An Algorithm for Semantic Web Services Composition Based on Output and Input Matching

Ying Li and Baotian Dong
School of Traffic and Transportation, Beijing Jiaotong University, Beijing, 100044, China
ling.ly@sohu.com

Abstract. Existed methods for automatic web services composition based on output and input matching are limited to deal with simple Composition Request (CR) which can only be satisfied by linear composition plan, but complicated CR asks for netty plan because the structure of netty plan can be complicated enough to deal with complicated problem while the structure of linear plan is too simple to do that. In order to overcome the shortcoming of the existed methods, a new algorithm is proposed in this paper which can deal with not only simple CR but also complicated CR. A web services connection matrix is used in the proposed algorithm. This matrix is constructed based on the services which participate in a composition to express all output and input matching relations among services. With this matrix, both simple CR and complicated CR can be processed.

1 Introduction

Automatic Web Services composition becomes one of the most challenges in the research area of web services because composing existed web services to form a new one can add value to those web services. There are two fundamentally different ways to handle automatic services composition[1]: one way is to start with the pre-defined generic composition and perform 1-1 search to replace every generic element of a composition with a real service; the other way is to describe a set of goals and try to achieve them by building the whole process from scratch. In the second way, how to build the whole process depends on the relation between services. There are several types of relation between services, such as interoperation relation [2], effect and precondition relation [3] and output and input matching relation [3, 4, 5]. The last two relations can be judged through properties of service because IOPE (Input,

Output, Precondition and Effect) are properties of semantic web services while the first relation can not. Furthermore, I/O parameters are the basic properties of services while P/E parameters are not; therefore research of automatic services composition focuses on output and input matching relation between services.

A composition plan can be expressed as a direction graph with service(s) as node and relation as edge. Each node (except start node and end node) has only one prior node and only one successive node in a linear plan while each node has at least one prior node and at least one successive node in a netty plan. The existed methods for automatic web services composition which are based on output and input matching try to find out all composition plans for a Composition Request (CR), but just acquires linear plans. These methods will be malfunction when they handle CR which can only be satisfied by netty plan. Linear plan can only express simple services composition while netty plan can express both simple composition and complicated composition, so the existed composition methods which are limited to deal with simple composition can not handle complicated composition. In order to overcome the shortcoming of the existed methods, a new method for automatic services composition based on output and input matching is proposed in this paper. The new method can deal with CR satisfied not only by linear plans but also by netty plans through using a web services connection matrix constructed on the services which participate in the composition. In addition, this method is for the semantic web service which is based on ontology [6] because web service is developing toward semantic web service.

2 Assumptions and Expressions

2.1 Assumptions

- Using ontology to express input/output parameters of web services.
- A web service provides a single functionality.

2.2 Related Expressions

- An entity in ontology can be expressed as a set of attributes, that is $\varepsilon = \{a_1, a_2, \ldots, a_n\}$, where ε is an entity and a_i (i =1, 2,..., n) is an attribute.
- I/O parameters of semantic web services based on ontology can be expressed as entities of ontology. A kind of parameters of service is a set of entities, that is P= $\{p_1, p_2, \ldots, p_n\}$, where P can be I/O and p_i is a set because it is an entity. If a service is w, its set of input parameters is $w.I = \{ i_1, i_2, \ldots, i_m\}$, and its set of output parameters is $w.O = \{o_1, o_2, \ldots, o_n\}$.
- Symbol "||" is used in this paper to denote the cardinality of a set, e.g. |X| denotes cardinality of set X.
- Give two entities $\varepsilon_1 = \{ a_1, a_2, \ldots, a_n \}$ and $\varepsilon_2 = \{ b_1, b_2, \ldots, b_m \}$. (1) If $\forall\ \varepsilon_1.a_i$, $(\exists\ \varepsilon_2.b_j)(\varepsilon_1.a_i = \varepsilon_2.b_j)$, then $\varepsilon_1 \subseteq \varepsilon_2$; if $|\varepsilon_1| = |\varepsilon_2|$, then $\varepsilon_1 = \varepsilon_2$.

- Give two entity sets $\delta_1 = \{p1, p_2, \ldots, p_m\}$ and $\delta_2 = \{q1, q_2, \ldots, q_n\}$. (1) if $\forall\ p_i \in \delta_1$, $(\exists\ q_j \in \delta_2) \wedge (p_i \subseteq q_j)$, then $\delta_1 \tilde{\subseteq} \delta_2$, it means δ_1 is approximately subsumed by δ_2. (2) if $\exists\ p_i \in \delta_1, (\exists\ q_j \in \delta_2) \wedge (p_i \subseteq q_j)$, then $\delta_1 \cap \delta_2 \neq \varnothing$.
- Give a entity set $\delta = \{p_1, p_2, \ldots, p_m\}$ and a entity p, if $(\exists\ p_i \in \delta)(p_i \supseteq p)$, then $p \tilde{\in} \delta$, it means p approximately belongs to δ.

3 Web Services Connection Matrix (WSCM)

The relation of two services can be expressed by the connection degree between them and the set of matching parameters (I/O) between them. WSCM is used to express all the relations between random two services in a set of services.

Definition 3.1 (WSCM): suppose a set of services is S ($|S|$=N), its WSCM is a N×N matrix with the sequence numbers of its rows/columns being all services from S; its element m_{ij} is a duality tuple <X, Y>, X is the connection degree between w_i and w_j, and Y is the set of matching parameters between w_i and w_j.

Connection degree is calculated by formula (1) while the set of matching parameters is calculated by formula (2).

$$ConDegree(w_1, w_2) = \frac{|\ w_1.O \cap w_2.I\ |}{|\ w_2.I\ |} \quad (1).$$

In formula (1), $\forall\ p_i \in w_2.I$, if $\exists\ q_j \in w_1.O\ \wedge\ q_j \supseteq p_i$, then $p_i\ \tilde{\in}\ (w_1.O \cap w_2.I)$.

$$ConPSet(w_1, w_2) = \{p_i \mid p_i \in w_2.I \wedge p_i \tilde{\in} w_1.O, 1 \le i \le |\ w_2.I\ |\} \quad (2).$$

In formula (2), $\forall\ p_i \in w_2.I$, if $\exists\ q_j \in w_1.O \wedge q_j \supseteq p_i$, then $(p_i \in w_2.I) \wedge (p_i\ \tilde{\in}\ w_1.O)$.

Note: WSCM can express a direction graph with service (expressed by row or column) as node and element (X ≠ 0 and Y ≠ $\varnothing$>) as edges. 3 properties of WSCM are described as follows:

Property 1: WSCM is an asymmetry matrix with all diagonal elements being <0, $\varnothing$>.

Property 2: Suppose service w_j receives outputs from k services as its inputs, then there are k elements $m_{x_i j}$ (i=1, 2,…, k) with $m_{x_i j}$.X>0 and $m_{x_i j}$.Y ≠ $\varnothing$ in WSCM;

Property 3: Suppose service w_i provides its outputs to k services as their inputs, then there are k elements m_{ix_j} (j=1, 2,…, k) in matrix with m_{ix_j}.X>0 and m_{ix_j}.Y ≠ $\varnothing$.

4 Composition Algorithm

4.1 Concepts and Theorems about Web Services Composition

Definition 4.1 (Graph for Web Services Composition, GWSC): give a CR, suppose set of candidate services is Φ_c, the set of user's inputs in CR is U_I while the set of requested outputs is U_O; G(V, E) is a direction graph with service from Φ_c as its node and the output and input matching relation between services as its edge. If G satisfies: (1)The tail node w_t and the head node w_h of each edge satisfy: $w_t.O \cap w_h.I \neq \varnothing$; (2)Is a connected graph; (3) has only one node called w_s with only outward edges and $w_s.I \tilde{\subseteq} U_I$; and has only one node called w_e with only inward edges and $w.I \tilde{\supseteq} U_O$; every service (except w_s and w_e) comes from Φ_c; (4) the inputs of any node except w_s can be satisfied. Then G is a GWSC for CR based on Φ_c.

Note: for simpleness, we do not strictly differentiate between service and node because a node is a service in a GWSC for a CR. Proof about 5 theorems in this paper is omitted for the limited space.

Theorem 1: suppose CR has a GWSC(V, E), the set of user's inputs is U_I and the set of requested outputs is U_O, then $V-\{w_s, w_e\}=\bigcup_{i=1,2,...,N} S_i$, and (1) $\forall\ w_i \in S_1$, $w_i.I \tilde{\subseteq} U_I$; (2) $\forall\ w_i \in S_i$ (i=2,…,N), $(\exists w_j \in S_{i-1})(\ w_i.I \cap w_j.O \neq \varnothing)$; (3) c $w_i \in S_N$, $w_i.O \cap U_O \neq \varnothing$, and $(\bigcup_{w_{x_i} \in (S_1 \cup S_2 \cup ... \cup S_N)} w_{x_i}.O) \tilde{\supseteq} U_O$. S_1, S_2 ,…,S_N are called N subset of services for GWSC, and S_i is called the ith subset of services for GWSC.

Definition 4.2 (useless service): suppose a GWSC(V, E) for a CR, and $\exists w_i \in V$, if $w_i.O \cap U_O = \varnothing$ and $(\nexists w_j \in V)(\ w_i.O \cap w_j.I \neq \varnothing)$, then w_i is a useless service.

Definition 4.3 (prior service and successive service): suppose two services w_1 and w_2, if $\exists\ p_i \in w_2.I \wedge p_i \tilde{\in} w_1.O$, then w_1 is the prior service of w_2 while w_2 is the successive service of w_1.

Note: deleting a service in a GWSC for a CR may cause its prior service to become a useless service and the inputs of its successive service to be unsatisfied. If the inputs of a service can not be satisfied, it can not be executed normally, so the inputs of its successive service can not be satisfied. Therefore deleting services may cause a chain reaction of the inputs of its direction and indirection successive services being unsatisfied.

Definition 4.4 (The Biggest Graph of Web Services Composition, BGWSC): suppose CR has a GWSC based on set of candidate services Φ_c, N subsets of GWSC are S_i (i=1,2,…,N), if $\nexists\ w_i \in (\Phi_c - V)$ and $w_i.I \cap w_j.O \neq \varnothing$ $(w_j \in (S_1 \vee S_2 \vee \cdots \vee S_{N-1}))$, then the GWSC is the BGWSC for CR based on Φ_c.

Definition 4.5 (the Smallest Graph of Web Services Composition, SGWSC): suppose CR has a GWSC (V, E) based on set of candidate services Φ_c, if w_e will be deleted because of the chain reaction of inputs of services being unsatisfied caused by deleting any service (except w_s and w_e) in V, then the GWSC is a SGWSC for CR based on Φ_c.

Note: the deletion of w_e means that the quested outputs can not be acquired.

Definition 4.6 (a call to a service): Suppose service w_i and its prior services is in set $\theta = \{ w_j | w_j.O \cap w_i.I \neq \varnothing \}$, if θ has a subset $\theta_c = \{ w_j | (\bigcup_{w_j \in \theta} w_j.O \tilde{\supseteq} w_i.I) \wedge ((\forall w_\alpha \in \theta)(\bigcup_{w_j \in \theta} w_j.O - w_\alpha.O) \tilde{\not\supseteq} w_i.I) \}$, then θ_c is a call to service w_i.

Definition 4.7 (a list of calls for a set of services): suppose a set of services is w_i (i=1,2,…,k) and the calls to w_i is set $\psi_i = \{ \theta_{ij} | j=1,2,\ldots,m_i \}$, then $\xi = \{ < w_i , \theta_{ij} > | \theta_{ij} \in \psi_i ; i=1,2,\ldots,k; 1 \leqslant j \leqslant m_i \}$ is a list of calls for w_i (i=1,2,…,k).

Note: the number of different lists of calls to a set of services can be calculated by formula (3). $$\prod_{i=1,2,..,k} |\psi_i| . \quad (3)$$

Definition 4.8 (share service, valid edge and invalid edge): in a GWSC for a CR, if a service has at least two calls to itself, it is a share service. Each edge between the share service and one of its prior services which belongs to at least one call is a valid edge. If an edge is not a valid one, it is an invalid edge.

Theorem 2: suppose CR has a GWSC based on set of candidate services Φ_c, and then CR will have only one BGWSC and at least one SGWSC based on Φ_c.

Theorem 3: suppose a CR has k SGWSCs based on set of candidate services Φ_c: G_1, G_2,…, G_k, and the BGWSC for the CR based on Φ_c is G_{MAX}, then: $\forall$ G_i (i=1, 2,…, k), G_i is a sub-graph of G_{MAX}.

Theorem 4: suppose a CR has a GWSC based on a set of candidate services and this GWSC has at least a share service, then the CR has at least two SGWSCs.

Theorem 5: suppose a CR has a GWSC, and service w in this GWSC has k prior services w_i (i=1, 2,…, k) and $\sum_{i=1,2,..,k} ConDegree(w_i, w) < 1$, then these k prior services can not form a call to w.

4.2Mechanism of the Composition Algorithm

4.2.1 The method of searching for all SGWSCs from the BGWSC for a CR

The BGWSC for a CR can be acquired according to theorem 1 and this BGWSC is the only one according to theorem 2. All SGWSCs can be acquired from a BGWSC according to theorem 3. Whether a BGWSC contains more than one SGWSC or not can be judged based on theorem 4. The number of SGWSC of a CR can be

calculated by formula (3) because a list of calls for the set of share services in the BGWSC decides a SGWSC. All different SGWSCs can be found through searching for all different lists of calls for the set of share services in a BGWSC.

Useless services can be deleted from a BGWSC to decrease the number of services before extracting SGWSCs. When extract a SGWSC, for each share service, take one of its calls from a list of calls, and delete the edges which tail nodes (services) do not belong to that call, then a share service becomes a non share service. Deleting edges may result in useless services. Therefore, useless services checking and deletion should be done after deleting an edge. Some non share services may have invalid edges in a BGWSC, these invalid edges should be deleted after transforming all share services to non share ones. When there are not share services, useless services and invalid edges, there is no a service which can be deleted from this GWSC, and this GWSC is a SGWSC.

Extracting a SGWSC from a BGWSC is done in the WSCM of the BGWSC for a CR by deleting useless services and invalid edges and transforming share services to non share services. The methods of doing so are given as follows:

(1) The method of deleting useless services

In the WSCM of a GWSC, if the connection degree value of every element in a row is 0, the service corresponding to this row is a useless service. Scan the WSCM to find out all such rows and delete them and their counterpart columns. Deleting useless service may result in its prior services becoming invalid ones, so delete useless services repeatedly until there is no more useless service in the WSCM.

(2) The method of deleting invalid edges

Deleting invalid edges is to find out valid edges for a node in fact. Searching for valid edges follows criterion 1: every service in a GWSC should receive outputs from its prior services as few as possible. Following this criterion, the number of nodes in a GWSC can be decreased and the execution efficient of the GWSC can be improved. The method of searching for valid edges following criterion 1 is described as follows:

In the WSCM of a GWSC, for w_j:

A. if $m_{ij}.X=1$, then edge $<w_i, w_j>$ is a valid edge.

B. if $(\bigcup_{1\le i\le N} m_{ij}.Y \tilde{\supseteq} w_j.I) \wedge ((\forall m_{\alpha j})(\bigcup_{1\le i\le N} m_{ij}.Y - m_{\alpha j}.Y) \not\tilde{\supseteq} w_j.I)$, then edge $< w_i, w_j >$ $(1 \leqslant i \leqslant N)$ is a valid edge.

(3) The method of transforming a share service to a non share service

Transforming a share service to a non share service is to delete those prior services which do not belong to specified call of the share service. After transforming, useless services checking and deletion should be done because deleting service may result in useless services.

4.2.2 The method of searching for all calls to a share service

A prior service whose connection degree value with the share service is equal 1 can form a call to the share service by itself. According to theorem 5, only the prior service whose connection degree value with the share service is less than 1 needs to join with other services to form a call to the share service. Different combinations of prior services form different calls. Use multitree with weight to find out all different

combinations of prior services of a share services. The weight of the multitree is the connection degree.

Put all prior services whose connection degree value with the share service are less than 1 in set θ_w. Construct a multitree with weight for every service in θ_w. Suppose the share service is w_j and a prior service of w_j is w_i, then the multitree constructed for w_i is called w_j - w_i tree. The way to construct such a tree by BFS (Breadth First Search) is described as follows:

(1) The root node of the tree is a null service and w_i is the only child node of the root node.

(2) Scan all leaf nodes from left to right. If a leaf node and all its ancestor node (except the root node) can form a call to w_j, then this node can not have a child any more; otherwise, select from θ_w all services which are neither the ancestor nodes of the leaf node nor the brother nodes on the left side of the leaf node to be child nodes of this leaf node. If neither one service can be selected from θ_w, this leaf node can not have a child any more. Repeat 2 until all leaf nodes can not have any child node.

Get all calls containing w_i from w_j - w_i tree and put them in set C_i, combine C_i (i=1, 2,…,| θ_w |) and delete repeated calls and get a new set which is the set of calls to w_j). If two calls are the same or one call subsumes another call, they are repeated calls.

4.2.3 The method of finding out all lists of calls for a set of share services

Suppose there are m share services w_1, w_2,…, w_m in a GWSC, and the set of calls to w_i is ψ_i (i=1, 2,…, m). Use multitree which is created by BFS to search for all different lists of calls for these m share services. The multitree has m+1 levels and its null root node is in level 0. Each node in level i-1 has <w_i, θ_{ij} > ($\theta_{ij} \in \psi_i$; j=1, 2,…,| ψ_i |) as its all child nodes, so nodes in the same level have the same child nodes. Each leaf node and all its ancestor nodes (except the root node) form a list of calls for these share services. Two random lists of calls are not the same because nodes in each level are different.

4.3 Steps of the Algorithm

(1) Sub-algorithm of deleting useless services: DeleteUServ (WSCM, Φ), Φ is the set of services of WSCM.

Step 1: Scan each row in WSCM. If $\forall w_i \in \Phi$ (i=1, 2,…, | Φ |), $\nexists m_{ij}$.X=0 (j=1,2,…,| Φ |), return; otherwise delete row w_i and column w_i from WSCM, $\Phi = \Phi - \{ w_i \}$, go to step 1.

(2)Steps of composition algorithm

Give set of candidate services Φ_c, the set of user's inputs U_I and the set of user' requested outputs U_O. CallsToAserv () and ListsOfCalls () are sub-algorithms. The first one is to get all calls to a share service and the second is to get all lists of calls for all share services in a GWSC.

Setp1: if $U_I \supseteq U_O$, go to step 2; otherwise if $\Phi_C \neq \varnothing$, get a service w from Φ_C, if $w.I \tilde{\subseteq} U_I$, let $\Phi_S = \Phi_S \cup \{w\}$, $\Phi_C = \Phi_C - \{w\}$, $U_I = U_I \cup w.O$. If $(\Phi_C = \varnothing) \wedge (U_I \not\tilde{\supseteq} U_O)$, return the message of composition failure and terminate the algorithm; otherwise go to step 1.

Step 2: calculate the web services connection matrix for Φ_S and let the matrix to be $WSCM_{\Phi_S}$.

Step 3: DeleteUServ ($WSCM_{\Phi_S}$, Φ_S), let $\Phi_{tmp} = \Phi_S$, $\varpi = \varnothing$, N=0, $\chi_{SGWSC} = \varnothing$, $\Omega = \varnothing$.

Step 4: if $\Phi_{tmp} = \varnothing$, go to step 7; otherwise get a service w_j from Φ_{tmp}, and let $\Phi_{tmp} = \Phi_{tmp} - \{w_j\}$. Scan column w_j in $WSCM_{\Phi_S}$, and let $\sigma_j = \{w_i | m_{ij}.X=1\}$, $\sigma_j' = \{w_i | 0< m_{ij}.X<1\}$. If $|\sigma_j'| \geqslant 3$ and $\bigcup_{w_i \in \sigma_j'} w_i.O \tilde{\supseteq} w_j.I$, let $\sigma_{tmp} = \sigma_j'$, if $|\sigma_j| > 0$, $\psi_j = \bigcup_{w_i \in \sigma_j} \{\{w_i\}\}$, go to step 5; otherwise if $(|\sigma_j| > 1) \wedge (\bigcup_{w_i \in \sigma_j'} w_i.O \not\tilde{\supseteq} w_j.I)$, $\psi_j = \bigcup_{w_i \in \sigma_j} \{\{w_i\}\}$; else if $(|\sigma_j| > 0) \wedge (\bigcup_{w_i \in \sigma_j'} w_i.O \tilde{\supseteq} w_j.I)$, $\psi_j = \bigcup_{w_i \in \sigma_j} \{\{w_i\}\} \cup \{\sigma_j'\}$. Go to step 4.

Step 5: if $\sigma_{tmp} = \varnothing$, if $|\psi_j| > 1$, let $\varpi = \varpi \cup \{<N+1, w_j, \psi_j>\}$, go to step 4; otherwise get a service w_R from σ_{tmp}, and let $\sigma_{tmp} = \sigma_{tmp} - \{w_R\}$, $\psi' = \varnothing$, $\omega_{CH} = \sigma_j' - \{w_R\}$, $V=\varnothing$, $E=\varnothing$, ψ' =CallsToAserv(w_R, ω_{CH}, ψ', V, E).

Step 6: if $\psi' = \varnothing$, go to step 5; otherwise get an element α from ψ', let $\psi' = \psi' - \{\alpha\}$. If $(\exists \beta)(\beta \in \psi_j) \wedge (\beta \supset \alpha)$, let $\psi_j = (\psi_j - \{\beta\}) \cup \{\alpha\}$; Otherwise if $(\nexists \beta)(\beta \in \psi_j) \wedge (\beta = \alpha)$, let $\psi_j = \psi_j \cup \{\alpha\}$. Go to step 6.

Step 7: if $\varpi = \varnothing$, let $WSCM_{\Phi}' = WSCM_{\Phi_S}$, $\Phi' = \Phi_S$, go to step 10; otherwise let v_R = t (t is the root node of the tree and it is also an empty node), get a element $<x, w_x, \psi_x>$ from ϖ, let l=0, H=$|\varpi|$, $V=\varnothing$, $E=\varnothing$, Ω =ListsOfCalls (v_R, w_x, ψ_x, ϖ, l, H, V, E).

Step 8: if $\Omega = \varnothing$, go to step14; otherwise get an element ξ from Ω, let $\Omega = \Omega - \{\xi\}$.

Step 9: if $\xi = \varnothing$, let $\varphi = \Phi^{'}$, go to step 10; otherwise get an element $<w_j, \theta_j>$ from ξ, let $\xi = \xi - \{<w_j, \theta_j>\}$, $\omega = \{ w_i \mid (w_i \notin \theta_j) \wedge (w_i.0 \cap w_j.I \neq \varnothing), 1 \leqslant i \leqslant |\Phi^{'}|\}$, if $|\omega| > 0$, let $m_{ij} = <0, \varnothing> (w_i \in \omega, 1 \leqslant i \leqslant |\Phi^{'}|)$ in $WSCM^{'}_{\Phi}$. DeleteUServ ($WSCM^{'}_{\Phi}, \Phi^{'}$), and go to step 9.

Step 10: if $\varphi = \varnothing$, go to step 13; otherwise get a service w_j from φ, let $\varphi = \varphi - \{ w_j \}$, if $\exists\, m_{\alpha j}.X=1$, let $m_{ij} = <0, \varnothing> ((0<i<|\Phi^{'}|) \wedge i \neq \alpha)$; Otherwise let $\eta = \{w_i | 0 < m_{ij}.X < 1\}$. If $|\eta| > 0$, let $\theta = w_j.I$, go to step 11; otherwise go to step 13.

Step 11: if $\theta = \varnothing$, go to step 12; otherwise get a service w_α from η, let $\eta = \eta - \{ w_\alpha \}$, let w_α satisfy $|\theta - m_{\alpha j}.Y| = MAX\{|w_j.I - m_{ij}.Y|\}$, let $\phi = \phi \cup \{ w_\alpha \}$, $i=1,2,...,|\Phi^{'}|$

$\theta = w_j.I - m_{\alpha j}.Y$, go to step 11.

Step 12: $\omega = \{w_i | (w_i \notin \phi) \wedge (w_i.0 \cap w_j.I \neq \varnothing)\}$, let $m_{ij} = <0, \varnothing> (w_i \in \omega) \wedge m_{ij}.X \neq 0$, $1 \leq i \leq |\Phi^{'}|$). DeleteUServ ($WSCM^{'}_{\Phi}, \Phi^{'}$), and Go to step 10.

Step 13: $\chi_{SGWSC} = \chi_{SGWSC} \cup \{\Phi^{'}\}$, if $\Omega = \varnothing$, go to step 14; Otherwise let $WSCM^{'}_{\Phi} = WSCM_{\Phi}$, $\Phi^{'} = \Phi_S$, go to step 8.

Step 14: Get all SGWSCs from χ_{SGWSC}, and calculate the QoS (Quality of Service) value for each SGWSC according to certain selection policy. Return the SGWSC with the optimized QoS value and terminate the algorithm.

In the algorithm describe above, step 1 acquires BGWSC; step 2 calculates WSCM for BGWSC; step 3 deletes useless service in BGWSC; step 4, 5 and 6 find out all share services from BGWSC and the sets of calls to each share service; step 7 acquires the set of lists of calls for all the share services; step 8 and 9 transform every share service to non share service; step 10, 11 and 12 delete invalid edges; step 13 puts a SGWSC into the set of all SGWSCs; step 14 acquires the SGWSC with the optimized QoS value. If there is not any proper composition which can meet the CR, the algorithm will return the failure information.

5 Related Works

Method in [4] uses heuristics to select the most proper service from the discovered services in each services discovery phase and acquires only one linear composition plan which contains the least services. Therefore, this method is used in the case that all the composition plans are linear for a CR and the CR requires the optimized composition plan which contains the least services.

Method in [7] uses entity matching to select the most similar service in the candidate services and acquires a composition as a tree. This tree is created from leaf

nodes to the root node and a composition plan is a path from the root node to a leaf node of the tree. Some composition plans may be omitted during the discovery of services in non leaf nodes. This may cause the optimized plan for a CR to be lost.

Methods in [3, 5] acquire a composition as an expression in which services are linked by sequence or nondeterminism operator while the parallel operator is used to link two inputs. An expression without any nondeterminism operator is a composition plan, so an expression with any nondeterminism operator will be divided into several expressions. Therefore, the composition plan is also linear.

In conclusion, the methods which use output and input matching to find out composition plan for a CR can only get linear composition plan, so they are limited to particular cases that CRs can only be satisfied by linear composition plans. Furthermore, method in [4] is limited to plan which contains the least services, method in [7] is likely to lose some plans. The method in this paper is more general than the existed methods for it can get not only linear but also netty plan so it can be used in general case, moreover this method will not lost any plan.

6 Conclusion and Future Work

It is a very important method for non predefined composition to utilize the output and input matching between services. The existed methods based on such matching can only deal with those CR which can be satisfied by linear composition plan, and it will be malfunction when the CR can only be satisfied by netty composition plan. In order to overcome the limitation of the existed methods, a new method is proposed in this paper which is also based on the output and input matching. The existed methods only focus on services discovery but ignore relation discovery, so they can only acquire linear plans with simple structure. On the contrary, the proposed method in this paper focuses on both services discovery and relations discovery because it searches for all relations among services after services discovery. The proposed method can acquire not only linear composition plans but also the netty plans just because it finds out all relations among the discovered services. In the proposed method, the set of selected services is found first; then a connection matrix for the set is constructed to find out all relations among the selected services; next certain operations are done to the matrix to find out all composition plans for a CR. The optimized plan is acquired by calculating QoS value for each plan and selecting the optimized one.

In future work, we will research on how to improve the efficiency of our method when the services which participate in a composition are on the increase.

References

1.N. Milanovic and M. Malek, "Architectural Support for Automatic Service Composition" , *Proceedings of the 2005 IEEE International Conference on Services Computing (SCC'05),* 2,133-140(2005).

2.B. Medjahed and A.Bouguettaya, "A Multilevel Composability Model for Semantic Web Services", *IEEE Trans. Knowledge and Data Eng,* 17(7),954-968(2005).

3. L. Freddy, L. Alain and N.S. Ecole, "Semantic Web Service Composition through a Matchmaking of Domain", *European Conference on Web Services (ECOWS'06)* ,33-242(2006).

4. S.C. oh, B.W. On , E.J Larson and D. Lee, "Web Services Discovery and Composition as graph search problem", *Proceedings of 2005 IEEE International Conference on e-Technology, e-commence and e-servic* ,784-786(2005).

5. L. Freddy, L. Alain and N.S. Ecole , "Semantic Web Service Composition Based on a Closed World Assumption", *European Conference on Web Services, (ECOWS'06)* 171-180(2006).

6. T.R. Gruber, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquistion,* 5(2), 199-220(1993).

7. A. Lerina, C. Gerardo and C. Anna, "An algorithm for Web service discovery through their composition", *Proceedings of the IEEE International Conference on Web Services (ICWS'04) ,* 332-339(2004).