International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 29, No. 2 (2015) 1559002 (16 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218001415590028

Conformance Checking and QoS Selection Based on CPN for Web Service Composition

Weitao Ha*, Guojun Zhang† and Liping Chen‡

College of Communication Engineering
Weinan Normal University
Weinan, 714000, P. R. China
**wnhwt@126.com*
†junzg@126.com
‡wnchlp@126.com

Received 23 May 2014
Accepted 16 October 2014
Published 7 January 2015

The development of new services by composition of existing ones has gained considerable momentum as a means of integrating heterogeneous applications and realizing business collaborations. More and more Web services with similar function attributes but different QoS are available. The performance of the composed service is determined by the involved Web services. Therefore, QoS properties are crucial for selecting the Web services to take part in the composition, which can identify the best candidate Web services from a set of functionally-equivalent services. Web service composition enables seamless and dynamic integration of business applications on the web. Due to the inherent autonomy and heterogeneity of component Web services, it is difficult to predict the behavior of the overall composite service. Conformance checking identifies failure and conflict of execution of composite Web service that ensures reliable execution. In this paper we use skyline computation to select services for composition efficiently, reducing the number of candidate services to be considered. Then a novel color Petri net model of Web service composition is presented that combines QoS-based optimal service selection and consistence verification. In the model we define aggregation functions, and use a Multiple Attribute Decision Making approach for the utility function to achieve optimal services selection of QoS properties. We also propose a consistence verification approach to identify potential logical inconsistence of the semantic Web service process before the deployment. Proofs are also presented. We evaluate our approach experimentally using both real and synthetically generated datasets.

Keywords: Service composition; web service selection; skyline; QoS optimization; web service consistence verification.

1. Introduction

Web Services are distributed applications that interoperate across heterogeneous networks and provide services that are hosted and executed on remote systems.

†Corresponding author.

Service oriented architecture (SOA) is gaining prominence as a key architecture to support Business Process Management (BPM) and integrate applications in diverse and heterogeneous distributed environments. Web services composition is becoming an efficient and cost-effective way to develop modern business applications, and it aggregates some existing Web services which are developed by different organizations and offer diverse functional, transactional, and nonfunctional properties.[1]

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) QoS selection of the component Web services, (3) consistence verification of composite Web service, and (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, to fulfill the user's QoS goal the component Web services are selected. 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. 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.[4] In this paper we focus on the QoS selection and consistence verification which could be dynamic and automatic and not finding and the execution step or on the recovery or replanning problems.

The research goal for Web service selection is to design a composite Web service to ensure not only correct and reliable execution but also optimal QoS. As a tremendous amount of different QoS Web services with same functions are spread all over the Internet, it is intractable to find the appropriate Web services satisfying the given goal quickly. What is more, using traditional methods, services with only one bad QoS attribute may be excluded from the result set, even though they are potentially good alternatives, and thus leads to information loss that significantly affects the retrieved results accuracy. But skyline computation is a nondiscriminating comparison of several numerical attributes at the same time and treats each service equally. In our current approach, we use skyline computation to define and compute the potentially interesting services for any type of user, which reduce the number of candidate services, and speed up the selection process.

We find that the color Petri net model allows describing not only a static vision of a system, but also 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 defined a colored Petri net extended with QoS properties and give the semantics for quality colored Petri net in terms of quality-timed transition model. We also propose a consistence verification approach based on the color 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 define aggregation functions, and use a Multiple Attribute Decision Making approach for the utility function to achieve QoS-based optimal service selection. Using the color Petri net achieves reliable execution and optimal QoS of composite Web service.

2. Related Work

In the last few years, although the problem of Web service selection and composition has received much attention of many researchers, designing a composite Web service which ensures not only correct and reliable execution but also optimal QoS remains an important challenge. Indeed, these two aspects of selection are always implemented separately.

Web services transactions have received much attention recently. Industrial Web services and transaction specifications emerge. WS-Atomic Transaction, WS-Business Activity and WS-TXM rely on ATM to define transactional coordination protocols. Like ATM, these protocols are unable, in most cases to model Business process due to their limited control structure. It also ensures reliability on behalf of process adequacy or the opposite. Indeed, a transactional pattern taken alone as a composition of transactional patterns can be considered as a transactional protocol.

On the one hand, WS-BPEL and WS-CDL follow a workflow approach to define services compositions and services choreographies.[8] Like workflow systems these two languages meet the business process need in term of control structure. However, they are unable to ensure reliability especially according to the designers' specific needs.

Transaction has achieved a great success in the database community.[5,19] One of the most important reasons is that the operations in database have clear transactional semantics. However, this is not the case in Web services. To solve this problem, the extension mechanism of WSDL can be exploited to explicitly describe the transactional semantics of Web services operations.[10,13]

There are many works to adopt three kinds of transactional properties proposed in Ref. 9 to express the different transactional semantics of Web services. Based on this classification, Vidyasankar and Vossen[16] analyze the termination property of a composite service. Bhiri *et al.*[3] define a set of transactional rules to verify the required failure atomicity specified by ATS,[14] given that the skeleton of a composite service and the transactional properties of its component services. Montagut and Molva[11] propose an approach to deduce the required transactional properties of every task based on ATS and then use the result to guide service selection.

For these researches Web services composition based on transactional properties ensures a reliable execution, however, an optimal QoS composite Web service is not guaranteed.

QoS guarantee for Web services is one of the main concerns of the SLA framework. There are projects studying QoS-empowered service selection. In Ref. 21, 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 Ref. 6, 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 *et al.*[20,21] 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[12] 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.[7] 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 Ref. 15. 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, transactional-based and QoS-based should be integrated.

3. Candidate Web Services Screening Based on Skyline Computation

3.1. *Basic concept*

The basic skyline consists of all nondominated database objects. That means all database objects for which there is no object in the database that is better or equal in all dimensions, but in at least one aspect is strictly better. Assuming every database object to be represented by a point in n-dimensional space with the coordinates for each dimension given by its scores for the respective aspect, we can formulate the problem as:

The Skyline Problem: Given set $O := \{o1, \dots, oN\}$ of N database objects, n score functions with si : $O \rightarrow [0, 1]$ and n sorted lists $S1, \dots, Sn$ containing all database objects and their respective score values using one of the score function for each list; all lists are sorted descending by score values starting with the highest scores. Wanted is the subset P of all nondominated objects in O, i.e. $\{o_i \in P \,|\, \neg\exists o_j \in O : (s_1(o_i) \leq s_1(o_j) \wedge \cdots \wedge s_n(o_i) \leq s_n(o_j) \wedge \exists q \in [1, \ldots, n] : s_q(o_i) < s_q(o_j))\}$.

3.2. *Screened skyline web services*

QoS-based service composition is a constraint optimization problem which aims at selecting individual services that meet QoS constraints and also provide the best value for the utility. For a composite Web service with n activities and l candidate services per activity, there are impossible combinations to be examined. Hence, 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.[17] Skyline computation offers a new solution of finding optimal data from huge datasets, whose computation can be expensive and whose applications require fast response times.

The main idea in our approach is to perform a skyline query on the services in each activity to distinguish between those services that are potential candidates for the composition, and those that cannot possibly be part of the final solution. The latter can effectively be pruned to reduce the search space.

Definition 1. Dominance. Given a service set S_{Ai} assigned to activity Ai having n candidate services: $S_{A_{i1}}, S_{A_{il}}, \ldots, S_{A_{\mathrm{in}}}$. QoS vector is d dimensions: $q_1(S_{A_{i1}}), q_2(S_{A_{i1}}), \ldots, q_d(S_{A_{il}}) S_{A_{iu}}$ is said to dominance $S_{A_{iv}}$, denoted $S_{A_{iu}} \prec S_{A_{iv}}$ as iff $S_{A_{iu}}$ is better than or equal to in all $S_{A_{iv}} \forall k \in [1, d] : q_k(S_{A_{iu}}) \leq q_k(S_{A_{iv}})$ and $\exists l \in [1, d] : q_k(S_{A_{iu}}) < q_k(S_{A_{iv}})$ attributes and strictly better in at least one attribute, i.e.

If $S_{A_{iv}}$ is neither dominated by nor dominates $S_{A_{iu}}$, then $S_{A_{iv}}$ and $S_{A_{iu}}$ 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.[18]

Definition 2. Skyline Web Services.[2] The skyline Web services of a service set S_{Ai}, denoted by SWS, comprises the set of those services that are not dominated by any other $\{S_{A_{iu}} \in S_{A_i} | \neg\exists S_{A_{iv}} \in S_{A_i} : S_{A_{iv}} \prec S_{A_{iu}}\}$ services, i.e. SWS = Services in SWS are skyline Web services of a service set S_{Ai}.

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. Figure 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 two-dimensional space, with the coordinates of each point corresponding to the values of the service in these two parameters. SWS includes four elements, SWS = $\{p1, p2, p4, p7\}$, because they are not dominated by

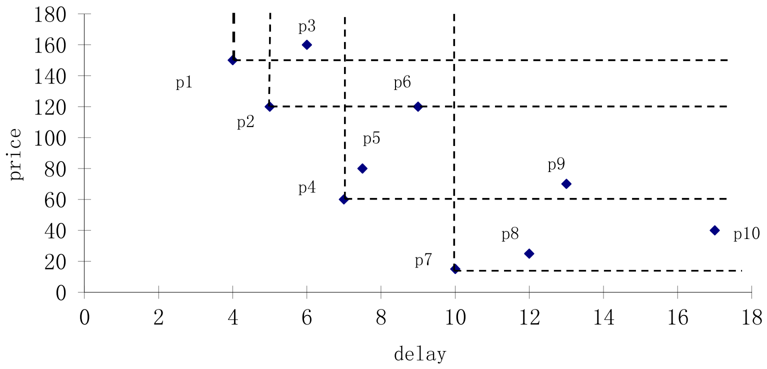

Fig. 1. Skyline services.

any other service. On the other hand, service $p6$ is not contained in the SWS, because it is dominated by the services $p2$ and $p4$.[2]

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 b may be the most preferred one due to its low price.

3.3. *Candidate web services screening algorithm*

0. Initialize a datastructure SWS $:= \phi$ containing records with an identifier and n real values indexed by the identifiers, initialize n lists $K_1, \ldots, K_n := \phi$ containing records with an identifier and a real value, and initialize n real values $p_1, \ldots, p_n := 1$

```
1.   Initialize counter $i := 1$.
1.1. Get the next object $S_{A i\text{new}}$ by sorted access on list $S_{Ai}$
1.2. If $S_{A i\text{new}} \in$ SWS, update its record's $i$th real value with $S_{Ai}(S_{A i\text{new}})$, else create
     such a record in SWS
1.3. Append $S_{A_{i\text{new}}}$ with $S_{Ai}(S_{A i\text{new}})$ to list $K_i$
1.4. Set $pi := S_{A i\text{new}}$ and $i := (i \bmod n) + 1$
1.5. If all scores $S_{Ai}(S_{A i\text{new}})(1 \leq i \leq n)$ are known, proceed with step 2 else with
     step 1.1.
2.   For $i = 1$ to $n$ do
2.1. While $pi = S_{Ai}(_{S_{A i\text{new}}})$ do sorted access on list $S_{Ai}$ and handle the retrieved
     objects like in step 1.2 to 1.3
3.   If more than one object is entirely known, compare pairwise and remove the
     dominated objects from SWS.
4.   For $i = 1$ to $n$ do
4.1. Do all necessary random accesses for the objects in $K_i$ that are also in SWS,
     immediately discard objects that are not in SWS
```

4.2. Take the objects of K_i and compare them pairwise to the objects in K_i. If an object is dominated by another object remove it from K_i and SWS
5. Output SWS as the set of skyline Web services

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.

4. A Color Petri Net with QoS Selection

4.1. *New web service ontology description*

OWL-S is an OWL-based Web service ontology, which supplies a core set of markup language constructs for describing the properties and capabilities of Web services in unambiguous, computer-interpretable form. The typical Web service ontology consists of three main components: the service profile, the process model, and the grounding which is shown in Fig. 2. The service profile describes what the service does by specifying the input and output types, preconditions and effects for advertising and discovering services. The process model gives a detailed description of a service's operation. The grounding contains the details of how an agent can access a service by specifying a communication protocol, parameters to be used in the protocol and the sterialization techniques to be employed for the communication. Specifically, it specifies the signature that is the inputs required by the service and the outputs generated; furthermore, since a service may require external conditions to be satisfied, and it has the effect of changing such conditions, the profile describes the preconditions required by the service and the expected effects that result from the execution of the service.

From service discovery and composition perspective, the typical Web ontology has the following shortcomings. First, hierarchical management is not implemented for Web service. That will cause low efficiency of Web service discovery. Second, a huge number of candidate services are discovered according to the service profile of typical ontology. When these candidate services are composed by model driven approach, searching state space may be so large as to be unusable.

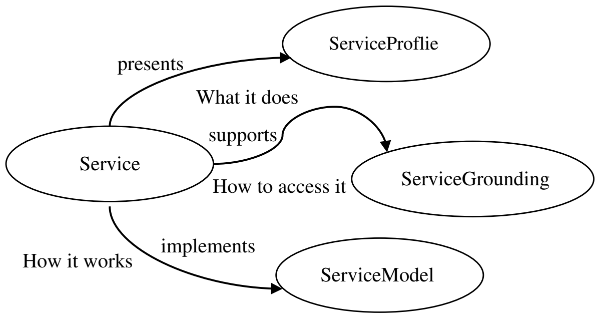

Fig. 2. Typical web service ontology.

In this paper, a new Web service ontology is proposed. The formalization definition of the new ontology is given as follows.

Definition 3. New Web Service Ontology. The new Web service ontology is a triple (SBA,SFI,QoS). SBA includes all of service basic attributes, as ID, name, URL of service description file, function, domain, provider information, etc. SFI is a set of interfaces. An interface is expressed as quad (in, pre, out, eff), where "in" is the set of input parameters; "pre" is interface precondition; "out" is the set of output parameters, and "eff" is execution result. QoS is expressed as nonfunctional attributes.

4.2. *A color QoS petri net*

Definition 4. Color QoS Petri Net (CQPN). The color QoS Petri net is a 13-tuple (Σ, P, T, A, N, ξ, G, B, α, β, γ, μ, IN), where:

Σ is a finite nonempty set of color $\Sigma = \{\text{SFI} \times \text{QoS}\}$,
P is a finite nonempty set of places,
T is a finite nonempty set of transitions $P \cup T = \emptyset$,
A is a flow relation $A \subseteq (P \times T) \cup (T \times P)$ for the set of arcs,
N is node function $N : A \to P \times T \cup T \times P$,
C is color function $C : P \to \Sigma$,
G is guard function $G : T \to \text{Expression}$, $\forall t \in T : [\text{Type}(G(t)) = \text{Bool} \wedge \text{Type}(\text{Var}(G(t))) \subseteq \Sigma$.

Expression is a functional expression; Type$(G(t))$ expresses type of $G(t)$; Var $(G(t))$ expresses the set of variables; Type(Var$(G(t))$) expresses type set of all of variables in function $G(t)$,

E is arc function $E : A \to \text{Expression}$, $\forall a \in A : [\text{Type}(E(a)) = C(p(a))\text{MS} \wedge \text{Type}(\text{Var}(E(a))) \subseteq \Sigma]$, and $C(p(a))$MS expresses multiple color set of place p of arc a,
$\alpha : T \to Q+$ is minimum time that is spent by firing transition,
$\beta : T \to (Q + \cup\{\infty\})$, $\alpha \le \beta$ is maximum time that is spent by firing transition,
$\gamma : T \to \text{Expression}, \forall t \in T : [\text{Type}(\gamma t(y)) = Q \wedge y \in [\alpha, \beta]]$, and γ is cost function of transition in the interval $[\alpha,\, \beta]$of delay time,
$\mu : T \to \text{SP}$, $\text{SP} = [0, 1]$ is the probability of transition executing,
IN is an initialize function.

Definition 5. A marking M is a vector $(M(p1), M(p1), M(p1), \ldots, M(p1))$, $\forall p \in P : M(p) \in C(p)\text{MS}$.

Definition 6. Enabling Condition.

$\forall \mu \in \text{Var}(G(t) : G(t)\langle b(\mu)\rangle = \text{true}$, $b(\mu) \in C(p)$, $p \in \{p \in P | \exists a \in A : N(a) = (p, t)\}$
$\forall p \in P : E(p, t) \le M(p)$.

When the two conditions are met transition t will enable under marking M, which is indicated $M[t\rangle$.

Definition 7. Firing Rule. When $\alpha(t) \leq d(t) \leq \beta(t)$ is true, transition t is fired. The fired result is $\forall p \in \{p \in P | \exists a \in A : N(a) = (p,t) \vee N(a) = (t,p)\} : M' = M(p) - E(p,t) + E(t,p)$, which is indicated $M[t\rangle M'$.

4.3. *Consistence verification*

(1) Conflict Detection of Parallel Component Service
Let transition t_1, t_2 correspond service $S_1, S_2, \exists p_1, p_2 \in P$, $(p_1,t_1) \wedge (p_2,t_2) \in A$, $C(p_1) = (\mathrm{SBA}_1, (\mathrm{in}_1, \mathrm{pre}_1, \mathrm{out}_1, \mathrm{eff}_1), \mathrm{QoS}_1) \wedge C(p_2) = (\mathrm{SBA}_2, (\mathrm{in}_2, \mathrm{pre}_2, \mathrm{out}_2, \mathrm{eff}_2), \mathrm{QoS}_2)$, and S_1 and S_2 are implemented parallel if and only if $(\mathrm{pre}_1 \wedge \mathrm{pre}_2 = \emptyset) \wedge M[t_1\rangle \wedge M[t_2\rangle$.

(2) Verification For Component Service Performability
In order to make service execution, it is first reachable. If transition t enables under marking $M(M[t >)$, service S corresponded to transition t is reachable.

Definition 8. Component Service Performability. Component service is performability if and only if pre of S is satisfied, and service S corresponded to transition t is reachable.

In order to analyze performablility of service S, we need judge if current state under marking M meets precondition of service S. We compile verification algorithm of component service performability as follow:

Input: service ontology, color QoS Petri net, initial state S_0, a set of semantic Web services WS;

Output: true or false;

```
1. set $p\mathrm{slist}(M_0) = S_0$;
2. set openlist $= \{M_0\}$;
3. while(openlist! $=$ NULL)
   (a) {retrieve an element $m$ from openlist;
   (b) add$\{n | n \in M \wedge (m,n) \in A$ to openlist;
   (c) For each $m' \in \{n | n \in M \wedge (m,n) \in A\}$ do
   (d) {set $ws = N((m,m'))$;
   (e) For each $s \in ps\mathrm{list}(m)$ do
   (f) Add $s' = \mathrm{Lit} \cup \{l' | l' \in s \wedge \neg l' \notin \mathrm{Lit}\}$ to $ps\mathrm{list}(m')\}$;
   (g) Remove $m$ from openlist};
   (h) End while
4. For each $ws \in$ WS do
   (a) For each $(m,n) \in A$ do
   (b) For each $s \in ps\mathrm{list}(m)$ do
   (c) {if($ws^{\mathrm{pre}} \not\subset s$) return false};
5. return true.
```

4.4. *Optimal QoS selection based on color QoS petri net*

The QoS attributes of CWS are decided by the QoS attributes of individual services and workgroup patterns.

There are three workgroup patterns are: (a) Sequential pattern. (b) AND-split and AND-join patterns. (c) XOR-split and XOR-join patterns. Sequential pattern is the fundamental pattern, because other patterns can be converted into sequential model. We can find how to do the conversions in many published research. In our approach, we only consider QoS properties aggregation of the Sequential pattern.

The QoS vector for a $\mathrm{CWS} = \{S_{A_{1i}}, \ldots, S_{A_{nm}}\}$ is expressed with $\mathrm{QCWS} = \{q_1(\mathrm{CWS}), \ldots, q_r(\mathrm{CWS})\}$, where $q_i(CWS)$ is the estimated end-to-end value of the ith QoS attribute and can be computed by aggregating the corresponding values of the component services. In our paper, we consider three types of QoS aggregation functions: (1) summation, (2) multiplication, and (3) minimum relation.

(1) Summation aggregation function

Summation aggregation function is used for price and response time, which is defined as follow,

$$q'(\mathrm{CWS}) = \sum_{i=1}^{n} q(S_{A_{il}}), \tag{1}$$

where i is number of component services for a CWS that is composed of component services from each service class WSi.

Reputation aggregation function is defined as follow,

$$q'(\mathrm{CWS}) = 1/n \sum_{i=1}^{n} q(S_{A_{il}}). \tag{2}$$

(2) Multiplication aggregation function

Multiplication aggregation function can be used for availability and reliability,

$$q'(\mathrm{CWS}) = \prod_{i=1}^{n} q(S_{A_{il}}) \tag{3}$$

(3) Minimum relation aggregation function

Throughput aggregation function:

$$q'(\mathrm{CWS}) = \min_{i=1}^{n} q(S_{A_{il}}). \tag{4}$$

4.5. *Utility function*

As the QoS service selection is embedded within the transactional service selection, every service in service class $S_{Ai} = \{S_{Ai1}, S_{Ai2}, \ldots, S_{Ain}\}$ can be assigned to the same activity after transactional service selection. The set of potential Web services for each activity is restricted by the transactional requirement. Indeed, the selection of a

Web service for an activity depends on the transactional property of Web services already assigned to the previous activities of the workflow.

The utility function of composite web service is used for evaluating the multi-dimensional quality, which maps the quality vector QoS into a single real value. The utility computation involves scaling values of QoS property to allow a uniform measurement of the multi-dimensional service qualities independent of their units and ranges. A weighing process is developed which can obtain weight of representing user priorities and preferences before the scaling process. In the scaling process, QoS attribute value is transformed between 0 and 1, which is realized by comparing it with the minimum and maximum possible value according to the available QoS information about alternative services.

Utility Function of CWS UF(CWS) is computed as follows,

$$\mathrm{UF(CWS)} = \sum_{k=1}^{l} w_k \cdot \frac{\max q_k(\mathrm{CWS}) - q_k(\mathrm{CWS})}{\max q_k(\mathrm{CWS}) - \min q_k(\mathrm{CWS})}, \tag{5}$$

where l is dimension of quality vector, and w_k is the weight of q_k to represent priorities of QoS attributes. $\max q_k(\mathrm{CWS})$ or $\min q_k(\mathrm{CWS})$ is the maximum and minimum aggregated values of the k-th QoS attribute for a given composite service, and they are obtained as follows,

$$\max q_k(\mathrm{CWS}) = \mathrm{AF}(\max q_k(S_{A_1}), \max q_k(S_{A_2}), \ldots, \max q_k(S_{A_n})), \tag{6}$$

$$\min q_k(\mathrm{CWS}) = \mathrm{AF}(\min q_k(S_{A_1}), \min q_k(S_{A_2}), \ldots, \min q_k(S_{A_n})), \tag{7}$$

where AF is aggregation function of the kth QoS attribute, and $\max q_k(S_{A_i})$ or $\min q_k(S_{A_i})$ is the maximum or minimum value of kth QoS attribute of candidate services S_{Ai} assigned to activity Ai. They are computed as follows,

$$\max q_k(S_{A_i}) = \max(q_k(S_{A_{i1}}), q_k(S_{A_{i2}}), \ldots, q_k(S_{A_{im}})), \tag{8}$$

$$\min q_k(S_{A_i}) = \min(q_k(S_{A_{i1}}), q_k(S_{A_{i2}}), \ldots, q_k(S_{A_{im}})). \tag{9}$$

In my research QoS-based optimal service selection is realized, when the overall utility value UF(CWS) is maximized.

5. Experimentation

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-TCv22.

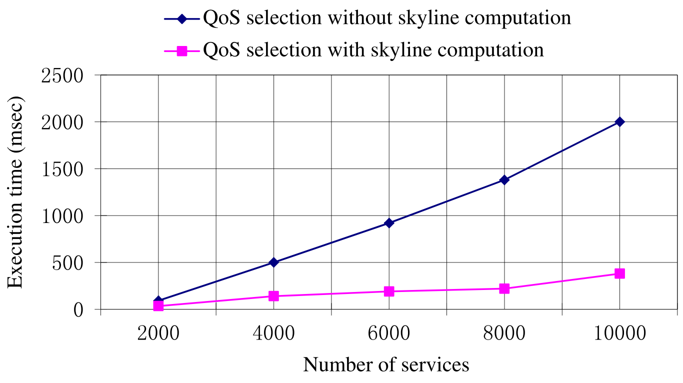

Fig. 3. Execution time.

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. Figure 3 illustrates the running time of QoS selection with (and without) skyline computation. Observe that the time without using skyline computation is higher using it.

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 2 GB RAM, Windows 7,

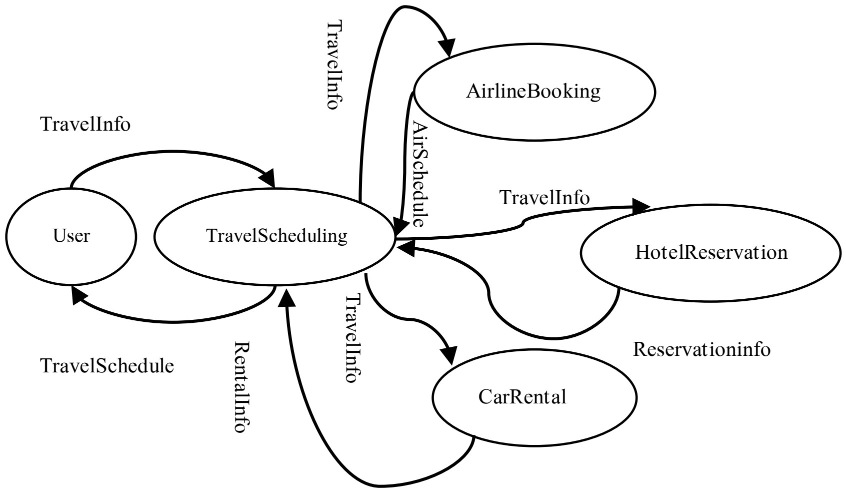

Fig. 4. Illustrative state diagrams for travel scheduling.

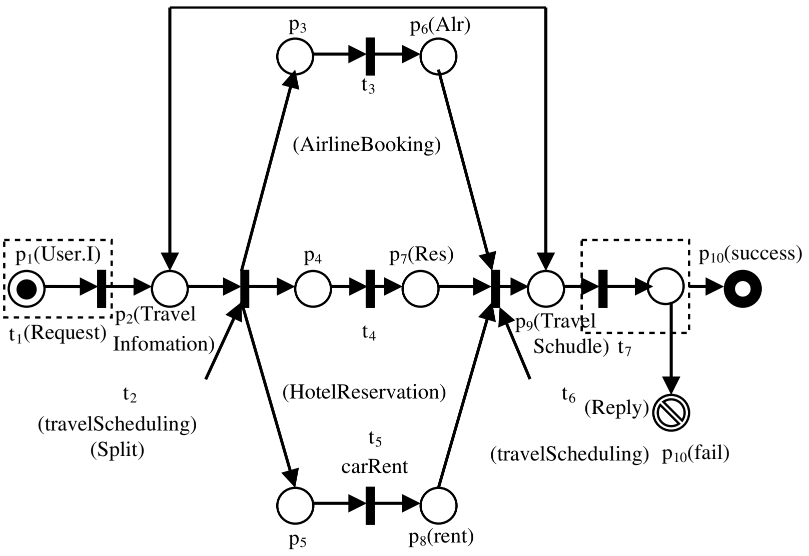

Fig. 5. TCWS-CPN for travel scheduling.

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 Figs. 4 and 5.

We select 12 atomic Web services for composite Web service, and carried out the experiment for ten groups in HA and EA and each group probability of transition executing is generated by random from 0.8 to 1.0. The results of experiment are shown in Table 1. The execution path selected by our approach can meet the requirement of users and Table 1 shows that with the increasing of the atomic Web services, the selection time is relatively stable.

Table 1. 10 groups in HA and EA.

Group	EA	HA
1	0.673	0.592
2	0.668	0.591
3	0.704	0.704
4	0.579	0.576
5	0.764	0.754
6	0.668	0.646
7	0.714	0.708
8	0.629	0.617
9	0.651	0.651
10	0.621	0.621
Average	0.667	0.645

6. Conclusion

In this paper, to capture the semantics of complex Web services combinations and their respective interactions we defined a colored Petri net extended with QoS properties and give the semantics for quality colored Petri net in terms of quality-timed transition model. We also propose a consistence verification approach based on the color 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 define aggregation functions, and use a Multiple Attribute Decision Making approach for the utility function to achieve QoS-based optimal service selection. Using the color Petri net achieves reliable execution and optimal QoS of composite Web service.

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 terms 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.

Acknowledgments

This work is partially supported by Shaanxi education Department foundation of China, No. 2013JK1119, by the projects of Weinan Teachers University, No. 13YKS006.

References

1. P. Albert, L. Henocque and M. Kleiner, Configuration-based workflow composition, in *Proc. 2005 ICWS* (2005), pp. 285–292.
2. M. Alrifai, D. Skoutas and T. Risse, Selecting skyline services for QoS-based web service composition, in *Proc. 19th Int. Conf. World Wide Web* (2010), pp. 88–101, ACM.
3. S. Bhiri, O. Perrin and C. Godart, Ensuring required failure atomicity of composite web services, in *Proc. Int. Conf. World Wide Web (WWW'05)* (2005), pp. 138–147.
4. J. El Haddad, M. Manouvrier and M. Rukoz, TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition, *IEEE Trans. Services Comput.* **3**(1) (2010) 73–85.
5. A. Elmagarmid, *Transaction Models for Advanced Database Applications* (Morgan-Kaufmann, 1992).
6. Y. Liu, A. H. H. Ngu and L. Zeng, QoS computation and policing in dynamic web service selection, in *Proc. Int. World Wide Web Conf.* (2004), pp. 66–73.
7. I. Maros, *Computational Techniques of the Simplex Method* (Springer, 2003).

8. B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu and A. K. Elmagarmid, Business-to-business interactions: Issues and enabling technologies, *VLDB J.* **12**(1) (2003) 59–85.
9. S. Mehrotra, R. Rastogi, A. Silberschatz and H. Korth, A transactional model for multidatabase systems, in *Proc. Int. Conf. Distributed Computing Systems (ICDCS'92)* (1992), pp. 56–63.
10. T. Mikalsen, T. Tai and I. Rouvellou, Transactional attitudes: Reliable composition of autonomous web services, in *Proc. Workshop on Dependable Middleware-Based Systems at the Dependable Systems and Network Conf.* (2002), pp. 113–122.
11. F. Montagut and R. Molva, Augmenting web services composition with transactional requirements, in *Proc. IEEE Int. Conf. Web Services (ICWS'06)* (2006), pp. 91–98.
12. G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization* (Wiley-Interscience, New York, USA, 1988).
13. P. F. Pires, M. R. F. Benevides and M. Mattoso, Building reliable web services compositions, in *Proc. Web, Web Services, and Database Systems, NODe Web and Database-Related Workshops* (2003), pp. 59–72.
14. M. Rusinkiewicz and A. Sheth, Specification and execution of transactional workflows, in *Modern Database Systems: The Object Model, Interoperability, and Beyond* (ACM Press/Addison-Wesley, 1995).
15. D. Skoutas, D. Sacharidis, A. Simitsis and T. K. Sellis, Serving the sky: Discovering and selecting semantic web services through dynamic skyline queries, *ICSC 2008* (2008), pp. 222–229.
16. K. Vidyasankar and G. Vossen, A multi-level model for web services composition, in *Proc. IEEE Int. Conf. Web Services (ICWS'04)* (2004), pp. 462–469.
17. Y. Wang and C. Dang, An evolutionary algorithm for global optimization based on level-set evolution and Latin squares, *IEEE Trans. Evol. Comput.* **11**(5) (2007) 579–595.
18. Y. Wang, Y.-C. Jiao and H. Li, An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme, *IEEE Trans. Systems, Man and Cybernetics C: Appl. Rev.* **35**(2) (2005) 221–232.
19. G. Weikum and G. Vossen, *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery* (Morgan-Kaufmann, 2002).
20. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam and Q. Z. Sheng, Quality driven web services composition, in *Proc. Int. World Wide Web Conf.* (2003), pp. 411–421.
21. L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam and H. Chang, Quality-aware middleware for web service composition, *IEEE Trans. Softw. Eng.* **30**(5) (2004) 311–327.

Weitao Ha received his M.S. degree in Computer Application Technology from Xidian University, Xi'an, China, 2008. He received his B.S. degree in Computer Education from Northwest Normal University, Lanzhou, China, 1999. Currently he is an Associate Professor in the College of Communication Engineering, Weinan Normal University, China. His research interests include service computing technique.

Guojun Zhang received his M.S. degree in Computational Mechanics from Northwestern Polytechnical University, Xi'an, China, 1986. He received his B.S. degree in Computational Mechanics from Northwestern Polytechnical University, Xi'an, China, 1983. Currently he is a Professor in the College of Communication Engineering, Weinan Normal University, China. His research interests include service computing technique.

Liping Chen received her M.S. degree in Computer Application Technology from Xidian University, Xi'an, China, 2006. She received her B.S. degree in Computer Application Technology from Northwest Normal University, Lanzhou, China, 1999. Currently she is an Associate Professor in the College of Mathematics and Information Science, Weinan Normal University, China. Her research interests include intelligent service composition.