R) to 25%. The service with the highest score after normalization is the one that best conforms to the service requester's expectations. The services selected by each task are shown in table 1.

Table 1: Details of scores and selection of QoS

Task	SID	Supplier and service		Time	Cost	Reliability	Score
Task1	2BAF4C50-6B85-11D7-BADE-000629DC0A53	SP3	Service1	39.92	8	0.854	0.747
Task2	84411F90-67AA-11D6-A961-000C0E00ACDD	SP7	Service1	9.681	46	0.809	0.709
Task3	1058EB70-CC16-11D9-9561-000629DC0A53	SP1	Service4	25.950	20	0.829	0.744
Task4	C691F8A0-3E06-11D7-9590-000629DC0A53	SP14	Service1	10.364	30	0.885	0.864
Task5	9DEED770-6AFF-11D8-8C1B-000629DC0A53	SP12	Service2	18.413	18	0.817	0.762
Task6	68D9EE80-180B-11D9-A73F-000629DC0A53	SP7	Service2	35.726	20	0.797	0.620

(3) Case 3

Holding QoS conditions constant, we also consider the workflow sequence, as shown in table 2. Although the reliability (0.268) of the service composition selected by this case (strategy 2) is lower than that (0.330) of the service composition selected by case 2 (strategy 1), this better satisfies the service requester's preferences, because he prefers low cost and time to high reliability.

Table 2: Details of scores and selection of workflow QoS

		Strategy 1	Strategy 2
Task	Task1	2BAF4C50-6B85-11D7-BADE-000629DC0A53	2BAF4C50-6B85-11D7-BADE-000629DC0A53
	Task2	84411F90-67AA-11D6-A961-000C0E00ACDD	AF341450-137B-11D7-9B59-000629DC0A53
	Task3	1058EB70-CC16-11D9-9561-000629DC0A53	1058EB70-CC16-11D9-9561-000629DC0A53
	Task4	C691F8A0-3E06-11D7-9590-000629DC0A53	607E7470-90EC-11D6-B746-000C0E00ACDD
	Task5	9DEED770-6AFF-11D8-8C1B-000629DC0A53	9DEED770-6AFF-11D8-8C1B-000629DC0A53
	Task6	68D9EE80-180B-11D9-A73F-000629DC0A53	9A0C09D0-3077-11D6-83CD-000C0E00ACDD
Workflow QoS	Time	129.690	107.723
	Cost	142	140
	Reliability	0.330	0.268
Score		0.578	0.595

4.2. Experiments and result analyses of the optimal selection system

(1) Examination of convergence

Suppose we have a workflow that includes 10 sub-tasks, each with 30 Web services that conform to the functional requirements. First, we check if the optimal selection system gradually converges to the best-fit solution when the genetic algorithm is used. Two experiments were conducted, recording the result of evolution for each 50 generations of evolution. After passing through 3000 generations, we indeed see gradual convergence taking place. As it can be seen in figure 8, the earlier generations have not yet improved enough to exceed the fitness values obtained where each sub-task selects the best service itself.

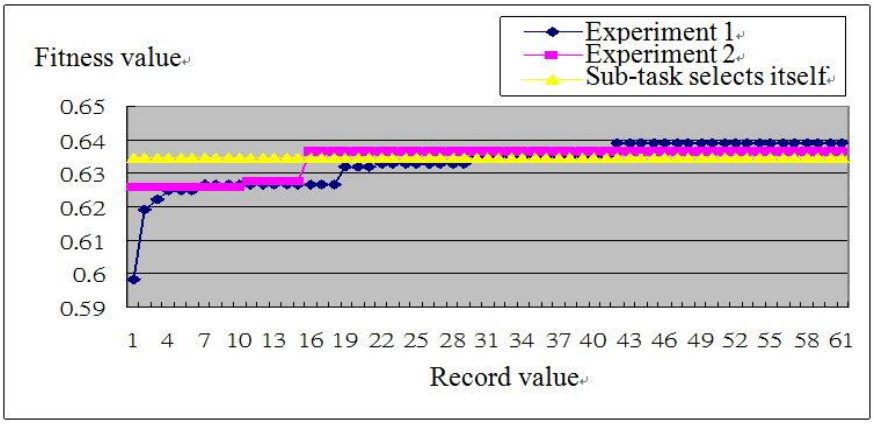

Figure 8: Comparison of GA strategy with each sub-task selecting the best service itself

After observing the results of experiments, we note that although the overall workflow QoS is influenced by each sub-service's QoS, individual selection of best service by each sub-task can maintain fitness at a certain level. Therefore, this study assumes that the service composition resulting from local optimization contains the good genetic components that can influence the overall QoS. In the next stage of experimentation, we exploit this fact by first having each sub-task individually select the best service, then use the result to seed the initial service composition for the genetic algorithm. This simply brings the default solution into the initial population so that its good genes can get passed along.

(2) Comparison of fitness values among different selection strategies

In this experiment, we simulate 100 runs of the two optimal selection systems for differing numbers of sub-tasks. Figure 9 compares the three strategies. We see that, when the number of subtasks is less than five, the two types of genetic algorithm both perform better than local selection of optimal services by individual sub-tasks. With more than five subtasks, however, seeding the population with the default solution is superior to the individual selection strategy, while the genetic algorithm with random initial population performs worse than both other strategies.

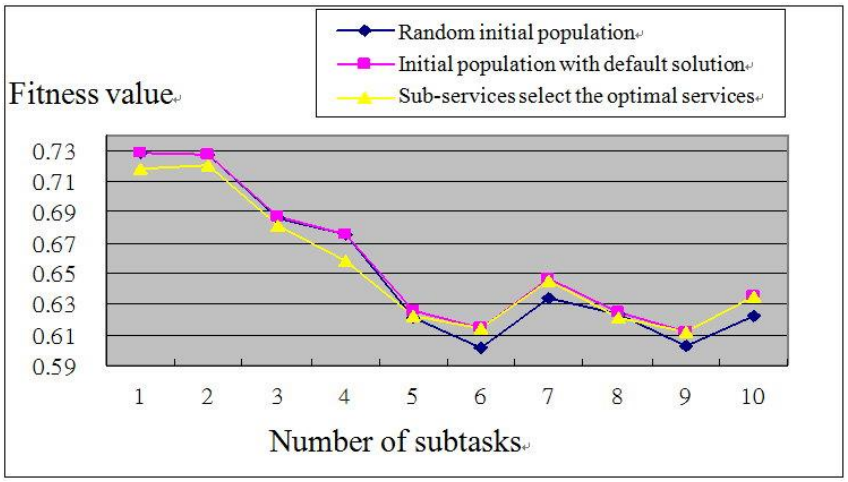

Figure 9: The fitness values of three types of selection strategies

From the above experiments, we see that the genetic algorithm with a seeded initial population is the best strategy of the three that we examined. When the number of services exceeds five, it outperforms the individual selection strategy slightly. Figure 10 gives the percentage of best-fit solutions found by the seeded genetic algorithm over 100 runs. Suppose hit rate represents the probability value to find the service combination with better performance by the strategy of initial population with default solution. We see that the runs with 6, 7, 9, and 10 subtasks all produce hit rates less than 15%, while 8 subtasks produces a hit rate of more than 60%. We infer some possible reasons for this result:

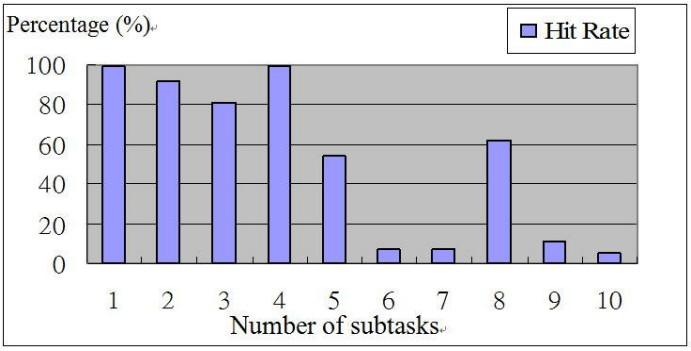

Figure 10: Distribution of hit rates of the best-fit solutions

(a) The number of the population may be too small, resulting in a lack of diversity in the evolution of the population.
(b) The number of evolution generations is too small to converge to the best-fit solution.
(c) Under this workflow scenario, there are fewer feasible solutions. Hence, the local selection strategy is likely to find a good solution, and the small set of possible solutions does not allow the genetic algorithm to find a much better solution.

(3) Exploration of abnormal reasons

We examine the 10-subtask situation further under all ten workflow scenarios to determine the reasons for suboptimal performance of selection strategies:

(a) Holding other conditions fixed, increase the population size from 100 to 500. From figure 11, we see that an increase in population size to 500 causes the hit rate to increase from 5% to 26%. However, an improvement in fitness score is not seen. Hence, the size of the population is not the main factor that causes poor performance.
(b) Holding other conditions fixed, the number of generations is increased to 5000. We again conduct two simulations, recording the population once every 50 generations. As seen in figure 12, this experiment demonstrates that increasing the number of generations can improve convergence to the fittest solution.
(c) Holding other conditions fixed, repeat the experiment on different workflows. Figure 13 shows that the hit rate increases from the original 5% to 55%. We can conclude that different workflows greatly influence the resulting service composition.

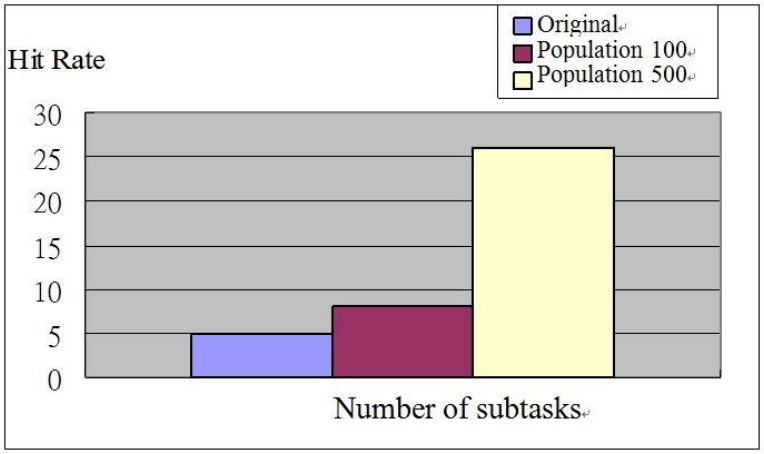

Figure 11: Comparison of hit rates for different population sizes

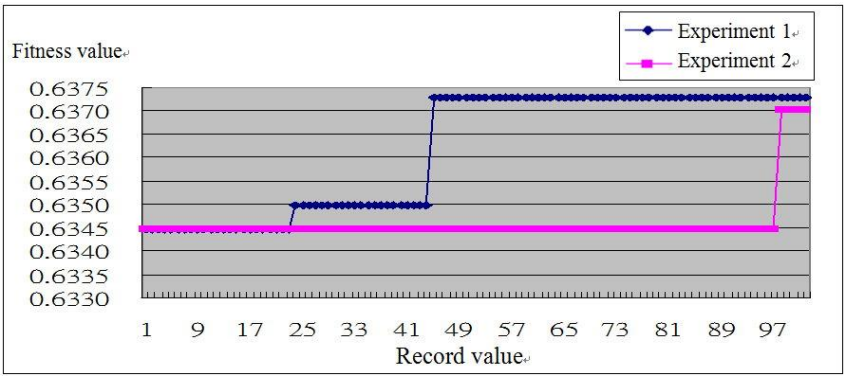

Figure 12: Convergence diagram after increasing the number of generations

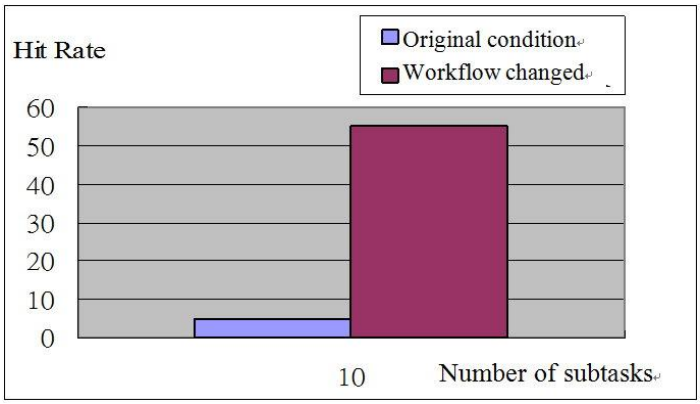

Figure 13: Comparison of hit rates after changing the workflow

5. Conclusions

Web services usually need to be integrated into business processes. When a composite service is required, workflow issues are often involved. We believe that, when searching for services, we must be able to actually simulate the operations and processes of the services and understand the overall service performance that may occur with workflows involved. In addition, the service requester must be provided with the appropriate recommendation for his/her preferences for QoS. To more closely match QoS to the requirements of service requesters, this study applies genetic algorithm to optimize on the simulated deduction of workflow QoS. Based on the weights supplied by service requesters, we attempt to improve the discovery procedure for finding composite services.

Our experiments show that actual QoS performance of composite services is better when considering workflow than when not considering workflow in the initial stage of service selection. Use the weights can also move the selection closer to the QoS preferences of service requesters. The experiments also show that genetic algorithm can reduce the time it takes to find the optimal service composition. It is originally believed that when there is only one workflow sub-task, it would not matter what kind of selection strategy is adopted. However, simulations show that even when the service number is one, when there is a recursive process implemented more than once, different selection strategies will select different tasks. As a result, the consideration of overall workflow QoS is important in the early stage of service selection.

REFERENCES

Agarwal, V., G. Chafle, K. Dasgupta, N.M. Kamik, A. Kumar, S. Mittal, and V. Srivastava, "Synthy: a system for end to end composition of Web services," *Journal of Web Semantics*, Vol. 3, No. 4:311-339, 2005.

Cao, H.J., H. Jin, X.X. Wu, and S. Wu, "Service flow: QoS-based hybrid service-oriented grid workflow system," *Journal of Supercomputing*, Vol. 53, No. 3:371-393, 2010.

Cardoso, J., A. Sheth, and J. Miller, "Workflow quality of service," In: *Proceedings of the International Conference on Enterprise Integration and Modeling Technique (ICEIMT'02)*, Valencia, Spain, 303-311, 2002.

Cardoso, J., A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and Web service processes," *Journal of Web Semantics*, Vol. 1, No. 3:281-308, 2004.

Chandrashekhar, H., B. Bhasker, "The Implications and Impacts of Web Services to Electronic Commerce Research and Practices," *Journal of Electronic Commerce Research*, Vol. 12, No. 3: 214-237, 2011.

Chen, M., N.K. Chen, and B. M. Shao, "Personalized Recommender System Using Entropy Based Collaborative Filtering Technique," *Journal of Electronic Commerce Research*, Vol. 4, No. 4: 128-139, 2003.

Chen, M., and M. J. Meixell, "Web Services Enabled Procurement in the Extended Enterprise: An Architectural Design and Implementation," *Journal of Electronic Commerce Research*, Vol. 4, No. 4: 140-155, 2003.

Gluhovsky, I., "Customoer behavior model for quality-of-service environments with many service levels," *Journal of Electronic Commerce Research*, Vol. 10, No. 1:29-41, 2009.

Grefen, P., K. Aberer, Y. Hoffner, and H. Ludwig, "CrossFlow: Cross-organizational workflow management in dynamic virtual enterprises," *International Journal of Computer Systems Science & Engineering*, Vol. 1, No. 5:277-290, 2000.

Holland, J.H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, *MIT Press*, Cambridge, MA, 1992.

Hsu, S.H.Y., S.K. Balasubramanian, R. Thakur, and S. Kulviwat, "Knowledge-based and online self-service," *Journal of Electronic Commerce Research*, Vol. 12, No. 2:133-151, 2011.

Jaeger, M.C., G. Rojec-Goldmann, and G. Muhl, "QoS aggregation for Web service composition using workflow patterns," *In: Proceedings of the Eighth IEEE International Conference on Enterprise Distributed Object Computing (EDOC'04)*, Washington, D.C., USA, 149-159, 2004.

Jaeger, M.C., G. Rojec-Goldmann, and G. Muhl, "QoS aggregation in Web service composition," In: *Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, Hong Kong, 181-185, 2005.

Klingemann, J., J. Wasch, and K. Aberer, "Deriving service models in cross-organizational workflows," In: *Proceedings of 9th International Workshop on Research Issues in Data Engineering (RIDE): Information Technology for Virtual Enterprises*, Sydney, Australia, 100-107, 1999.

Ko, J.M., C.O. Kim, and I.H. Kwon, "Quality-of-service oriented web service composition algorithm and planning architecture," *Journal of Systems and Software*, Vol. 81, No. 11:2079-2090, 2008.

Kasap, N., H. Aytug, and S.S. Erenguc, "Provider selection and task allocation issues in networks with different QoS levels and all you can send pricing," *Decision Support Systems*, Vol. 43:375-389, 2007.

Kuter, U., E. Sirin, B. Parsia, D. Nau, and J. Hendler, "Information gathering during planning for Web service composition," *Journal of Web Semantics*, Vol. 3, No. 2-3:183-205, 2005.

Luarn, P., and H. H. Lin, "A Customer Loyalty Model for E-Service Context," *Journal of Electronic Commerce Research*, Vol. 4, No. 4: 156-167, 2003.

Ma, Y. and C. Zhang, "Quick convergence of genetic algorithm for QoS-driven web service selection," *Computer Networks*, Vol. 52, No. 5:1093-1104, 2008.

Maximilien, E.M. and M.P. Singh, "A framework and ontology for dynamic Web services selection," *IEEE Internet Computing*, Vol. 8, No. 5:84-93, 2004.

Nazif, H. and L.S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Applied Mathematical Modelling*, Vol. 36, No. 5:2110-2117, 2012.

Park, C.S. and S. Park, "Efficient execution of composite Web services exchanging in tensional data," *Information Sciences*, Vol. 178, No. 2:317-339, 2008.

Qi, L.Y., W.M. Lin, W.C. Dou, J. Jiang, and J.J. Chen, ""A QoS-aware exception handling method in scientific workflow execution," *Concurrency and Computation-Practice & Experience*, Vol. 23, No. 16:1951-1968, 2011.

Ran, S. "A framework for discovering Web services with desired quality of services attributes," In: *Proceedings of the International Conference on Web Services (ICWS '03)*, Las Vegas, 208-213, 2003a.

Ran, S. "A model for Web services discovery with QoS," ACM SIGecom Exchanges, Vol. 4, No. 1:1-10, 2003b.

Rao, J. and X. Su, "Toward the composition of semantic Web services," *Lecture Notes in Computer Science*, Vol. 3033:760-767, 2004.

Salamatian, K. and S. Fdida, "Measurement based modeling of quality of service in the Internet: a methodological approach," *Lecture Notes in Computer Science*, Vol. 2170:158-174, 2001.

Sirin, E., B. Parsia, and J. Hendler, "Filtering and selecting semantic Web services with interactive composition techniques," *IEEE Intelligent Systems*, Vol. 19, No. 4:42-48, 2004.

Su, S., C. Zhang, and J. Chen, "An improved genetic algorithm for web services selection," *Distributed applications and interoperable systems (DAIS 2007), Lecture Notes in Computer Science*, Vol. 4531:284-295, 2007.

Tao, Q., H.Y. Chang, C.Q. Gu, and Y. Yi, "A novel prediction approach for trustworthy QoS of web services," *Expert Systems with Applications*, Vol. 39, No. 3:3676-3681, 2012.

Wang, H. and S.H. Wang, "Adaptable algorithm for designed web process sequence data analysis," *Journal of Electronic Commerce Research*, Vol. 10, No. 2:104-113, 2009.

Xia, Y., H.P. Wang, Y. Huang, and L. Yuan, "A stochastic model for workflow QoS evaluation," *Scientific Programming*, Vol. 14, No. 3-4:251-265, 2006.

Zeng, L., B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng, "Quality driven Web services composition," *In: Proceedings of the 12th International conference on World Wide Web*, Budapest, Hungary, 411-421, 2003.

Zeng, L., B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE transactions on software engineering*, Vol. 30, No. 5:311-327, 2004.

Zhang, J. "Trustworthy Web services: actions for now," *IEEE IT Professional*, Vol. 7, No. 1:32-36, 2005.

Zhang, M.W., B. Zhang, Y. Liu, J. Na, and Z.L. Zhu, "Web service composition based on QoS rules," *Journal of Computer Science and Technology*, Vol. 25, No. 6:1143-1156, 2010.

Appendix A: Details of suppliers and services

SID	Supplier	Service	Time	Cost	Reliability
08F4DA20-60C0-11D6-8194-000C0E00ACDD	SP1	Service1	9.882	58	0.840
1BC47600-7F6A-11D7-9124-000629DC0A53	SP1	Service2	17.712	20	0.770
15EE20E0-3F63-11D9-AC66-000629DC0A53	SP1	Service3	46.748	14	0.847
1058EB70-CC16-11D9-9561-000629DC0A53	SP1	Service4	25.950	20	0.829
22433D10-99F8-11D7-8DA6-000629DC0A53	SP2	Service1	19.759	26	0.758
29201E70-08A0-11D8-95A0-000629DC0A53	SP2	Service2	47.342	6	0.732
2BAF4C50-6B85-11D7-BADE-000629DC0A53	SP3	Service1	39.92	8	0.854
2E6856A0-1186-11D9-A2B3-000629DC0A53	SP3	Service2	47.672	4	0.750
39FF8410-F0D7-11D6-8F10-000629DC0A7B	SP3	Service3	27.042	88	0.863
5A391070-C5D7-11D9-9561-000629DC0A53	SP4	Service1	6.628	44	0.780
5D9C85A0-6246-11D6-8194-000C0E00ACDD	SP4	Service2	35.302	8	0.808
607E7470-90EC-11D6-B746-000C0E00ACDD	SP4	Service3	23.168	16	0.788
64600C80-86B0-11D5-A4A5-0004AC49CC1E	SP5	Service1	23.223	18	0.727
68650750-B58C-11D8-83B1-000629DC0A53	SP5	Service2	51.53	6	0.714
777CA100-BEA4-11D8-B362-000629DC0A53	SP5	Service3	6.382	46	0.776
8349A990-180B-11D9-A73F-000629DC0A53	SP6	Service1	51.017	12	0.782
59CD4BF0-0E2E-11D7-AF73-000629DC0A53	SP6	Service2	12.852	44	0.817
84411F90-67AA-11D6-A961-000C0E00ACDD	SP7	Service1	9.681	46	0.809
68D9EE80-180B-11D9-A73F-000629DC0A53	SP7	Service2	35.726	20	0.797
8831C380-6240-11D6-8194-000C0E00ACDD	SP8	Service1	16.903	22	0.803
BB821FD0-A0F1-11D9-9561-000629DC0A53	SP8	Service2	20.475	34	0.781
8A678CB0-ED69-11D7-A0CA-000629DC0A53	SP9	Service1	15.495	30	0.759
6E615020-D950-11D5-80AA-0004AC49CC1E	SP9	Service2	45.306	12	0.826
188D5550-CC17-11D9-9561-000629DC0A53	SP9	Service3	3.5759	54	0.832
8CFA6C50-D00A-11D5-8833-0004AC49CC1E	SP10	Service1	55.487	4	0.788
0DD31990-2148-11D8-B936-000629DC0A53	SP10	Service2	23.712	84	0.784
96ED5F70-4CDA-11D8-B936-000629DC0A53	SP11	Service1	35.727	14	0.763
21372600-D35A-11D9-9561-000629DC0A53	SP11	Service2	46.489	14	0.823
9A0C09D0-3077-11D6-83CD-000C0E00ACDD	SP12	Service1	7.804	42	0.770
9DEED770-6AFF-11D8-8C1B-000629DC0A53	SP12	Service2	18.413	18	0.817
A8C9C0B0-60C0-11D6-8194-000C0E00ACDD	SP12	Service3	10.480	58	0.775
AF341450-137B-11D7-9B59-000629DC0A53	SP13	Service1	15.636	36	0.764
C34CE250-0D8B-11D7-AF73-000629DC0A53	SP13	Service2	32.751	16	0.742
C691F8A0-3E06-11D7-9590-000629DC0A53	SP14	Service1	10.364	30	0.885
CDC602E0-14C4-11D6-A0DC-000C0E00ACDD	SP14	Service2	14.931	48	0.805
F651C8A0-17EA-11D7-9B59-000629DC0A53	SP14	Service3	61.282	6	0.831

Appendix B: Details of services that conform to service conditions

Task	SID	Supplier	Service
Task1	2BAF4C50-6B85-11D7-BADE-000629DC0A53	SP3	Service1
	8CFA6C50-D00A-11D5-8833-0004AC49CC1E	SP10	Service1
	C34CE250-0D8B-11D7-AF73-000629DC0A53	SP13	Service2
	59CD4BF0-0E2E-11D7-AF73-000629DC0A53	SP6	Service2
Task2	84411F90-67AA-11D6-A961-000C0E00ACDD	SP7	Service1
	6E615020-D950-11D5-80AA-0004AC49CC1E	SP9	Service2
	39FF8410-F0D7-11D6-8F10-000629DC0A7B	SP3	Service3
	68650750-B58C-11D8-83B1-000629DC0A53	SP5	Service2
	AF341450-137B-11D7-9B59-000629DC0A53	SP13	Service1
Task3	1058EB70-CC16-11D9-9561-000629DC0A53	SP1	Service4
	8349A990-180B-11D9-A73F-000629DC0A53	SP6	Service1
	08F4DA20-60C0-11D6-8194-000C0E00ACDD	SP1	Service1
	F651C8A0-17EA-11D7-9B59-000629DC0A53	SP14	Service3
	A8C9C0B0-60C0-11D6-8194-000C0E00ACDD	SP12	Service3
	CDC602E0-14C4-11D6-A0DC-000C0E00ACDD	SP14	Service2
Task4	607E7470-90EC-11D6-B746-000C0E00ACDD	SP4	Service3
	1BC47600-7F6A-11D7-9124-000629DC0A53	SP1	Service2
	0DD31990-2148-11D8-B936-000629DC0A53	SP10	Service2
	5A391070-C5D7-11D9-9561-000629DC0A53	SP4	Service1
	8831C380-6240-11D6-8194-000C0E00ACDD	SP8	Service1
	5D9C85A0-6246-11D6-8194-000C0E00ACDD	SP4	Service2
	C691F8A0-3E06-11D7-9590-000629DC0A53	SP14	Service1
	21372600-D35A-11D9-9561-000629DC0A53	SP11	Service2
Task5	8A678CB0-ED69-11D7-A0CA-000629DC0A53	SP9	Service1
	BB821FD0-A0F1-11D9-9561-000629DC0A53	SP8	Service2
	9DEED770-6AFF-11D8-8C1B-000629DC0A53	SP12	Service2
	15EE20E0-3F63-11D9-AC66-000629DC0A53	SP1	Service3
	188D5550-CC17-11D9-9561-000629DC0A53	SP9	Service3
	64600C80-86B0-11D5-A4A5-0004AC49CC1E	SP5	Service1
Task6	9A0C09D0-3077-11D6-83CD-000C0E00ACDD	SP12	Service1
	68D9EE80-180B-11D9-A73F-000629DC0A53	SP7	Service2
	2E6856A0-1186-11D9-A2B3-000629DC0A53	SP3	Service2