Journal of Computers Vol. 29 No. 5, 2018, pp. 177-189
doi:10.3966/199115992018102905014

Reliability Prediction and QoS Selection for Web Service Composition

Weitao Ha[1*], Wei Zhao[2]

[1] School of network security and information, Weinan Normal University, Weinan 714099, Shaanxi, China
wnchlp@126.com

[2] School of Mathematics and Physics, Weinan Normal University, Weinan 714099, Shaanxi, China
wnhwt@126.com

Received 7 June 2017; Revised 7 November 2017; Accepted 7 December 2017

Abstract. The key issues in the development of Web service composition are the dynamic and efficient reliability prediction and the selecting component services appropriately. In this paper, we discuss Web service composition in two ways: reliability prediction and QoS optimal selection. Firstly, we propose a reliability prediction model based on Petri net. To address the complex connecting relationship among subservices, places of basic Petri net for Input and Output are extended to some subtypes for multi-source input place and multiuse output place. Secondly, we use a new skyline algorithm based on a R-tree index. The index tree is traversed to judge whether it is dominated by the candidate Skyline sets. Experimental evaluation on real and synthetic data shows the effectiveness and efficiency of the proposed approach.

Keywords: atomic services, effectiveness and efficiency, QoS, reliability, service selection, skyline, utility function, Web service composition, workflow relationship

1 Introduction

A new business opportunity is created by the possibility of providing value-added Web services. Such value-added Web services can be built through the integration and composition of basic Web services available on the Web. Generally, after the user submits requests that the composite service achieves, along with some constraints and preferences that need to be satisfied, Web services composition will be implemented through four steps:

(1) abstract component services discovery;
(2) selections of the component Web services;
(3) consistence verification of composite Web service;
(4) execution of the composite Web service;

At the first step, automatic services composition system finds multi-group potential services from abstract service layer of network which satisfy user's function requests. During the second step, the component Web services are selected to fulfill the user's goal. As a tremendous amount of available web services with identical function attributes but different QoS are spread all over the Internet, it is intractable to find the appropriate web services satisfying the given goal quickly. Besides, the composite Web service will not guarantee reliable execution and consistency if the component services are chosen only according to QoS attributes. Therefore, it is necessary to predict component Web services reliability before composition. The third step of the composition process finds failure and conflict of some specific component services by consistence verification. At the final step, reliable and QoS optimal composite Web service is implemented.

The task of building Web service compositions is much more difficult because service components may be developed, operated, and evolved by independent organizations. Each organization has its own

* Corresponding Author

business rules, Web services must be treated as strictly autonomous units. Heterogeneity manifests itself through structural and semantic differences that may occur between semantically equivalent Web services. Thus, there is no guarantee that some components will not deviate from the specification, once a system is under operation. Furthermore, Quality of Service (QoS) parameters play a major role in determining the success or failure of the composed application. Different users may have different requirements and preferences regarding the QoS. QoS properties are important factors which determine whether users satisfy execution result of Web composite service. A QoS-aware approach to service composition is therefore needed, which maximizes the QoS of composite service executions by taking into account the constraints and preferences set by the users. As a tremendous amount of available web services with identical function attributes but different QoS is spread all over the Internet, when selecting web services for composition through the repository UDDI (Universal Description, Discovery and Integration), it is intractable to find the appropriate web services satisfying the given goal quickly

The research goal of this paper is to build a composite Web service which can ensure not only correct and reliable execution but also optimal QoS. We find that the Petri net model allows describing a static vision of a system and its dynamic behavior, and it is expressive enough to capture the semantics of complex Web services combinations and their respective interactions. In this paper, we propose a reliability prediction model based on Petri net. For atomic services, a staged reliability model is provided which predict reliability from network environment availability, service availability of hermit equipment, discovery reliability, and binding reliability. To address the complex connecting relationship among component Web services, places of basic Petri net for Input and Output are extended to some subtypes for multi-source input place and multi-use output place. Moreover, we use skyline computation to select services for composition effectively and efficiently, reducing the number of candidate services to be considered. We also define aggregation functions, and use a Multiple Attribute Decision Making approach for the utility function to achieve QoS-based optimal service selection. Experimental evaluation on real and synthetic data shows the effectiveness and efficiency of the proposed approach.

2 Related Works

One notable research topic that has attracted much attention recently is automatic Web service composition which ensures not only reliable execution but also optimal QoS. Indeed, these two aspects of selection always are implemented separately.

A service composition is formed as follows:

(1) Web service providers publish their web service and invocation interfaces with WSDL;

(2) Register the web services to UDDI;

(3) The user submits the goal which the composite service achieves, along with some constraints and preferences that need to be satisfied;

(4) Application programs discover the needed web services;

(5) Send the requests via invocation interfaces.

During this process, interfaces play an important role in service composition. Therefore, Ding et al. [1] attempts to develop a method to compute the reliability of service based on the interface reliability. A formal service component signature model and a language-independent dynamic behavior model were proposed for Service Component Architecture. Rules have been developed to compute port expression reliability. Path-based approach computes the reliability of all the possible execution paths. The reliability of each path is computed by multiplying the reliabilities of the components on that path. The system reliability is computed by averaging over all the paths [2]. Web services can be viewed as component-based systems, making these findings useful in modeling reliability for web service applications. Zo et al. [3] focuses on deriving reliability measures at the application level. It examines reliability of an entire application created from a set of web services. It places greater emphasis on combining reliability measures for different web services, than on determining the antecedents of reliability for each web service. It views reliability of web services as a measure of failure-free operation, without modeling the source and type of failure. Some researchers view reliability as a non-functional characteristic of web services [4]. Still others characterize it as a component in setting service level agreements with web services vendors [5].

The existing methods of reliability prediction have the following disadvantages:

(1) Reliability prediction of composite service ignores reliability evaluation for component services;

(2) There are few effective methods to translate service composition specification into reliability prediction model;

(3) Reliability evaluation is combined with performance evaluation.

QoS guarantee for Web services is one of the main concerns of the SLA framework. There are projects studying QoS-empowered service selection. In [6], authors present a QoS-aware Web service compositions which is middleware-supporting quality-driven. But the method is based on integer linear programming and best suited for small-size problems as its complexity increases exponentially with the increasing problem size. In [7], the authors propose an extensible QoS computation model that supports an open and fair management of QoS data by incorporating user feedback. However, the problem of QoS-based composition is not addressed by this work. The work of Zeng at al. [8-9] focuses on dynamic and quality-driven selection of services. The authors use global planning to find the best service components for the composition. They use linear programming techniques [10] to find the optimal selection of component services. Linear programming methods are very effective when the size of the problem is small, but suffer from poor scalability due to the exponential time complexity of the applied search algorithms [11]. Despite the significant improvement of these algorithms compared to exact solutions, both algorithms do not scale with respect to the number of candidate web services, and hence are not suitable for real-time service composition. There are many available Web services with identical function attributes but different QoS where a composite service is interested in viewing the best Web service based on multiple QoS criteria. Skyline computation is used to select web services for composition through the repository UDDI in [12]. It models the problem as a skyline query known as the maximum vector problem. As web service sets used for skyline processing are often huge, computation can be expensive, and efficient algorithms are vital for selecting web services. With the advance of multi-core architectures and other parallel computing platforms, parallel skyline algorithms offer a new way. The proposed skyline based algorithm in this paper is complementary to these solutions as it can be used as a pre-processing step to prune non-interesting candidate services and hence reduce the computation time of the applied selection algorithm.

With the above quotation, the approaches implement conventional optimal QoS composition, but composing optimal QoS Web services does not guarantee a reliable execution of the resulting composite Web service. Therefore, reliability prediction and QoS Optimizing should be integrated.

3 Reliability Prediction for Web Service Composition

Due to the inherent autonomy and heterogeneity of web service it is difficult to predict the overall behavior of a composite service. Unexpected behavior or failure implement of a component service might not only lead to its failure but also may bring negative impact on all the participants of the composition. Web service composition process must provide reliable and consistent execution.

The service reliability can be understood as the probability of a successful execution of a Web service. A successful service execution in this case should be understood as achieving desired result of Web service, no execution errors appearing, and maintaining overall agreed-upon quality of service. For composite services, there are two levels of reliability: atomic service reliability and composite service reliability. Atomic service reliability is the reliability of the service as an independent unit. Drawing the line and isolating the service from its surroundings for reliability analysis are still hard, since service reliability is affected by operating process and the operating environment. The composite service reliability is affected by the reliability of each service in the composition, the reliability of the operating environment, and the service interactions in the form of usage profile. Usage profiles are of great concern in the frequency of executing each service and each interaction between services. When a composite service is composed dynamically, usage profiles will be unknown beforehand and can be observed only during execution. Therefore, they form a complex challenge when analyzing composite service reliability [13].

In this section we analyze network environment availability, service availability of hermit equipment, discovery reliability, and binding reliability to find the cause of failure and present staged reliability models for atomic services. On the foundation of atomic services reliability models, we also present reliability computation approach of composite services.

3.1 Reliability Model for Atomic Web Service

An atomic service is the smallest organization unit of web service. In other words, it would be impractical to decompose an atomic service. An atomic service has a fine-grained structure which can be designed, implemented, and tested independently. The atomic service reliability is defined as the probability that an atomic service invocation will be completed successfully. This means a correct response to the atomic service invocation is successfully received under the specified conditions and time constraints. In this case, the reliability can be determined by the following factors: network environment availability, Service availability of hermit equipment, discovery reliability and binding reliability [14].

Network environment availability expresses how much communication links affect atomic service reliability. This factor is more important in the wireless mobile network. So we will calculate network environment availability R_{nea} of wireless mobile network. Set d_c to express distance between atomic service node and master station of wireless mobile signal. $d_c \geq 0$, and d_c obeys logarithmic normal distribution. We can obtain average of d_c's logarithm μ_d from empirical value. σ_d is standard deviation of d_c's logarithm. The probability density function $f_d(d_c, \mu_d, \sigma_d)$ is $f_d(d_c, \mu_d, \sigma_d) = \exp(-(\ln(d_c) - \mu_d)^2 / (2\sigma_d^2)) / (\sqrt{2\pi} d_c \mu_d)$. Network environment availability is computed by $R_{nea} = \int_0^{d_{\max}} f_x(x, \mu_d, \sigma_d)$, where $d_{\max}$ is effective distance maximum of atomic service node.

Reliability of service hermit equipment expresses how hermit equipment availability affects atomic service reliability. N expresses number of atomic services of running hermit equipment at some point of time. ω is expected arrival number of atomic services for hermit equipment. In general, random variables N follows Poisson distribution whose parameter is ω. So distribution probability is $P(N = n) = (\omega)^n \exp(-\omega) / n!$. Reliability of service hermit equipment is defined by R_{hea} $R_{hea} = \sum_{n=1}^{N} \exp(P_f \cdot t_a) \cdot P_n$. In this formula, P_f is failure probability when n atomic services reach hermit equipment at the same time, and t_a is run time of atomic service.

Reliability of the service discovery is the probability of return correct service which is requested by user at the given time interval when user request discovery service to service registry. In the process of service discovery, two errors may occur: the requested service isn't found as it doesn't exit or list in the service list; service discovery criteria provided by user is incorrect or inconsistence between service descriptions and service function cause service discovery error. When the two errors are considered together, the formula reliability of the service discovery R_{dis} is shown as the following.

$$R_{dis} = N - n / N \sum_{i=1}^{N-n} \alpha_i / (N - n) \quad \textbf{(1)}$$

To the formula, N is the total number of services which are found according to user's request, and n is the number of error services of discovery, and α_i is mistake probability of the ith service in grammar logic.

Binding reliability is probability of binding specific from abstract service, after the service is found. After finding appropriate services for user, UDDI will feed back these specific services matching abstract a service to user in the process of service execution, and one of specific services is bound to implement. In process of binding two errors may occur:

(1) The mistakes of WSDL service description document lead to wrong binding. The probability of error is denoted by P_{F_1}. Let λ denote failure intensity parameter.

(2) Binding service can't satisfy user's QoS needs. The probability of error is denoted by P_{F_2}. Let μ denote failure intensity parameter.

Binding reliability R_{bind} is shown as the following:

$$R_{bind} = 1 - (\lambda \prod_{i=1}^{n} P_{F_{1i}} + \mu \prod_{i=1}^{n} P_{F_{2i}}) \quad \textbf{(2)}$$

The reliability of atomic is $R_a = R_{nea} * R_{hea} * R_{dis} * R_{bind}$.

3.2 Reliability Prediction for Web Service Composition

Reliability of composite Web service depends on reliability of atomic services, but that is not enough to only learn reliability of atomic services. Atomic service interactions and composition patterns all have an effect on reliability of service composition.

Petri Net have become a powerful system modeling tool used for the analysis of a wide range of systems coming from different domains (e.g., distributed computing, telecommunication, control systems, workflow management) and characterized by situations of concurrency, synchronization, causality and conflict. It is a model that is used to describe distributed systems and system structures, verify distributed systems, and also can simulate the operation of the system. Therefore, Petri net is particularly applicable to describe component-based multithreading, distributed software system structures, and constitutes a running relationship between the various components of a software system. Petri-net, compared to most other Web service composition models, is better able to describe subsequent executions of software system, such as a Web server composition.

A basic petri net is a 3-tuple (P, T, F) where:

$P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places,

$T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions,

$F = (P \times T) \bigcup (T \times P)$ is a finite set of arcs (flow relation).

A Petri net is basically characterized by places, transitions and arcs defining its structure and it is graphically represented by a directed bipartite graph in which places are drawn as circles, transitions are drawn as bars, input and output arcs are drawn as arrows and inhibitor arcs are drawn as circle headed arrows.

Several approaches proposed in [15-17] have some things in common: (1) They use Petri net to model control flows or path in the process of service composition. Inputs and outputs of places in Petri net only come from user's request. Connecting relationship among component services is described by one-to-one, but one-to-many and many-to-one relationships aren't considered. (2) They all estimate the reliability of composite service based on its internal structure. However, composite service reliability is affected by the reliability of each component service in the composition, and the reliability of the operating environment. Existing methods including literature [15-17] assume that reliability of atomic and component services are known on the basis of which reliability of composite service is computed. Besides, environmental impacts for composition reliability are overlooked. Using Petri nets to model Web service composition is not a new idea. However, our model has three aspects of innovation. Firstly, places of basic Petri net for Input and Output are extended to some subtypes for multi-source input place and multi-use output place. These can address the complex connecting relationship among subservices and realize dynamic organization of subservices. Secondly, ε Place and ε transition with combine-select and copy-split mechanisms are introduced for complex connecting relationship such as 1:n and n:1 [18]. Thirdly, we incorporate reliability of atomic services into reliability prediction for web service composition, based Petri nets model. During this process, reliability of atomic services combining composition patterns will form firing probability of transition. These can realize reliability prediction of composite service.

Definition 1. ECWS (Extended Composite Web Service). We define a Petri Net model to extended composite Web Service as a 6-tuple (P,T,WS,R,α,λ),where

(1) P is a finite nonempty set of places. $P = P_I \bigcup P_O \bigcup P_\varepsilon$, $P_I \bigcap P_O = \phi$. P_I is input places union of all component services. $P_I \bigcup P_{Ii} (I = 1, 2, 3, ..., K)$. In this Petri net model, P_I consists not just input places of user request, but two types of input places. One is input places of service coming from output places of other services. The other is input places requested by some services that no information can match. P_O is output places. The extended output places have three types. First, output places are target outputs of user request. Second, output places will be taken as input places of some component services. Third, output places will never be used. $P_O = \bigcup P_{Oi} (i = 1, 2, 3, ..., k)$. P_ε is general places.

(2) T is a finite set of transitions, corresponding to candidate component services execution, $P \bigcap T = \phi$. $T = T_s \bigcup T_\varepsilon$. T_ε denotes a finite nonempty transitions set of all component services. T_ε denotes a finite nonempty transitions set in which combine-select and copy-split mechanisms are introduced for complex

connecting relationship such as 1:n and n:1.

(3) R denotes relationship between P and T. $R = (P \times T) \bigcup (T \times P)$

(4) WS is component services set, and every component service denotes a Petri net model. $WS = \{WS_1, WS_2, WS_3, \cdots WS_k\}$. $WS_i = (P_{Ii}, P_{Oi}, T_i, \lambda_i)$. P_{Ii} and P_{Oi} denote input places and output places of component service WS_i. T_i is a finite set of transitions of component service WS_i. λ_i denotes firing probability of transition T_i for component service WS_i within the required time t, and $\lambda_i = R_a$. Atomic service reliability R_a is used to express probability of component service executed successfully.

(5) $\alpha : WS \to Type$ denotes aggregation mapping function. Type express control structures in Web services composition patterns. $Type \in \{null, sequence, and, or, loop\}$. Use CS to aggregate more complex composition. $CS := (WS_i \oplus WS_{i+1}) \mid (WS_i \otimes WS_{i+1}) \mid (WS_i \parallel WS_{i+1}) \mid (\lfloor WS_i)$.

(6) λ denotes firing probability for composite service WS. Firing probability for composite service WS reflects reliability of Web service composition. Reliability for an component web service is measured by R_a. The aggregated reliability of a composite service depends on the structure of the business process, the degree of independence between web services, and whether there are multiple web services (in the set of selected services) capable of supporting the task. In this paper, we present reliability of Web service composition with different the control structures in Web services composition patterns.

(7) In the sequence pattern $WS_{seq} = WS_1 \oplus WS_2 \oplus WS_3 \oplus \cdots \oplus WS_k$, firing probability λ_{seq} is $\lambda_{seq} = \prod_{i=1}^{n} (\lambda_i)$.

(8) In the parallel pattern $WS_{par} = WS_1 \parallel WS_2 \parallel WS_3 \parallel \cdots \parallel WS_k$, firing probability λ_{par} is $\lambda_{par} = \max_{i \le k} (\lambda_i)$.

(9) In the conditional pattern $WS_{con} = WS_1 \otimes WS_2 \otimes WS_3 \otimes \cdots \otimes WS_k$, firing probability λ_{con} is $\lambda_{con} = \prod_{i=1}^{k} (b_i \lambda_i)$, where bi is execution probability of branch i.

(10) In the loop pattern $WS_{loop} = \lfloor WS_i$, firing probability λ_{con} is $\lambda_{loop} = (\lambda_i)^v$, where v is iterations of component service WS_i.

4 Candidate Web Services Selection

Reliability prediction ensures reliable execution of composite Web service, but it is not enough to provide optimal composite Web service for users. The process of service composition contains some service classes and each service class can be achieved by a set of functionally-equivalent concrete services. The goal of QoS-aware service composition is to select best candidate services from each service class to satisfy QoS requirements and optimize overall QoS of the composite service as far as possible. However, it is not practical to perform an exhaustive search to find the best combination that satisfies QoS constraints in this scenario, because the number of functional equivalent services is very large. Skyline approach is also used to deal with the uncertainty of service in the process of selection [19]. Service selection is an important issue in the area of service computing. As the number of services and service providers proliferate, there are a large number of candidate, most likely competing, services for fulfilling a desired task. Performing an exhaustive search can be very expensive in terms of computation time and, therefore, inappropriate for run-time service selection in applications with many services and dynamic needs. Thus, service selection is becoming important for helping users to identify desirable services. User preferences play a key role during the selection process. Skyline computation offers a new solution of finding optimal data from huge data sets, whose computation can be expensive and whose applications require fast response times.

Our approach for QoS based service selection decreases the range of choices without effectively pruning the potential candidates by taking advantage of the skyline method. The basic selection with QoS-based skyline is to compute all skyline services in a static environment and then recommend them to be selected, while the advanced selection deals with real cases in practical applications.

4.1 Skyline Query

The skyline computation is an elegant summary method over multidimensional datasets [20]. A commonly cited example for the use of a skyline computation is assisting a tourist in choosing a set of interesting hotels from a larger set of candidate hotels. Each hotel is identified by two attributes: a distance from a specific point (such as a location on a beach), and the price for the hotel. To assist a tourist in narrowing down the choices, the skyline operator can be used to find the set of all hotels that are not dominated by another hotel. Hotel a dominates hotel b if a is at least as close as b and at least as cheap as b, and offers either a better price, or is closer, or both compared to b. Fig. 1 shows an example dataset and the corresponding skyline; the distance of the hotel from the beach is shown on the x-axis and the hotel price is plotted along the y-axis. The skyline is the set of points a, c, d, i, and j.

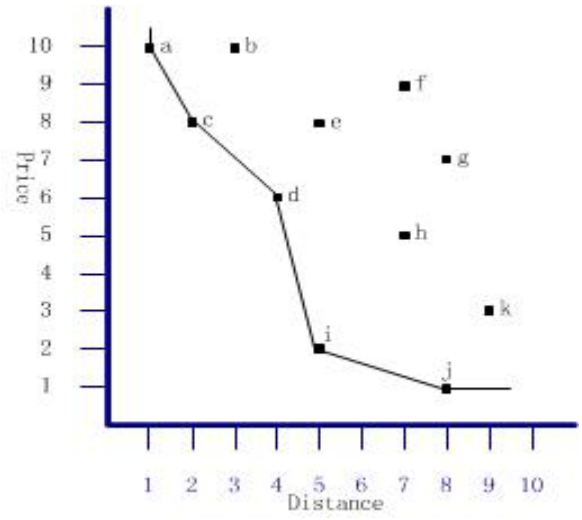

Fig. 1. Example dataset and its skyline

The skyline can be generalized to multidimensional space where a point a dominates another point b if it is as good or better than b in all dimensions, and is better than b in at least one dimension. Implicit in this definition of the skyline operation is the notion of comparing the goodness along each dimension. A common function for determining this property is to use the minimum function. However, skyline computation can easily be extended to consider other functions, such as max [21].

Definition 2. Skyline. Given a dataset P of n points in d-dimensional space. Let p and p´be two different points in P, we say p dominates p´iff for all i, p[i]≤p´[i] and for at least one i, p[i] < p´[i], where p[i] is the i-th dimension of p and 1 ≤i ≤d. The skyline points are those points in P that are not dominated by any other point in P.

4.2 Optimal QoS selection for Web Service Composition

Definition 3. Dominance. Given a service set WS_i assigned to activity A_i having n candidate services: $WS_{A_{i1}}, WS_{A_{i2}}, \cdots, WS_{A_{in}}$, QoS vector is d dimensions: $q_1(WS_{A_{i1}}), q_1(WS_{A_{i2}}), \cdots, q_1(WS_{A_{in}})$. $WS_{A_{iu}}$ is said to dominance $WS_{A_{iv}}$, denoted $WS_{A_{iu}} \prec WS_{A_{iv}}$ as $WS_{A_{iu}}$ iff is better than or equal to $WS_{A_{iv}}$, $\forall k \in [1, d]$: $q_k(ws_{A_{iu}}) \le q_k(ws_{A_{iv}})$ and $\exists l \in [1, d] q_l(ws_{A_{iu}}) < q_l(ws_{A_{iu}})$ n all attributes and strictly better in at least one attribute, i.e..

If $ws_{A_{iv}}$ is neither dominated by nor dominates $ws_{A_{iu}}$, then $ws_{A_{iv}}$ and ws_{A_u} are incomparable. The notion of dominance handles requirement, since comparing matched services takes into consideration the degrees of match in all parameters, instead of calculating and using a single, overall score.

Definition 4. Skyline Web Service [22]. Skyline service of a service class WS, denoted by SWS, comprises those services in WS that are not dominated by any other services, i.e., $SWS = \{ws_i \in WS \mid \neg ws_j \in WS : ws_j \succ ws_i\}$.

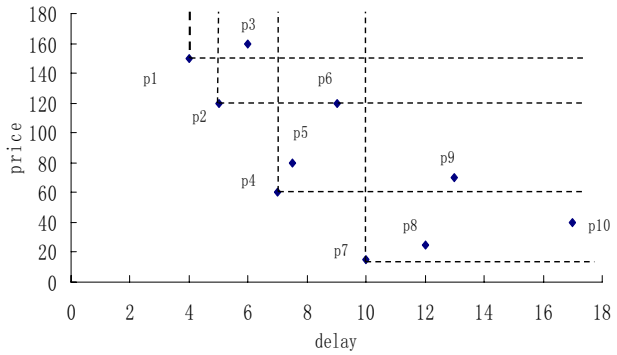

Fig. 2. Skyline Services

We observe that only those services that belong to the SWS are not dominated by any other functionally equivalent service, are valid candidates for the composition. This provides a valid pruning of the number of candidate services. Fig. 1 shows an example of skyline services of candidate services for a certain activity. Each service is described by two QoS attributes, namely delay and price. Hence, the services are represented as points in the 2-dimensional space, with the coordinates of each point corresponding to the values of the service in these two parameters. SWS includes four elements, SWS=$\{p_1,p_2,p_4,p_7\}$, because they are not dominated by any other service. On the other hand, service p6 is not contained in the SWS, because it is dominated by the services p_2 and p_4 [22].

The skyline Web services provide different trade-offs between the QoS attributes, and are incomparable to each other, as long as there is no pre-specified preference scheme regarding the relative importance of these attributes. For example, for a specific user, service a may be the most suitable choice, due to its very low delay and despite its high price, while for the other user, service may be the most preferred one due to its low price.

Consider an n-dimensional QoS attributes data space. We use architecture proposed [19] to achieve distributed parallel QoS selection effectively. The candidate services of predicting reliability are divided into data blocks. The data blocks are dispatched to many slaver servers for parallel processing. Then, the local skylines from service data points are generated in subdivided data blocks.

In this process, local skylines generated by all slaver servers are merged and integrated into a global skyline [19]. Skyline algorithm is the key in optimal QoS selection. We used a new skyline algorithm based on aR-tree index, and The index tree is traversed to judge whether it is dominated by the candidate Skyline sets · These are important difference from [19].

Local skyline service selection. Any data block dispatched to slaver server is denoted D, and D is n-dimensional data set. (D is actually QoS property set of candidate service, and n is the dimension of QoS space). p is a point, $p \in D, p = (id, p_{[1]}, p_{[2]}, \cdots, p_{[d]})$.

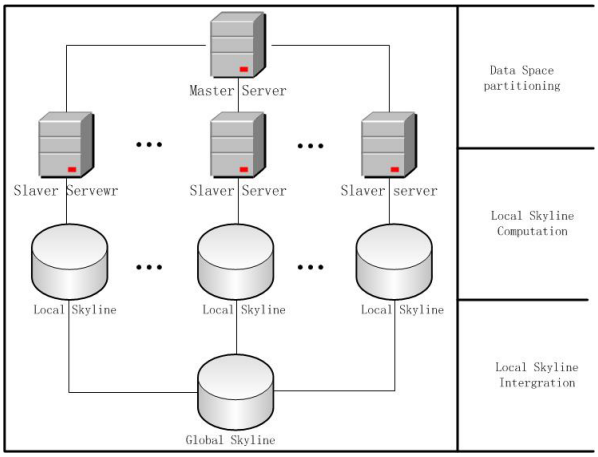

Fig. 3. Architecture of distributed parallel QoS selection [19]

Definition 5. Domination set D({p}). $p \in D$, D({p}) is services set dominated by service p.

Definition 6. Skyline score $|D(p)|$. $|D(p)|$ is defined number of points in set D({p}).

Definition 7. Skyline coverage distance d(p,rp). d(p,rp) expresses the smallest distance between not skyline representative point of Web service and the nearest skyline representative point of Web service.

Definition 8. Evaluation function of skyline service $F(\Psi(k), S)$. $F(\Psi(k), S) = d(p, rp) \cdot |D(p)|$. $\Psi(k)$ is skyline service set and S is candidate service set.

The local candidate services of each partition are then computed by using $F(\Psi(k), S)$. The local skyline service set is {max($F(\Psi(k), S)$). Local skyline service selection will implement when the local skyline service set of every data block is produced.

Global skyline service selection. There are two steps in the process of global skyline service selection: (1) Transmitting local skyline services from slave servers to one centralized server through network connection; (2) Computing the global skyline by merging local skyline services [19]. Due to the existence of dominance relationship between local skyline services from different slave servers, some local skyline services could be filtered by appropriate skyline services from different servers. We use an approximate solution based on R-tree index to implement global skyline service selection. In centralized server, n local skyline service sets are stored by R-tree. The index tree is traversed to judge whether it is dominated by the candidate Skyline service sets. We use the R-tree because of its ubiquity in multidimensional indexing and its use in other static-data skyline algorithms such as [23].

Algorithm 1. GSSS(k,S)

```
Input: S the set of all local skyline service sets
     K number of slave servers
Output: Ψ(k) global skyline service set
1: int index=0;
2: initialize root entry of R-tree L=n;
3:for all local skyline service set s_i ∈ S do
4:  E=max(F(s_i))
5:  if (E is not dominated by any local skyline service s_j form other
slave servers) then
6:   for all children of E_c of E do
7:       if (is not dominated by any local skyline service s_j form
other slave servers) then
8:       Ec→L;
9:     end if
10:  end for
11:  if (E is leaf point) then
12:    insert p to Ψ(k)
13:    end if
14:   end if
15: end for
```

As seen in Algorithm1, list L is maintained, which stores all local skyline services by exploiting R-tree index structure. For local skyline services, maximum of evaluation function of skyline service is found to judge if local skyline service is dominated by local skyline services of other slave servers. This can prune the local skyline services early with maximum evaluation function but are dominated by other local skyline services. These local skyline services will be removed from L. E in line 4 expresses some local skyline service with maximum evaluation function in L. Algorithm1 can finally compute all local skyline service sets S and find global skyline services which is put in $\Psi(k)$.

5 Experimentation

We can use skyline Web services in SWS as new candidate services which are variables for QoS aggregation function. On that basis, we can calculate utility value. Finally, we select Qos-based optimal service that maximizes the overall utility value from SWS.

In the section, we use two scenarios to evaluate the effectiveness and the efficiency of our approach. In the first scenario, different services are generated to implement the activities of example workflow. The result shows that the composite services which are composed by the selected component services not only are executed correctly and reliably but have optimal QoS. In the second one, we use the OWL-S service retrieval test collection OWLS-TC v22. The execution time of QoS services selection with skyline computation is compared with that without skyline computation.

For the first scenario, we use the OWL-S service retrieval test collection OWLS-TC v22. This collection contains services retrieved mainly from public IBM UDDI registries, and semi-automatically transformed from WSDL to OWL-S. We apply skyline to select the best candidates for QoS selection. We compare execution time of QoS selection using skyline computation with the time without using it. Fig. 4 illustrates the running time of QoS selection with (and without) skyline computation. Observe that the time without using skyline computation is higher using it.

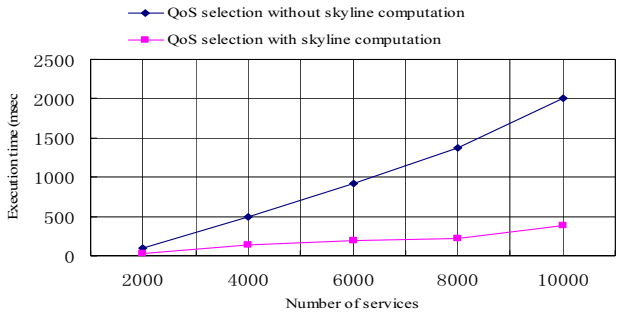

Fig. 4. Execution time

The second scenario is implemented as follow. In order to evaluate the behavior of our service selection approach, we write program whose input is a workflow composed of n activities and the output is a TCWS corresponding to a list of elementary Web services or composite Web services assigned to each activity of the input workflow. Experiments were conducted by implementing the proposed service selection approach with the program on a PC Core i3 with 2GB RAM, Windows 7, and Java 2 Enterprise Edition V1.5.0. The experiments involved composite services varying the number of activities and varying the number of Web services. The example in this paper is based upon a travel scheduling service composition which is depicted by state diagram in Fig. 5 and petri net in Fig. 6. We select twelve atomic Web services for composite Web service, and carried out the experiment for ten groups. As the comparison, we also use Nelson model [24-27] to calculate the reliability of the example. The formula to calculate the reliability is $R = 1 - \sum_{i=1}^{m} f_i / n_i P(E_i)$, where f_i denotes the number of failures out of n_i runs, and $P(E_i)$ is the probability that the input datum is taken from the subspace E_i in usage. In our case, E_i; i=1,2,...10 represent test cases. We assume that the every test case has the same chance to be executed. Then, the reliability of our reliability prediction approach is shown by curve EA in Fig. 7 and the reliability of Nelson model is shown by curve HA in Fig. 7. The difference between two reliabilities is shown in Fig. 7, which is very weak. Thus, we can conclude that our method to calculate the reliability of service composition is reasonable.

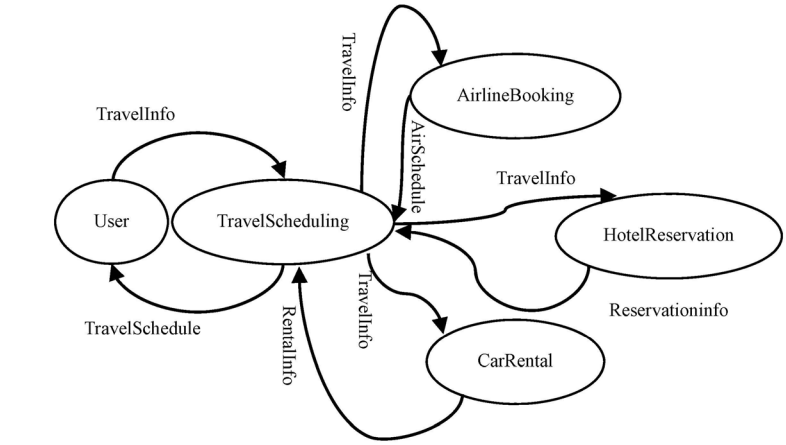

Fig. 5. Illustrative state diagrams for travel scheduling

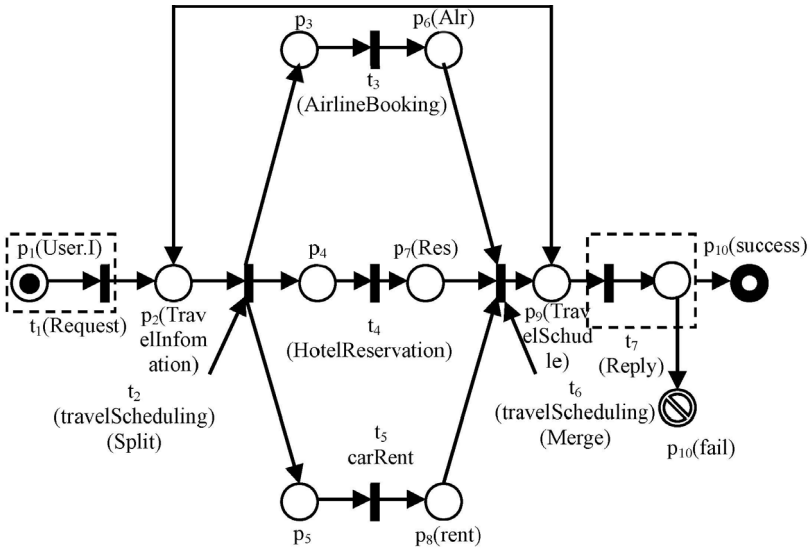

Fig. 6. TCWS-CPN for travel scheduling

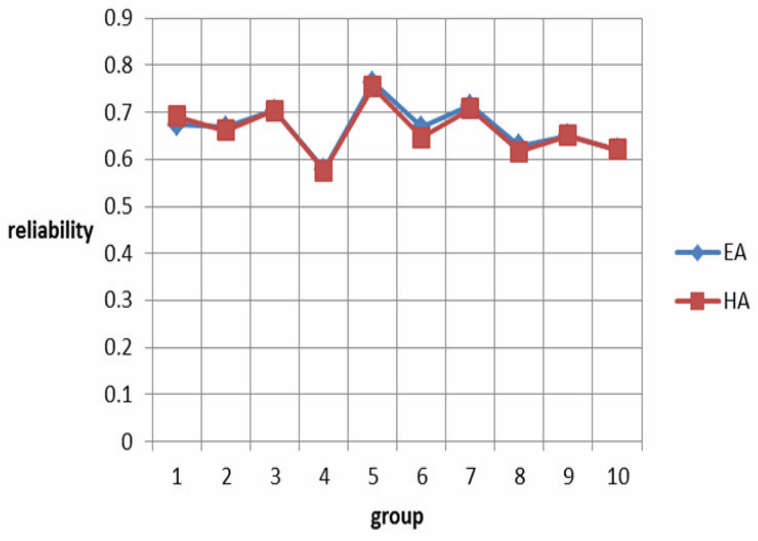

Fig. 7. Ten groups in HA and EA

6 Conclusions

In this paper, to capture the semantics of complex Web services combinations and their respective interactions we defined a Petri net extended with QoS properties and give the semantics for quality Petri net in terms of quality timed transition model. We also propose a consistence verification approach based on the Petri net which can detect the logical inconsistence of the semantic Web service process before the deployment, enhancing the robust of the process and the user's satisfaction. We propose a reliability prediction model based on Petri net. For atomic services, a staged reliability model is provided which predict reliability from network environment availability, service availability of hermit equipment, discovery reliability and binding reliability. To address the complex connecting relationship among subservices, places of basic Petri net for Input and Output are extended to some subtypes for multi-source input place and multiuse output place. Moreover, we use skyline computation to select services for composition effectively and efficiently, reducing the number of candidate services to be considered. We also define aggregation functions, and use a Multiple Attribute Decision Making approach for the utility function to achieve QoS-based optimal service selection. Experimental evaluation on real and synthetic data shows the effectiveness and efficiency of the proposed approach.

Web services in SWS as new candidate services which are variables for QoS aggregation function and select Qos-based optimal service that maximizes the overall utility value from SWS. As shown by experiment results, our approach is quadratic in term of selection and service size, in the majority of cases, which is the best solution in terms of QoS.

Our future work will focus on partial user Query satisfaction, using message-oriented methods. Additionally, we are working on failure recovery of TCWS execution considering transactional properties and techniques for Web service enforcement.

Acknowledgements

This work is partially supported Shaanxi education department foundation of China, No 16JK1273, by the Education Reform Project of Weinan Normal University, No. JG201624.

References

[1] Z.H. Ding, M.Y. Jiang, A. Kandel, Port-based reliability computing for service composition, IEEE Transactions on Services Computing 5(3)(2012) 422-436.

[2] R.H. Reussner, H.W. Schmidt, I.H. Poernomo, Reliability prediction for component-based software architectures, Journal of System and Software 66(3)(2003) 37-48.

[3] H.J. Zo, D.L. Nazareth, H.K. Jain, Measuring reliability of applications composed of Web services, in: R.H. Sprague (Ed.), Proceedings of the 40th Hawaii International Conference on System Sciences, HICSS-40, Waikoloa, HI, USA, January 29, 2007, IEEE Computer Society, 2007, pp. 1530-1605.

[4] A. Arsanjani, B. Hailpern, J. Martin, P. Tarr, Web wervices: promises and compromises, ACM Queue, March 1(1)(2003) 48-58.

[5] L.J. Jin, V. Machiraju, A. Sahai, Analysis on service level agreement of Web services, HP Labs Report HPL-2002-180, HP Laboratories, 2002.

[6] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, H. Chang, Quality-aware middleware for Web service composition, IEEE Trans. Softw, Eng. 30(5)(2004) 311-327.

[7] Y. Liu, A.H.H. Ngu, L. Zeng, Qos computation and policing in dynamic web service selection, in: Proc. the Thirteenth International World Wide Web Conference, 2004.

[8] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q.Z. Sheng, Quality driven web services composition, in: Proc. the

Twelfth International World Wide Web Conference, 2003.

[9] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, Qos-aware middleware for web services composition, IEEE Trans. on Software Engineering 30(5)(2004) 311-327.

[10] G.L. Nemhauser, L.A. Wolsey, Integer and combinatorial optimization, COR@L Technical Report 12T-020, 1988.

[11] I. Maros, Computational Techniques of the Simplex Method, Kluwer Academic, Norwell, MA, 2003.

[12] D. Skoutas, D. Sacharidis, A. Simitsis, K. Timos Sellis, Serving the sky: discovering and selecting semantic Web services through dynamic skyline queries, in: Proc. the 2008 IEEE International Conference on Semantic Computing, 2008.

[13] A. Immonen, D. Pakkala, A survey of methods and approaches for reliable dynamic service compositions, Service Oriented Computing and Applications 8(2)(2014) 129-158.

[14] C.L. Xie, J.G. Ren, A dynamical reliability prediction algorithm for composite service. <https://www.hindawi.com/journals/mpe/2014/917903/abs/>, 2014.

[15] R. Hamadi, B. Benatallah, A Petri net-based model for web service composition, in: D. Klaus, X.F. Zhou (Eds.), Proceedings of the 14th Australasian database conference, ADC 2003, Adelaide, South Australia, February 2003, CRPIT 17, Australian Computer Society, Adelaide, Australia, 2003, pp. 191-200.

[16] Y. Cardinale, J.E. Haddad, M. Manouvrier, M. Rukoz, Web service composition based on petri nets: review and contribution, in: Proc. International Workshop on Resource Discovery, 2013.

[17] P.C. Xiong, Y.S. Fan, M.C. Zhou, A Petri net approach to analysis and composition of web services, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions 40(2)(2010) 376-387.

[18] Y.D. Lin, Key technology of semantic Web service composition and its application, [PhD thesis] Guangzhou, China: South China University of Technology, 2013.

[19] D. Papadias, Y. Tao, G. Fu, B. Seeger, An optimal and progressive algorithm for skyline queries, in: Y.H. Alon, G.I. Zachary, D. AnHai (Eds.), Proceedings of ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, ACM, 2003, pp. 467-478.

[20] M. Xie, Software reliability models: past, present and future, in: N. Limnios, M. Nikulin (Eds.), Recent Advances in Reliability Theory, Statistics for Industry and Technology, Birkhäuser, Boston, MA, 2000, pp. 55-90.

[21] M. Morse, J.M. Patel, W.I. Grosky, Efficient continuous skyline computation, Information Sciences 177(2007) 3411-3437.

[22] M. Alrifai, D. Skoutas and T. Risse, Selecting skyline services for QoS-based web service, Raleigh, in: Proc. the 19th International Conference on World Wide Web, 2010.

[23] J. Wu, L. Chen, Q. Yu, L. Kuang, Y.L. Wang, Z.H. Wu, Selecting skyline services for QoS-aware composition by upgrading MapReduce paradigm, Cluster Comput 16(2013) 693-706.

[24] D. Kossmann, F. Ramsak, S. Rost, Shooting stars in the sky: an online algorithm for skyline queries, in: A. Bernstein, Y.E. Ioannidis, R. Ramakrishnan, D. Papadias (Eds.), Very Large Databases (VLDB), Hong Kong SAR, China, 2002, pp. 75-286.

[25] W. Hussein, T. Peng, G.J. Wang, A weighted throttled load balancing approach for virtual machines in cloud environment, Int. J. of Computational Science and Engineering 11(4)(2015) 402-408.

[26] Z.Y. Shao, B. Yang, Proof of retrievability with efficient verification, Int. J. of Embedded Systems 7(3-4)(2015) 203-215.

[27] R. Eswari, S. Nickolas, Effective task scheduling for heterogeneous distributed systems using firefly algorithm, Int. J. of Computational Science and Engineering 11(2)(2015) 132-142.