A Framework for Transmission Cost Aware Service Selection

Zhiling Luo
Computer Science
Zhejiang University
Hangzhou, China
Email: luozhiling@zju.edu.cn

Ying Li
Computer Science
Zhejiang University
Hangzhou, China
Email: cnliying@zju.edu.cn

Jianwei Yin
Computer Science
Zhejiang University
Hangzhou, China
Email: zjuyjw@cs.zju.edu.cn

***Abstract*—The procedure of picking services bound to abstract tasks is usually called service selection in Service Oriented Architecture. In recent years, most studies focus on improving the Quality of Service (QoS) of the composed service. These techniques, however, are facing a new challenge, brought by the big data era, namely, the time and money wasted in data transmission, called transmission cost, cannot be optimized locally like QoS. To address this challenge, in this paper, we study and formalize the problem of transmission cost aware service selection, named TcSS. Owing to the insufficient service transfer rates, we propose a framework on a relaxation problem by making use of the service network ontology structure. The entire framework comprises two stages, an off-line stage to arrange the service network information from logs and an online stage to satisfy the service selection requirement efficiently. The solution of the relaxation problem is an approximation of the original TcSS with the approximate ratio guarantees. Finally, extensive experiments on real data establish the effectiveness and efficiency of our approach.**

***Keywords*-service selection, transmission cost, service location, graph analysis, clustering**

I. Introduction

Recently more information systems, such as enterprise workflow management system (WfMS) [1] and Office Automation (OA), are constructed by SOA (Service Oriented Architecture) [2]. Service computing techniques are becoming more and more practical. Meanwhile, the ideal theories and methods of service computing are facing new challenges caused by the practical industry environment. Specifically, existing techniques, focusing on supporting functional requirement, ignore the energy and money wasted in fulfillment. **Service selection**, one of the most critical steps in service composition, determines the nonfunctional qualities of service process in varieties aspects, such as the price, reliability and reputation [3]. Most methods, presented by different researchers and communities, are based on the optimal selection on QoS (Quality of Service) [4]. With the growth of data volume, data transmission is becoming more and more important [5]. For instance, image data transferred for batch image processing task can be as large as a few Gigabytes. The time or money wasted in the data transmission is the **transmission cost**. For huge data, the transmission cost cannot be ignored (e.g. batch image processing task). The transmission cost can be identified by the response time in classical service selection approaches. However, it is not accurate enough because the response time just represents the time recorded from invoking service to responding user. The response time is usually much smaller than the transmission time, especially when the size of data is huge. Service selection technique targeting the transmission cost is the **Transmission cost aware Service Selection** (abbr. TcSS). Differing with other service qualities, transmission cost has two basic characteristics.

1) **Globally optimized.** The whole transmission cost of a service process is a global optimization on all participant services. Because transmission cost is context sensitive, there is not an optimal choice without considering the services selected before and after. The lowest transmission cost happens only with all nearby services have the smallest transmission cost. Therefore, Local optimization policy based approaches [6] cannot handle TcSS.
2) **Runtime environment related.** The runtime environment affects the transmission cost in practice because of two different organizational styles, **Centralized controlling** (abbr. Co) and **Self-organizing** (abbr. So) [7].

Therefore, in this paper, we address TcSS and propose a unified solution framework for both Co-TcSS and So-TcSS. Our solution is based on following two key ideas.

- The transmission cost can be estimated by the transfer rate and the data volume. The former can be extracted from **service logs** and the latter can be fetched from the user requirement.
- Services are not randomly distributed in the network. A certain number of services are naturally gathered as a service set (the gathering assumption [8]). This network feature can be used to predict the transfer rates missed in logs.

Here are two concrete challenges in solving TcSS.

- **Insufficient transfer rates**. Fetching transfer rate from logs is easy while the data to be fetched from may be insufficient. Taking 1000 services for an example, the number of possible data transferring logs is 1000×1000 while the possible logs may be as small as

30000. Therefore the density[1] of data is $30000/(1000\times 1000) = 3\%$.It means that it is difficult to use the limited logs to select the optimal services directly. Aiming at overcoming this challenge, we propose the **relaxation problem** of original TcSS and solve the relaxation problem with high efficiency and accuracy.

- **Query efficiency**. Service selection is an interaction between the system and user. It is important to ensure a short calculating time and quick response. To face this challenge, our solution divides the time-wasting calculation in the off-line stage and leaves an efficient online stage. The efficiency analyzed at the end of this paper also shows that our online query is efficient.

To overcome these challenges, we divide the solution framework into two stages. The first stage is fetching the service transfer rate matrix from service logs. In this stage, we propose an approach, called **Threshold Spanning-tree Clustering** (abbr. TStC), to cluster the services. The second stage is handling a selection query online using the service clusters by a shortest path algorithm, called **Short-path Selection** (abbr. SpS). Two sub-algorithms, Co-SpS and So-SpS, are introduced in detail for Co style and So style, resp. The efficiency and effectiveness of our approaches are confirmed by strict theoretical reasoning and experimental reports.

Contribution: Our major contributions in this paper are summarized as follows.

- The transmission cost aware service selection (TcSS) and two sub-problems (Co-TcSS and So-TcSS) are studied and formalized.
- A united framework for both Co-TcSS and So-TcSS is proposed.
- We propose an off-line algorithm, called TStC, to predict the location of the services and the distance matrix by service logs.
- We extend So-SpS to a united online selection algorithm, called SpS, using the service location and distance information.

The rest of this paper is organized as follows: Section II introduces preliminaries, formalizes TcSS, including Co-TcSS and So-TcSS, and the relaxation problem, resp. Section III illustrates the solution framework. Section IV discusses the evaluation on our approaches. Finally, discussion and conclusion are respectively given in section V and VI.

II. Problem Formalization

A. Service Selection

Service selection, a critical part of service composition, is picking out appropriate services to fulfill specific tasks in a service process. We note that the candidate services can be represented as a matrix, called **candidate service matrix**

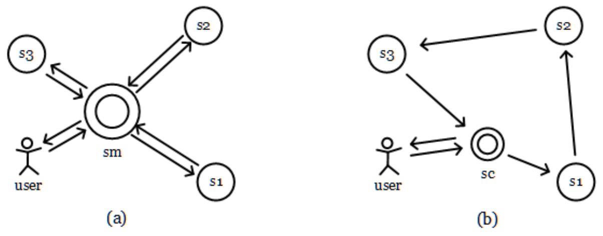

Figure 1. Two organization styles: figure (a) is Co style and figure (b) So style. $s1, s2, s3$ represent participant services, sm the manager and sc the collaborator.

(CSM), denoted by S. Each column of CSM is a candidate service group for a task. The column number, denoted by M, is the amount of tasks and the row number, denoted by N, is the amount of candidate services for a task. To simplify the discussion, we assume that the number of candidate services is the same for each group. If it is not the same, we can set N equal to the largest service number and fill 0 for empty positions in other groups. S_{ij}, or $S_{(ij)}$ denotes the No. j candidate service for task t_i. A general definition of service selection is present as follows.

Definition II.1 (General Service Selection) *Given a process p consisting of tasks $\mathcal{T} = t_1, t_2, ..., t_M$ and the candidate service matrix S in the size of $N \times M$, service selection is selecting the optimal $\hat{S}_i, i \in [1, M]$ for each task t_i from $S_{i1}, S_{i2}, ..., S_{iN}$ to optimize the process in evaluation indicator $\mathcal{F}(p)$.*

In the rest of this paper, we use the vector Θ to denote the selecting result. Namely, for any $\theta_i = k$, it means selecting S_{ik} as $\hat{S}_i$. Therefore, Θ is the variable for the evaluation indicator $\mathcal{F}$. And $\mathcal{F}(p)$ in above definition is exactly $\mathcal{F}(\Theta)$.

B. Two Organization Styles

Before we define the evaluation indicator $\mathcal{F}(\Theta)$ for transmission cost, we need to specify the data transmission deeply. The executing environments of the service process adopt two organization styles, due to different service protocols and standards.

- **Centralized-controlling (Co)**. A centralized-controlling service process has a center service, called **manager**. In this paper, we use s^M (or sm in some figures)[2] to denote manager. It is usually a special service receiving the user's request and controling the data communication of different participant services. Most composite Web services adopt this style according to the standard of BPEL4WS (BPEL for Web Service).
- **Self-organizing (So)**. Every participant services is e-qual in a self-organizing service process. The ser-

[1]The detailed definition of data density is defined in (12) in Section III-B1

[2]The lower-case s stands for a normal service and the capital S stands for CSM.

vice communicates with others by sending/recieving messages directly. Besides, a **collaborator** is used to handle the user's request. In this paper, we use s^C (or sc) to denote collaborator. This style supports a high efficient data transferring and always be used in service collaboration [9] (e.g. data backup service and video processing service).

Figure 1 shows these two organization styles. Considering a process calling $s1$, $s2$ and $s3$ sequentially, the data transferring paths of two styles are totally different. In (a), namely Co style, the path is $user \rightarrow sm \rightarrow s1 \rightarrow sm \rightarrow s2 \rightarrow sm \rightarrow s3 \rightarrow sm \rightarrow user$. In (b), namely So style, the path is $user \rightarrow sc \rightarrow s1 \rightarrow s2 \rightarrow s3 \rightarrow sc \rightarrow user$. The different data transferring paths determine the different transmission cost calculating methods.

C. Transmission Cost

The **transmission cost**, in a narrow sense, is the time or money wasted in the data transferring. Since the money can also be measured by the time with the appropriate economic converting function, which is beyond the topic of this paper, we use the transmission time to represent the transmission cost. As we have already known, the transmission time is the quotient of data volume and transfer rate. The data volume, denoted as ω, is available for a particular selection query. However, the transfer rate is hard to determined because the complexity of the network. We can model the network as an undirected graph $G =< V, E, \Psi >$, called **service graph**. V is the set of services in the environment, E denotes the set of edges and $\Psi : E \rightarrow \mathbb{R}^*$ represents the function from edges to weights. An edge between two services means a known data transfer rate and the weight is the specific value. The weight can be 0 or a positive real number. If a weight is 0, it means that the transfer rate is unknown.

We denote $\mathcal{B}$ as the adjacency matrix of G. Therefore $\mathcal{B}(s_i, s_j) = \Psi(e(s_i, s_j))$. For a determined network, we can define a bandwidth matrix $\bar{\mathcal{B}}$, in which $\bar{\mathcal{B}}(s_i, s_j)$ means the bandwidth between s_i and s_j. Differing with $\mathcal{B}$, the bandwidth, denoted as ϵ, in $\bar{\mathcal{B}}$ is non-zero and determined while unknown.

Definition II.2 (Transmission Cost of Co) *Given the participant services $s_1, s_2, ..., s_M$, the manager service s^M and the data volume ω_i, the transmission cost $\mathcal{I}^{Co}$ is determined by*

$$\mathcal{I}^{Co} = \sum_{i=1}^{M} \frac{2\omega_i}{\bar{\mathcal{B}}(s_i, s^M)} \tag{1}$$

Definition II.3 (Transmission Cost of So) *Given the participant services $s_1, s_2, ..., s_M$, the collaborator service s^C and the data volume ω_i, the transmission cost $\mathcal{I}^{So}$ is determined by*

$$\mathcal{I}^{So} = \frac{\omega_1}{\bar{\mathcal{B}}(s^C, s_1)} + \sum_{i=2}^{M} \frac{\omega_i}{\bar{\mathcal{B}}(s_{i-1}, s_i)} + \frac{\omega_{M+1}}{\bar{\mathcal{B}}(s_M, s^C)} \tag{2}$$

D. Problem Statements

As we defined, transmission costs of Co style and So style are different. Therefore, TcSS can also be divided into two sub-problems, Co-TcSS and So-TcSS.

Definition II.4 (Co-TcSS) *Given a process p consisting of tasks $\mathcal{T} = t_1, t_2, ..., t_M$ and the candidate service matrix S, in the size of $N \times M$, Co-TcSS is selecting the optimal $\hat{S}_i, i \in [1, M]$ for each task t_i from $S_{i1}, S_{i2}, ..., S_{iN}$ to minimize evaluation indicator $\mathcal{I}^{Co}(\Theta)$.*

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & \mathcal{I}^{Co}(\Theta) \\ s.t. \quad & \theta_i \in \{1, ..., N\}, i \in 1, ..., M \end{aligned} \tag{3}$$

Definition II.5 (So-TcSS) *Given a process p consisting of tasks $\mathcal{T} = t_1, t_2, ..., t_M$ and the candidate service matrix S, in the size of $N \times M$, So-TcSS is selecting the optimal $\hat{S}_i, i \in [1, M]$ for each task t_i from $S_{i1}, S_{i2}, ..., S_{iN}$ to minimize evaluation indicator $\mathcal{I}^{So}$.*

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & \mathcal{I}^{So}(\Theta) \\ s.t. \quad & \theta_i \in \{1, ..., N\}, i \in 1, ..., M \end{aligned} \tag{4}$$

E. Relaxation Problem

The original TcSS, either Co-TcSS or So-TcSS, is difficult to be solved accurately because of following two reasons.

- Each bandwidth ϵ in $\bar{\mathcal{B}}$ is unknown. A detailed transfer rate $\hat{b}$ is under a certain distribution $f(\epsilon)$. The transfer rate b, in $\mathcal{B}$, collected from log is sampled from $f(\epsilon)$. It is difficult to estimate the accurate bandwidth ϵ by transfer rate b.
- Some transfer rate data cannot be collected, remains 0, in transfer rate matrix $\mathcal{B}$. The estimation is less accurate if the transfer rate matrix is more sparse.

Therefore, we need use the characteristics of the service network in solving TcSS. As we have studied in previous work [8], the services are not randomly distributed in the Internet. On the contrary, some services are located together because they belong to the same service provider. The cluster of services is called **service area** (area, for short). The services in a same area can communicate with each other in a quite high speed. Formally, we define a partial relation R_λ on those services in the same area by a threshold λ.

$$s_i R_\lambda s_j \Leftrightarrow \mathcal{B}(s_i, s_j) > \lambda \tag{5}$$

The threshold λ is the smallest transfer rate between two services in the same area or the largest rate between two services in different areas. λ can be summarized from all of transfer rate b in $\mathcal{B}$ by solving the following losing function minimizing problem. In (6), $N_\mathcal{B}$ is the number of none-zero elements in $\mathcal{B}$ and L is the number of services in the context.

$$\lambda = \arg \underset{x}{\text{minimize}} \quad \frac{1}{N_\mathcal{B}} \sum_{i,j \in [1,L] \& \mathcal{B}(i,j) > x} e^{x - \mathcal{B}(i,j)} \tag{6}$$

There are two important corollaries about the transfer rate b from the gathering assumption [8].

Corollary II.1 (Transitivity)

$$\forall s_i, s_j, s_k, \mathcal{B}(s_i, s_j) > \lambda \wedge \mathcal{B}(s_j, s_k) > \lambda \Rightarrow \mathcal{B}(s_i, s_k) > \lambda \tag{7}$$

Corollary II.2 (Uniformity) *Given the transfer rates $b_1, b_2, ..., b_{n*(n-1)/2}$, n is the service number, in an area, the distribution function $b_i \sim f(\epsilon_i, \kappa_i)$ and $\epsilon_1 = \epsilon_2 =, ..., \epsilon_{n*(n-1)/2}$, $\kappa_1 = \kappa_2 =, ..., \kappa_{n*(n-1)/2}$. ϵ_i is the bandwidth and κ_i denotes other parameter.*

From the perspective of set theory, (7) shows that relation R_λ is transitive. And the area is essentially the transitive closure of R_λ. The area of the service s is called the **location**, denoted as $C(s)$. We use the **area distance** $D(C_i, C_j)$ to represent the reciprocal of average transfer rate between two service areas.

With the definition of service area and area distance, we introduce **service distance**, denoted as D^*. The service distance between two services s_i and s_j is determined by following theorem, from the gathering assumption.

Theorem II.3 (Service Distance) *The distance of two services is either the area distance of their location if they belong to different areas or 0 if they belong to the same area.*

$$D^*(s_i, s_j) = \begin{cases} D(C(s_i), C(s_j)) & \text{if } C(s_i) \neq C(s_j) \\ 0 & \text{if } C(s_i) = C(s_j) \end{cases} \tag{8}$$

It notes that area distance and service distance are both opposite metrics with the transfer rate, a high transfer rate means a short distance. Considering two services s_i and s_j, we use $1/D^*(s_i, s_j)$ to replace $\bar{\mathcal{B}}(s_i, s_j)$ approximately. With service distance, we provide the relaxation problem of TcSS.

Definition II.6 (Relaxation Co-TcSS) *Given a process p consisting of tasks $\mathcal{T} = t_1, t_2, ..., t_M$ and the candidate service matrix S, in the size of $N \times M$, relaxation Co-TcSS is selecting the optimal $\hat{S}_i, i \in [1, M]$ for each task t_i from $S_{i1}, S_{i2}, ..., S_{iN}$ to minimize the sum of service distances.*

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & \sum_{i=1}^{M} 2D^*(s_i, s^M)\omega_i \\ s.t. \quad & \theta_i \in \{1, ..., N\}, i \in 1, ..., M \end{aligned} \tag{9}$$

Definition II.7 (Relaxation So-TcSS) *Given a process p consisting of tasks $\mathcal{T} = t_1, t_2, ..., t_M$ and the candidate service matrix S, in the size of $N \times M$, relaxation So-TcSS is selecting the optimal $\hat{S}_i, i \in [1, M]$ for each task t_i from $S_{i1}, S_{i2}, ..., S_{iN}$ to minimize the sum of service distances.*

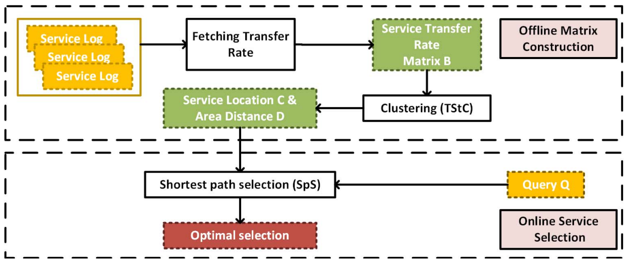
Service Log
Service Log
Service Log
Fetching Transfer Rate
Service Transfer Rate Matrix B
Offline Matrix Construction
Service Location C & Area Distance D
Clustering (TStC)
Shortest path selection (SpS)
Query Q
Optimal selection
Online Service Selection

Figure 2. TcSS solution framework: The above part is the off-line stage and the below part is the online stage.

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & D^*(s^C, s_1)\omega_1 + \sum_{i=2}^{M} D^*(s_{i-1}, s_i)\omega_i \\ & + D^*(s_M, s^C)\omega_{M+1} \\ s.t. \quad & \theta_i \in \{1, ..., N\}, i \in 1, ..., M \end{aligned} \tag{10}$$

A service s_Θ is selected by the relaxation problem, (9) or (10), And $s_{\bar{\Theta}}$ is the optimal service selected by the original problem, (2) or (3). The approximate ratio is defined as (11).

$$\sigma = \frac{\mathcal{B}(s_\Theta, s)}{\mathcal{B}(s_{\bar{\Theta}}, s)} \tag{11}$$

In (11), s is a random service. The expectation of this approximate ratio is quite close to 1. It supports that our solution for the relaxation problem is an appropriate approximation of original solution.

III. Framework

A. Overview

To solve the relaxation TcSS (Co-TcSS or So-TcSS), our solution framework consists of two stages as represented in Fig. 2.

- **Off-line Stage**. The off-line stage extracts the transfer rate matrix $\mathcal{B}$ from service logs, clusters the service into different areas as service location C and specifies the area distance matrix D.
- **Online Stage**. The online stage selects the optimal services using the service location C and distance matrix D.

B. Off-line Stage

1) Fetching Transfer Rate: As the first step in off-line stage, the transfer rate matrix $\mathcal{B}$ is extracted from logs. In detail, this phase consists these steps:

- Summarizing the logs from each service and constructing the vertex set V. Each vertex s denotes a service.
- Iterating logs, connecting s_i with s_j by an undirected edge $e^*(s_i, s_j)$ and marking weight $\Psi(e^*(s_i, s_j)) = b$ if there is a log record that service s_i sends data to s_j in transfer rate of b. We use E^* to denote the set of edges e^*.

Algorithm 1 TStC algorithm

Input: $\mathcal{B}$: sparse transfer rate matrix in size of $L \times L$. λ : the transfer rate threshold.
Initialize: C =zeros(L).
Output: C: service location. D: area distance matrix.

```
1: for $i = 1$ to $L - 1$ do
2:     $[S_1, S_2]$ =BiggestTransferRateFromDiffCluster($\mathcal{B}, C$);
3:     if $\mathcal{B}(S_1, S_2) < \lambda$ then
4:         break;
5:     end if
6:     if $C(S_1) == 0 \&\& C(S_2) == 0$ then
7:         $C$ =AddNewCluster($S_1, S_2, C$);
8:     else if $C(S_1) == 0$ then
9:         $C(S_1) = C(S_2)$;
10:    else if $C(S_2) == 0$ then
11:        $C(S_2) = C(S_1)$;
12:    else
13:        $C$ =MergeCluster($C(S_1), C(S_2), C$);
14:    end if
15: end for
16: $K$ =NumberOfCluster($C$);
17: $D$ =zeros($K, K$);
18: for $i = 1$ to $K$ do
19:     for $j = i + 1$ to $K$ do
20:         $C_1$ =ClusteFromNumber($C, i$)
21:         $C_2$ =ClusteFromNumber($C, j$);
22:         $s$ =SumOfEdgesInTransferRate($C_1, C_2$);
23:         $n$ =NumberOfEdges($C_1, C_2$);
24:         $d = \lambda * n/s$; $D(i, j) = d$; $D(j, i) = d$;
25:     end for
26: end for
27: return $C, D$;
```

- Simplifying the edge set E^* as E by merging the edges connecting the same vertex pair. The weight of the unified edge is the average of weights of the original edges.
- Extracting the adjacency matrix of graph $G =< V, E, \Psi >$ as transfer rate matrix $\mathcal{B}$.

However, transfer rate matrix $\mathcal{B}$ is quite sparse because the logs contain only a small part of service connections. The **density** of $\mathcal{B}$ is defined as (12)

$$density(\mathcal{B}) = \frac{|\{\mathcal{B}_{ij} : \mathcal{B}_{ij} \neq 0, i, j \in \{1, ..., L\}\}|}{L^2} \tag{12}$$

In (12) L is number of services in the environment, namely $L = |V|$ and $\mathcal{B}$ is also in size of $L \times L$.

2) Threshold Spanning-tree Clustering: By (8), to optimize the path distance, we need to determine the location of each service C and the area distance D by clustering the sparse transfer rate matrix $\mathcal{B}$. Most of existing clustering algorithms, including hierarchy clustering and spectral clustering, can be used to find C. However, these algorithms cannot fully take advantage of the service network features. Therefore we propose a spanning-tree based clustering algorithm, called **Threshold Spanning-tree Clustering (TStC)**. The comparison of TStC with other clustering algorithms is reported by the experiments in Section. IV. Algorithm 1 presents the pseudocode of TStC. It consists of following steps:

1) Constructing a maximum spanning tree (always selecting the edge with the largest weight) on service graph G until the weight on selected edge is smaller than λ. The connected edges form a forest with several trees.
2) Classifying these services in the same tree into an service area C_i.
3) Calculating the distance D of two areas C_i and C_j by (13).

$$\begin{aligned} D(C_i, C_j) =& \frac{1}{2} * (\frac{\sum_{C(s_p)=C_i} \sum_{C(s_q)=C_j} \mathcal{B}(s_p, s_q)}{N_{ij}} \\ &+ \frac{\sum_{C(s_q)=C_i} \sum_{C(s_p)=C_j} \mathcal{B}(s_p, s_q)}{N_{ji}}) \end{aligned} \tag{13}$$

In (13), N_{ij} means the number of edges from services in C_i connecting to services in C_j, and similar is N_{ji}. In step 1, we note that for any two services s_i and s_j in the same area, namely $C(s_i) = C(s_j)$ if a group of service $s_1, s_2, ..., s_t$ can be found. And $\mathcal{B}(s_i, s_1) > \lambda$,$\mathcal{B}(s_1, s_2) > \lambda$,...,$\mathcal{B}(s_t, s_j) > \lambda$. Therefore, it is sufficient for the relation R_λ, see Corollary II.1.

Essentially, TStC is an unsupervised learning algorithm and the number of clusters K in our algorithm is learned. It is suitable to the practice, since the number of service areas is unknown.

C. Online stage

1) Short-path Selection: Since the service location C and area distance matrix D can be got by the off-line stage, we can select the optimal services by minimizing (9) (or (10)) giving a specific query Q. We propose a shortest path based selecting algorithm, called **Short-path Selection (SpS)**. The main idea is using a directed graph, called **candidate graph**, to represent the candidate services and the connections. The length of each edge is the product of service distance D^* and weight ω. Fig. 3 presents the candidate graphs for Co style and So style. By selecting the shortest path from the first vertex to the last vertex, we can determine the optimal services. For example, in Fig. 3 (a), the path $sm \rightarrow s11 \rightarrow sm \rightarrow s23 \rightarrow sm \rightarrow s32...sm1 \rightarrow sm$, means selecting $s11$,$s23$,$s32$,...,$sm1$ as the participant services and $\Theta = (1, 3, 2, ..., 1)$. Here is a theorem about the path and TcSS.

Theorem III.1 (Short Path) *The candidate services in the shortest path of candidate graph are the optimal services for TcSS.*

Proof: Considering Co-TcSS at first, if the optimal services construct a path $p(\Theta) = s^m \rightarrow s_{1,\theta_1} \rightarrow s^m \rightarrow s_{2,\theta_2} \rightarrow ... \rightarrow s^m$ and the shortest path is $p(\hat{\Theta}) = s^m \rightarrow s_{1,\hat{\theta}_1} \rightarrow s^m \rightarrow s_{2,\hat{\theta}_2} \rightarrow ... \rightarrow s^m$. Then the path distance of $p(\Theta)$ is $\mathcal{D}^{Co}(\Theta) = \sum_i^M D^*(s_{i,\theta_i}, s^M)\omega_i$ and the path distance of $p(\hat{\Theta})$ is $\mathcal{D}^{Co}(\hat{\Theta}) = \sum_i^M D^*(s_{i,\hat{\theta}_i}, s^M)\omega_i$ using the definition of path distance of So. Since Θ is optimal selection, $\mathcal{D}^{Co}(\Theta) < \mathcal{D}^{Co}(\hat{\Theta})$. However the $p(\hat{\Theta})$ is the

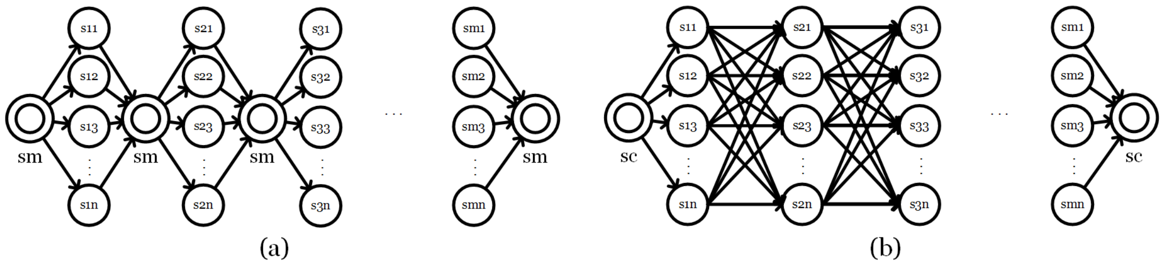

Figure 3. Co path and So path selection: (a) presents the Co path and (b) the So path

Algorithm 2 So-SpS algorithm

Input: C : service location in length K. D: area distance matrix in size $K \times K$. S candidate services matrix in size $N \times M$. Ω: data size weight
Initialize: $l = 1$.
Output: Θ: selection result.

```
1:  $V[l].s = S^c$
2:  $l = 2$
3:  for $i = 1$ to $M$ do
4:      for $j = 1$ to $N$ do
5:          $V[l].s = S(i, j)$; $V[l].j = j$; $V[l].dist = MAX$; $l++$;
6:      end for
7:  end for
8:  $V[l].s = S^C$;
9:  $si$ =FindMinDist($V$);
10: while $si > 0$ do
11:     if isLast($V[si]$) then
12:         $F' = V[si].dist$; break;
13:     end if
14:     for $i$ iterates next group do
15:         if $V[i].dist > V[si].dist + \omega[si] * D(C(V[si]), C(V[i]))$ then
16:             $V[i].dist = V[si].dist + \omega[si] * D(C(V[si]), C(V[i]))$;
17:             $V[i].last = si$;
18:         end if
19:     end for
20:     $si$ =FineMinDist($V$);
21: end while
22: for $i = M$ to 0 do
23:     $\theta[i] = V[si].j$; $si = V[si].last$;
24: end for
25: return $\Theta$;
```

shortest path meaning that $\mathcal{D}^{Co}(\hat{\Theta}) < \mathcal{D}^{Co}(\Theta)$ which is conflict with former description. Therefore, in Co-TcSS, the shortest path ensures that the services selected are optimal. The proof on So-TcSS is similar and omitted here. ∎

Theorem III.1 ensures that the result of SpS is the accurate solution for the relaxation TcSS. Algorithm 2 presents the detailed pseudocodes for Co-SpS. So-SpS is similar to Co-SpS and omitted here, due to the space limitation of this paper.

IV. Evaluation

A. Preliminaries

There are two core algorithms in our solution framework, TStC and SpS.

- To evaluate the effectiveness and efficiency of TStC, we made some experiments comparing with other approaches. The reports below support that TStC is an appropriate algorithm with highest accuracy and acceptable efficiency.
- The effectiveness of SpS is guaranteed by theorem III.1. And its efficiency is affected by two parameters: the candidate group number M and the service number in each group N. The time complexity is $\mathcal{O}(MN)$. In most situation, the number of candidate services each group is confirmed. Therefore, the time complexity is $\mathcal{O}(M)$. The experiment reports on So-SpS can be found in our previous work [8]. The Co-SpS is similar in efficiency. Therefore we omit the experiment on SpS in this paper.

B. Experiments

In this part, we report the experimental evaluations by comparing our method (TStC) with the state-of-the-art algorithms. The programs are implemented in Matlab and performed on a computer with Intel(R) Xeon X5670, 2.93GHz CPU and 8 GB memory.

We set up a service environment with 10 computers and deploy 500 services programmed by Python on these computers, 50 each. Each service writes the logs when it receives data, executes calculating and sends data. Before our experiment, we randomly call some services to accumulate the logs and collect these logs.

We compare the accuracy of the proposed approach (**TStC**) with existing approaches: **hierarchical clustering** [10] and **spectral clustering** [11] (SC). The different linkage way in hierarchical clustering produce different effect, we use **inner squared distance hierarchical clustering (wardHC)**, **nearest distance hierarchical clustering (singleHC)**, **furthest distance hierarchical clustering (completeHC)** and **group average hierarchical clustering (averageHC)**. The detailed experiment plan contains these steps:

1) Extracting transfer rate matrix ($\mathcal{B}$) from logs.
2) Testing each clustering algorithm (TStC, wardHC, singleHC, completeHC, averageHC and SC) on $\mathcal{B}$. Each clustering outputs the service location C and the area distance matrix D.
3) Constructing CSM S and data volume parameter ω from user query Q.

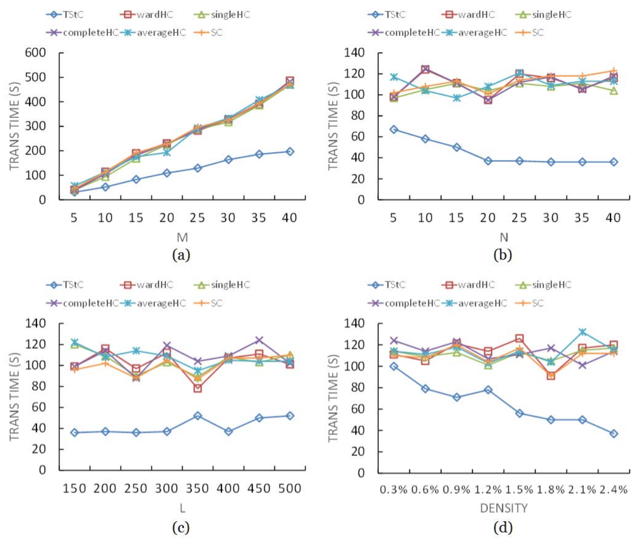

Figure 4. Exp. on effectiveness with different parameter combinations. (a) uses different M, (b) uses different N, (c) uses different L and (d) uses different density.

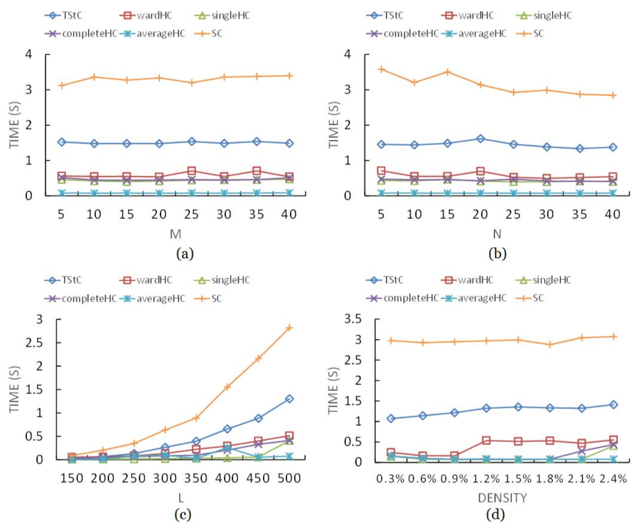

Figure 5. Exp. on efficiency with different parameter combinations. (a) uses different M, (b) uses different N, (c) uses different L and (d) uses different density.

4) Executing SpS on each pair of $< C, D >$ and returning the selection result $Theta$, namely Θ.
5) Executing the service process in different Θ and recording the transmission time.

In this experiment, the density of transfer rate matrix is 2.4%. And we set $\lambda = 143$ by solving (6).

1) Effectiveness: The experiments results vary on four parameters: 1) M defined as the candidate group number, 2) N defined the service number in each group, 3) the density of transfer rate matrix $\mathcal{B}$ and 4) L the size of $\mathcal{B}$. We use whole transmission time (trans time, for short) to represent the transmission cost, evaluating the effectiveness of each algorithm. The less the trans time, the better.

Fig. 4 reports the comparison of different algorithms using different parameters. In Fig. 4 (a), we use different candidate group number M. We note that the trans time of all algorithms grow with the candidate group number M. And **TStC** performs better than other algorithms. In (b), we use different service number N for each group. **TStC** outperforms others with the lowest transmission cost. But at length, with increasing N, trans time is close to steady-state. It is because that our algorithm can select the optimal services when $N = 20$, as shown in figure. More candidate services (25 or more), do not change the optimal selection result. In (c), we use different service number (L) of transfer rate matrix $\mathcal{B}$ and TStC outperforms batter. In (d), we vary the density of transfer rate matrix $\mathcal{B}$ from 0.3% to 2.4%. We can find that **TStC** gets a better effect within the growing of density. Through out this experiment results, **TStC** always gets a better selection than other algorithms.

2) Efficiency: Even though TStC is used in the off-line stage and it doesn't affect the efficiency of the online query, we still evaluate the time complexity between other algorithms.

The efficiency of different algorithms are reported in Fig. 5. Fig. 5 (a) (b) (c) and (d) use different parameters similar to Fig. 4. As reported in the figure, **TStC** is faster than **SC**. And **HC** is the fastest but the selection is not optimal as we find in above experiment results. Especially, in (c), we can find that both **TStC** and **SC** increase with L. The time complexity of either **TStC** or **SC** is $\mathcal{O}(L^2)$.

V. Related Works

There are lots of research achievements in service selection. However, the complexity of the problem makes it hard to find a general efficient method for all situations. This section concentrates on development of the QoS based selection and the comparison of our approach.

In recent years, the quality of service (QoS) [12] based selection is the major research direction. The QoS of a service is a set of features, including reputation, price, reliability, etc. Different service process structures (sequence, loop, branch and parallel) compose QoS in different ways. For instance, the price of sequence structure is the sum of the price of each participant service. The price of parallel structure is the max price of each participant service. The QoS based selection is also called QoS aware selection or QoS driven selection.

The single-destination of QoS based selection is optimizing one QoS indicator and the representative works including the work of Zeng, et al. [3]. He focused on dynamic selection of services and used a global planning method to find the optimal services. His work made the foundation of using global optimization method in service selection. After that, most discussions on service selection convert it into a global integer optimization problem, also call Integer Programming. Due to the time complexity of integer programming, which is NP-hard, some researchers introduced Mixed linear programming (MIP) in service selection. Ardagna et al. extended the MIP model to include local constraints [6]. However, the model has the exponential time complexity, which performs poorly with increasing load. Like [6], Alrifai et al. decomposed the original quality constraint into local constraints [13]. Searching with local constraints has a linear time complexity. However, the decomposition process still has an exponential time complexity.

Besides one QoS based selection, more researchers paid attentions on the synthesis selection on two or more QoS indicators [14] at the same time. The **multi-QoS selection problem** is a multi destinations optimization problem. Skyline was introduced to identify the multi QoS selection. A representative work is WS-Sky [15], a complete service selection framework proposed by Benouaret et al. WS-Sky does not only use skyline in service selection but also cover the uncertain QoS. Other topics on QoS based service selection more concentrate on the detailed problem in practice. Qu et al. discussed a context-aware cloud service selection method in [16].

All these QoS-based selection approaches cannot handle TcSS because differing with QoS indicators, transmission cost is context sensitive. There is not an optimal choice without considering the services selected before and after. The lowest transmission cost happens only with all nearby services have the smallest transmission cost. Therefore, it is impossible to find optimal services using these approaches.

VI. Conclusion and Future Work

In this paper, we study and formalize the problem of transmission cost aware service selection (addr. TcSS). TcSS cannot be solved suitably by classical QoS based service selection approaches because the transmission cost is sensitive to context. Another challenge comes form the insufficient of network transferring rate data. Therefore, a relaxation problem is proposed to overcome the problem of data insufficient by taking advantage of the network structure. To solve the relaxation problem with high efficiency and accuracy, we propose a solution framework consisting of two stages. In the first stage, called the off-line stage, the transfer rate data between services is extracted from service logs. Then services are clustered into different areas using a spanning-tree based algorithm, called TStC. In the second stage, called the online stage, a user's query is satisfied by selecting the optimal service using a shortest path based algorithm (SpS). Both the effectiveness and efficiency of our approach are evaluated by the extensive experiments on real-world data, compared with existing approaches. Furthermore, the approximate ratio of our solution is estimated and its mathematical expectation is a constant close to 1. Finally, an interesting direction for future work concerns the unified solution framework that covers both TcSS and QoS based service selection.

VII. Acknowledgements

The project supported by Zhejiang Provincial Natural Science Foundation of China under grant No.LY15F02007 and Science and technology Program of Zhejiang Province under grant No. 2012C01037-1.

References

[1] W. M. P. Van der Aalst, *Process mining: discovery, conformance and enhancement of business processes*. Springer, 2011.

[2] SOA, "http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf." http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

[3] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311 – 327, 2004.

[4] I. Guidara, N. Guermouche, T. Chaari, S. Tazi, and M. Jmaiel, "Pruning based service selection approach under qos and temporal constraints," in *ICWS*, ICWS, pp. 9–16, IEEE, 2014.

[5] M. Perrone, L. Liu, L. Lu, K. Magerlein, C. Kim, I. Fedulova, and A. Semenikhin, "Reducing data movement costs: Scalable seismic imaging on blue gene," in *IPDPS*, IPDPS, pp. 320–329, IEEE, 2012.

[6] D. Ardagna and B. Pernici, "Global and local qos constraints guarantee in web service selection," in *ICWS*, ICWS, IEEE, 2005.

[7] C. Peltz, "Web services orchestration and choreography," *Computer*, vol. 36, no. 10, pp. 46–52, 2003.

[8] L. Zhiling, L. Ying, and Y. Jianwei, "Location: A feature for service selection in the era of big data," in *ICWS*, ICWS, pp. 515 – 522, IEEE, 2013.

[9] X. Zhu, B. Wang, and S. Wang, "Reputation-driven web service selection based on collaboration network," in *ICWS*, ICWS, pp. 704–705, IEEE, 2011.

[10] R. L. Cilibrasi and P. Vitnyi, "A fast quartet tree heuristic for hierarchical clustering," *Pattern recognition*, vol. 44, no. 3, pp. 662–677, 2011.

[11] W. Chen, Y. Song, H. Bai, C. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 3, pp. 568–586, 2011.

[12] D. A. Menasce, "Qos issues in web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, 2002.

[13] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *WWW*, WWW, pp. 881–890, ACM, 2009.

[14] T. Yu and K.-J. Lin, "Service selection algorithms for composing complex services with multiple qos constraints," in *ICSOC*, ICSOC, pp. 130–143, Springer, 2005.

[15] K. Benouaret, D. Benslimane, and A. Hadjali, "Selecting skyline web services from uncertain qos," in *SCC*, SCC, pp. 523–530, IEEE, 2012.

[16] L. Qu, Y. Wang, M. A. Orgun, L. Liu, and A. Bouguettaya, "Context-aware cloud service selection based on comparison and aggregation of user subjective assessment and objective performance assessment," in *ICWS*, ICWS, pp. 81–88, IEEE, 2014.