An Integrated Algorithm for QoS-Aware Logistics Service Composition

Bao JianMin
Department of Internet of Things
Nanjing University of Posts and Telecommunications
Nanjing, P.R.China
bao@njupt.edu.cn

Liu Jie
Department of Internet of Things
Nanjing University of Posts and Telecommunications
Nanjing, P.R.China
1214073505@njupt.edu.cn

***Abstract*—With the rapid development of the logistics industry, there are more and more logistics services on the Internet platforms. However, a single atomic service has not been able to meet the complex needs of users. So the service composition in large-scale environment is very meaningful. In this paper, we propose a web service composition approach for logistics, which consists of three main steps: 1) Using semantic model to automatically generate the service matching graph to meet the request; 2) An algorithm for computing the optimal QoS based on the shortest path of Dijkstra; 3) Based on the previous semantic matching mechanism and the optimal aggregate QoS value as the constraint condition, the redundant service in the composited graph is removed; This method can be used for automatically discover and composite a minimum number of services from a large number of services that satisfies both multiple QoS criteria and semantic matching between logistics services.**

Keywords- Logistics Service; Semantic web services; QoS; Service composition.

I. INTRODUCTION

Logistics service platform can integrate various forms and large quantity of logistics services, greatly improve the efficiency of supply chain and save costs, enhance the competitiveness of the supply chain [1]. Given a complex user requirement, a single atomic service in the logistics service platform cannot meet the needs of the user, so it is necessary to compose services to obtain complex functions. In the process of service composition, how to construct the service composition which satisfies the QoS restriction and obtain the optimal QoS has been the research hotspot. In the context of big data, the number of logistics services is increasing dramatically, and the manual generation of service composition are very lagging behind. Therefore, it is great significance to automatically generate the composited services which can satisfy the users' needs and have the optimal QoS in the mass services.

In the process of service composition, the choice of services usually need to consider both functional and non-functional requirements [2]. The semantic matching between input and output of services is the core standard of functionality. In the service-oriented domain (SOA), a large number of studies focus on the use of semantic technology to achieve automatic web service composition [3]. The automatic web service composition method makes extensive use of AI technology, especially the AI planning [4-5], because the automatic composition problem can be equivalently converted to AI planning problem. These existing methods typically use WSDL or BPEL-WS to describe the input and output of the service.

The other aspect of service composition that needs to be considered is nonfunctional. A lot of research in non-functional generally refers to the quality of service (QoS) such as delay, throughput, price, reliability, etc. When the user gives a requirement, the logistics service provide a large number of available candidate services. These services have many service functions similar or identical, then the optimal QoS will have great help to generate the complex task of service composition. Therefore, it is a hot topic to research the QoS-aware web service composition from different perspectives.

The structure of this paper is follows. Section 2 disscuss some related works. Section 3 defines logistics service and QoS calculation model. Section 4 introduces our algorithm. Section 5 simulates and evaluates our algorithm.

II. RELATED WORK

In recent years, QoS-based web service composition optimization has become an important topic in service-oriented software engineering. The traditional QoS-aware service composition technique can only obtain local optimization results. Because they need pre-defined business processes. [6] consider the logistics service QoS weights to provide a QoS aware service composition algorithm. But their services are small in size, few in service, and the QoS parameter is not complete, and it is not suitable for the complex logistics service composition under the mass of service environment. [1] provides an algorithm for constructing a global optimal QoS service path in a large scale environment. But their logistics service contains only sequence pattern not applicable to normal service composition. [5] use the existing AI planner to solve QoS optimal service auto-composition problem, but the calculation of composited QoS only considers the simple summation method. [3] provides a way to seeking quality of service composition in the semantic dimension, which is not flexible enough.

In this paper, we use the semantic matching mechanism to automatically generate the logistics service composition, and use the search algorithm to obtain the optimal solution with minimum number of services and guarantee the optimal QoS.

III. PROBLEM MODELING

This section presents models for logistics services, semantics, and QoS.

A. *Logistics Service QoS Model*

Definition 1. A logistics service request R is defined as a two tuple $R=\{In_r, Out_r\}$, In_r is an input set, Out_r is an output set.

Definition 2. A semantic logistics service with Qos can be defined as a four tuple $Ls=\{Ln_{ls}, In_{ls}, Out_{ls}, QoS_{ls}\}\in L$; Ln_{ls} is the name of the logistics service. In_{ls} is a set of input sets needed to invoke Ls. Out_{ls} is the output set executed by Ls. $QoS_{ls}=\{q_{ls}^1,...,q_{ls}^n\}$ is the set of QoS values associated with the service. L is a collection of all available services in the service registry.

Definition 3. The QoS attribute set is a five-tuple $\{T, P, R_1, R_2, A\}$,where T (time) is the duration of the logistics service, P (price) is the price of the logistics service, R_1 (reliability) is the possibility of logistics service according to the commitment to the implementation, R_2 (reputation) is the reputation of the logistics service. A (accuracy) refers to the accuracy of the conversion after the completion of the order. Such a logistics service QoS can be expressed as a vector:

$$q_{ls}^i=(q_T(Ls_i), q_C(Ls_i), q_{R_1}(Ls_i), q_{R_2}(Ls_i), q_A(Ls_i)) \qquad (1)$$

B. *Semantic Service Model*

We use semantic input and output to match the inputs and outputs of services together to form the functionality of multiple services. In order to measure the quality of matching, we need a service semantic I / O information matching mechanism. Reference [7] provides such a mechanism.

The ontology concept is used here as the semantics of service input and output. The ontology concept set S can be defined as: $Ont(In_{ls}, Out_{ls}\subseteq O)$.

Definition4. Given $C_1, C_2\subseteq O$, We define "$\otimes: O\times O\rightarrow O$" such that $C_1\otimes C_2=\{c_2\in C_2 \mid match(c_1,c_2), c_1\in C_1\}$, The operators here is not commutative.

The above operators to define the full and partial match between the concepts.

A Logistics Service $Ls=\{In_{ls}, Out_{ls}\}$ is relevant to a request $R=\{In_r, Out_r\}$. where $In_r\subseteq O$ refers to the requirements of the input, $Out_r\subseteq O$ means the desired output. Then $In_r\otimes In_{ls}=In_{ls}$ and $Out_r\otimes Out_{ls}=Out_{ls}$ respectively represents the demand input and the service input complete match, the expected output and the service output complete match.

C. *Service Composition and QoS computing model*

Definition 5. Given A logistics service Ls, The serial execution of the Ls is $Ls_0\rightarrow Ls_1$,where $Ls_0\supseteq Ls.In, Ls_1\supseteq Ls.Out, Ls_0, Ls_1$ is a set of services.

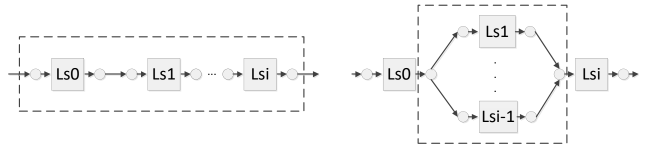

Figure 1. Serial execution and Parallel execution structure

Definition 6. Given A logistics service Ls, The Parallel execution of the Ls is $P_0\Rightarrow P_1\Rightarrow P_0...\Rightarrow p_n$.

a) $P_k(o\le k\le n)$ is a set of parameters and represents the level of parallel execution. which satisfies the condition $(P_0=Ls.In)\wedge(P_n\supseteq Ls.Out)$;

b) $P_k\Rightarrow P_{k+1}(0\le k\le n-1)$ is a step in the parallel execution of Ls. Represents a number of parallel calls with the structure of the sub-services, if the collection of these sub-services for the PL_k need to meet the following conditions:

$$\forall Ls\in PL_k((P_k\supseteq L^{'}.In)\wedge(P_{k+1}=P_k\cup(\bigcup_{Ls^{'}\in PL_k} Ls^{'}.Out))) \quad (2)$$

c) $\forall i,j(0\le i\le n-1\wedge 0\le j\le n-1\wedge i\ne j)$ satisfy: $PL_i\cap PL_j=\Phi$; R_1

TABLE I. AGGREGATION METHOD OF QOS METRICS

index	Serial execution	Parallel execution
T	$\sum_{i=0}^{k} q_T(Ls_i)$	$\max_k(q_T(Ls_1), q_T(Ls_2),..., q_T(Ls_k))$
P	$\sum_{i=0}^{k} q_p(Ls_i)$	$\sum_{i=0}^{k} q_p(Ls_i)$
R_1	$\prod_{i=0}^{k} q_{R_1}(Ls_i)$	$\min_k(q_{R_1}(Ls_1), q_{R_1}(Ls_2),..., q_{R_1}(Ls_k))$
R_2	$\prod_{i=0}^{k} q_{R_2}(Ls_i)$	$\min_k(q_{R_2}(Ls_1), q_{R_2}(Ls_2),..., q_{R_2}(Ls_k))$
A	$\prod_{i=0}^{k} q_A(Ls_i)$	$\min_k(q_A(Ls_1), q_A(Ls_2),..., q_A(Ls_k))$

For the nonlinear properties of A, R_1, R_2 will cause the contradiction between the global QoS and local QoS. [8] provides a linear normalization method which can solve this contradiction. After normalization, the QoS of a single service is $(\overline{q_T}(Ls), \overline{q_C}(Ls), \overline{q_{R_1}}(Ls), \overline{q_{R_2}}(Ls), \overline{q_A}(Ls))$. When a user is given a weight $(w_T, w_C, w_{R_1}, w_{R_2}, w_A)$ the comprehensive QoS cost is:

$$CQoS(Ls) = w_T \bullet q_T(Ls) + w_C \bullet q_C(Ls) + w_{R_1} \bullet q_{R_1}(Ls) + w_{R_2} \bullet q_{R_2}(Ls) + w_A \bullet q_A(Ls) \quad (3)$$

The aggregated QoS cost of a logistics service path can be expressed as:

$$QCost(path) = \sum\nolimits_{Ls_i \in serviceList} CQoS(Ls_i) \quad (4)$$

IV. COMPOSITION ALGORITHM

Based on the QoS-aware composition model defined above, this section details how to use the multi step strategy to automatically compose a large number of services with optimal QoS and minimum of service composition. The main process of this method is: First, given a user's needs, generate a match graph associated with this requirement. When the graph is established, a forward search algorithm based on the optimal label is used to find the global optimal QoS. Finally, the match graph is pruned based on the former information, and the redundant service and the equivalent service are removed to get a service composition with the minimum number of services.

A. Generation of the Service Match Graph

A logistics service composition request includes a user specified input and desired output, and an available service set. First, determine all available related services, and use the semantic model to calculate all the possible matches between input and output. Finally, a matching service map containing many possible valid composition is output. Figure 1 is an example of a service matching graph.

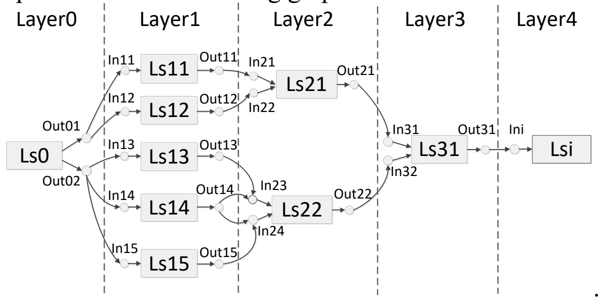

Figure 2. Example of service matching graph

In short, the matching graph is generated by calculating all the invokable services layer by layer. The output of the source service of the first layer is the required input In_r .The input to the last layer of service is the desired output Out_r . In each layer, a logistics service that matches the input to the output of a layer is used as a candidate service (L.6). Then for each candidate service, the algorithm detects whether all unmatched inputs match with the output of the service. Removes all matching services from the unmatched set of current services. Until there is no matching service available. Adds all matching services to the $L_{selected}$ set and adds their output to the available concept availCon. The service in $L_{selected}$ is removed from the available service set and update the set of available services. Then the service in $L_{selected}$ is added to the ServiceList as a service at this layer in the matching graph. The operation of each layer is repeated until there is no service that can be matched, and the algorithm terminates. The pseudo code of the algorithm is as follows:

Algorithm 1. Generate matching graph

```
1: Function Ls GRAPH( $R = \{In_r, Out_r\}, L$ )
2:     $C = In_r; L' = L; ServiceList = \{Ls_0, Ls_i\}$
3:     $unmatched\_In = []$; $avaliCon = In_r; i = 0$;
4:     repeat  $C = In_r; L' = L; ServiceList = \{Ls_0, Ls_i\}$
5:         $L_{selected} = \Phi; i = i + 1$;
6:         $L_{rel} = \{L \in L' \mid availCon \otimes In_{Ls} \neq \Phi$
7:         $L_{rel} = L_{rel} \setminus ServiceList$
8:         for all  $Ls = \{In_{Ls_i}, Out_{Ls_i}\} \in L_{rel}$  do
9:             $U_{set} = unmatched\_In[Ls_i]$
10:            $M_{set} = C \otimes U_{set}$
11:            $unmatched\_In[Ls_i] = U_{set} \otimes M_{set}$
12:            if  $M_{set} = \Phi$  then
13:                $L_{selected} = L_{selected} \cup Ls_i$
14:                $availCon = availCon \cup Out_{Ls_i}$
15:        $L' = L' \cup Ls_{selected}$
16:        $SelectList = SelectList \cup Ls_{selected}$
17:        $C = C \cup availCon$
18:        $availCon = \Phi$
19:    until  $L_{selected} = \Phi$
20:    return  GRAPH(ServiceList)
```

B. Optimal QoS-Aware Composition

After generating the semantic matching graph. The next step is to compute the optimal end-to-end QoS. There are a lot of composited services in semantic matching graph. It is extremely inefficient to compute QoS for all composited services. In fact, the optimal QoS of the composited service is equal to the optimal composite QoS of the end node in the composite service. The optimal end-to-end QoS can be obtained by computing the aggregate QoS value of each input and output in the matching graph by the shortest path algorithm.

As shown in Figure 1, there are two types of nodes: The boxes represent logistics service and the dot represents the ontology concept. A concept node can have multiple inputs, each of which represents a path that can take the concept. The optimal cost of a concept is determined by the optimal value of all its input paths. While the service node needs all the concept nodes to match before they can arrive. A Serial structure when a service has only one input, and a parallel structure when there are multiple inputs. The cost of a service is calculated using the rules of definitions 5 and 6. For the optimal input of each concept, we use the Dijkstra-based label-setting algorithm [9] to perform the forward search from Ls_0 to Ls_1 to compute the optimal QoS.

Algorithm 2. Optimal QoS-Aware Service Composition

```
1:  Function Opt_QoS (GRAPH, QoS)
2:      queue $\rightarrow$ (0, Source) ;# 0=best cost,1=worst cost
3:  while $queue \neq \Phi$ do
4:  update=false; $Ls_A \leftarrow queue$
5:      $newAggQos = aggQoS(Ls_A)$
6:      for all $Ls_B$ match $Ls_A$ do
7:          $InMatched = \{i_m : i_m \in (Out_{Ls_A} \cap In_{Ls_B}) \wedge i_m \in In_{Ls_B}$
8:          for all $i_m \in InMatched$ do
9:              if $newAggQos < i_m.aggQos$ then
10:                 $i_m.aggQos = newAggQos$
11:                 $i_m.op = Ls_A$
12:             end if
13:         end for
14:         $newCost = aggQoS(Ls_B)$
15:         $queue \leftarrow (newCost, Ls_B)$
16:     end for
17: end while
18: return queue
```

C. Service Minimzation

In general, the service matching graph produced in 3.1 will contain many redundant services. The more services in the graph will result in a larger scale, leading to a larger search space and increasing the cost of the final composition. In order to reduce the number of services in the composition, we use the following techniques to optimize the Service composition graph:

This process is divided into two steps: a) delete the useless service; b) delete the equivalent service;

a) The first step is to traverse the matching graph from Ls_i to Ls_0. From the last layer, layer by layer forward search output related services. Finally, the service that is not traversed is useless to the final destination, and then deletes it. If the service $Ls_i = \{In_i, Out_i\}$ in layer i and the service $Ls_{i-1} = \{In_{i-1}, Out_{i-1}\}$ in layer i-1 satisfy $Out_{i-1} \otimes In_i \neq \Phi$, then Ls_i is the output-related service of Ls_{i-1}. In each layer, the output of the service matches the input of the previous layer, and the service is considered to be useful. The algorithm terminates until the number of levels is zero

Algorithm 3.Remove useless service in matching graph

```
1:  Function Min_GRAPH( $Ls \in ServiceList$ )
2:      $\min\_M = \{Ls_N\}; Useless = \{\}; i = N-1;$
3:      while i>1
4:          $R = \{\}$
5:          for all Ls in layer i
6:              if $Out \otimes In \neq \Phi \mid Out \subseteq Ls, In \subseteq M$
7:                  $R = \{Ls\}$
8:              else $Useless = \{Ls\}$
9:              end if
10:         end for
11:         $\min\_M = \min\_M \cup R$
12:         i=i-1
13:     end while
14: return min_M
```

b) The second step to delete the equivalent service. In the service composition graph some services have equivalent inputs. Then leave the service with the parameters of the advantages, delete other services. For a set of equivalent logistic services, if the aggregate QoS cost of service i is smaller than that of service j, the service i is selected as the dominant service and the service j is deleted;

V. EXPERIMENT

This section first explains the generation data rules of the simulation experiment. Then, in order to verify the effectiveness and efficiency of the proposed method, we compare it with the GA algorithm [10] and the three-step strategy [6] in terms of execution time and QoS cost. At each stage of the experiment, the corresponding indexes were recorded. The following are the main introduction of the experimental process and results.

A. Data generation rules

Considering the characteristics of logistics service, we use the semantic rule model in [11] to generate the semantic of logistics service. The QoS attributes of the logistics service are randomly generated and the rules are as follows:

TABLE II. DATA GENERATION RULE

Semantic service generation rules	The logistics service with semantics is generated using the rules in [12]
QoS	P (price) and T (time) are randomly selected between [0.5-1.5], R1 (Reliability), A (Accuracy), R2 (Reputation) Randomly selected between [0.5-1.0]

B. Experimental Process and Results

The experiment is based on Java, and the test set is generated by 4.1 rules, including 8 sets of datasets (from Test1 to Test8), each with a growing number of services (from 100 to 8000). There are four file types in each test set: the input and output parameters of the service are described in WSDL file; the ontology of the parameters and their relationship are described in OWL file; the service QoS is described by WSLA file; the request file describes the user request; Each experiment selects a set of data sets as the input of the system, and obtains the service composition with the minimum number of services and the best QoS after the match and search by composited planner.

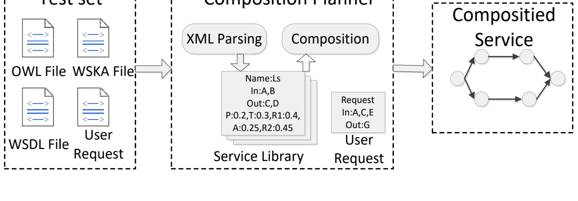

Figure 3. System experimental architecture

Table 2 shows the experimental results for each data set. The first column shows the name of the dataset. The second column shows the number of services in each data set. The third column shows the number of services in the optimized service composition graph, which represents the real search space for the user's needs. The last column shows the number of layers is the combination of solution execution path length.

TABLE III. RESULT OF EXPERIMENTAL DATA

Dataset	Service Number	Min_service	Layer
Test1	100	8	3
Test2	400	5	3
Test3	900	10	5
Test4	1600	22	8
Test5	2500	40	21
Test6	3500	20	12
Test7	6000	22	12
Test8	8000	40	20

Next, we evaluate the algorithm in this paper. In order to compare with the three-step strategy, the GA algorithm, we perform all the algorithms in the same environment. In this experiment, the optimized QoS is unified as the final goal of each algorithm. The results of the tests are shown in Fig.3 and Fig.4 for different numbers of services:

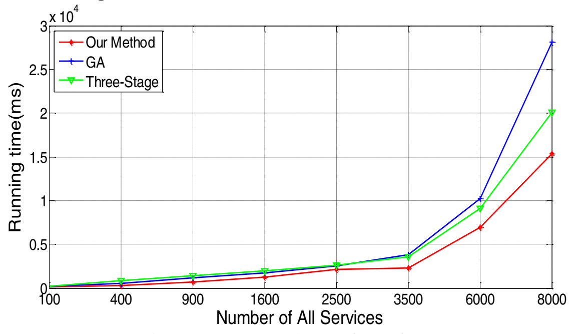

Figure 4. Comparison of run times

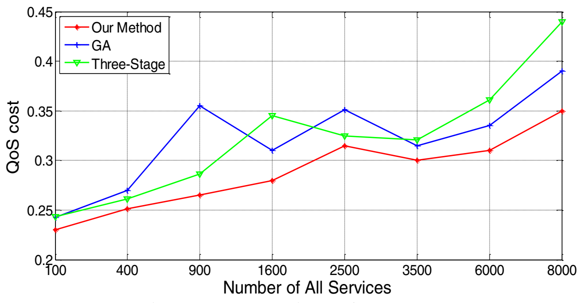

Figure 5. Comparison of QoS Costs

As shown in Fig.4 and Fig.5, when the number of services is less than 2500, the execution time of this method is slightly lower than that of GA and GA. But with the increasing number of services, the advantages of this method are more obvious. This shows the algorithm is more effective in large-scale environment. In terms of cost QoS, the algorithm is always maintained at a lower level, which has obvious advantages compared with the three-step strategy and the GA algorithm.

ACKNOWLEDGMENT

This work is support by the Nanjing University of Posts and Telecommunications Research Fund(NO.NY209017).

REFERENCES

[1] Yu Y, Chen J, Lin S, et al. A dynamic QoS-aware logistics service composition algorithm based on social network[J]. IEEE Transactions on Emerging Topics in Computing, 2014, 2(4): 399-410.

[2] O'Sullivan J, Edmond D, Ter Hofstede A. What's in a Service?[J]. Distributed and Parallel Databases, 2002, 12(2-3): 117-133.

[3] Lecue F, Mehandjiev N. Seeking quality of web service composition in a semantic dimension[J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(6): 942-959.

[4] Phan M, Hattori F. Automatic web service composition using congolog[C]//26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06). IEEE, 2006: 17-17.

[5] Naseri M, Towhidi A. Qos-aware automatic composition of web services using ai planners[C]//Internet and Web Applications and Services, 2007. ICIW'07. Second International Conference on. IEEE, 2007: 29-29.

[6] Huang Z G, Xu J X, Yu Y. Three-stage algorithm for dynamic service composition[J]. Computer Integrated Manufacturing Systems, 2012, 18(8): 1711-1718.

[7] P. Rodriguez Mier, C. Pedrinaci, M. Lama, and M. Mucientes. An Integrated Semantic Web Service Discovery and Composition Framework[J]//IEEE Transactions on Services Computing, 2015 (DOI 10.1109/TSC.2015.2402679).

[8] Hu Q. Composite web service selection system based on QoS linearization and shortest path[J]. China Sciencepaper/Zhongguo Keji Lunwen, 2012, 7(4): 267-276.

[9] Rodriguez-Mier P, Mucientes M, Lama M. A dynamic qos-aware semantic web service composition algorithm[C]//International Conference on Service-Oriented Computing. Springer Berlin Heidelberg, 2012: 623-630.

[10] Canfora G, Di Penta M, Esposito R, et al. A framework for QoS-aware binding and re-binding of composite web services[J]. Journal of Systems and Software, 2008, 81(10): 1754-1769.

[11] Dao C, Xu C, Chunlai C. Semantic and rules based upon mediator dynamic web service composition in logistics information application[C]//Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on. IEEE, 2008: 532-536.