The Pennsylvania State University

The Graduate School

The Harold and Inge Marcus Department of Industrial and Manufacturing Engineering

MATHEMATICAL THEORY OF SERVICE COMPOSITION AND SERVICE NETWORKS

A Dissertation in

Industrial Engineering and Operations Research

by

Liying Cui

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

May 2011

The dissertation of LiYing Cui was reviewed and approved* by the following:

Soundar Kumara
Allen E. Pearce/ Allen M. Pearce Professor of Industrial and Manufacturing Engineering
Professor of Computer Science and Engineering
Dissertation Advisor
Chair of Committee

A. Ravi Ravindran
Professor of Industrial and Manufacturing Engineering

Tao Yao
Assistant Professor of Industrial and Manufacturing Engineering

Hong Xu
Professor of Supply Chain & Information Systems

Réka Albert
Professor of Physics and Biology

Dongwon Lee
Associate Professor of Information Systems &Technology

Paul Griffin
Professor of Industrial and Manufacturing Engineering
Peter and Angela Dal Pezzo Head Chair of the Department of Industrial and Manufacturing Engineering

*Signatures are on file in the Graduate School

ABSTRACT

This dissertation addresses the theoretical foundations of service composition and develops network based techniques, mathematical programming, and stochastic models for service composition.

For the first time, a mathematical theory, which forms the foundation to define service science as a rigorous discipline is explored. The theorems of solution existence and convergence are generated. Mutuality/Duality theory of service space and concept space is established.

The Concept Service (CS) network matrix is developed to study the network structure of service composition. The potential of CS network provides insights to identify important services and concepts. This information is used to connect the initial query with sub-components in CS network, thus reducing the computational time for the planning algorithms. A set of heuristic service composition algorithms is developed based on CS network matrix.

Based on the mathematical theory established, three types of multi-criteria programming models are designed to find optimal service composition solutions for the offline-planning problem. The three types of mathematical programming models are: Multi Criteria Programming(MCP), Multi Criteria Goal Programming for Optimal composition(MCGPO) and Multi Criteria Goal Programming for Non-optimal composition(MCGPN). MCP model can generate an optimal solution if solutions satisfying all the customer's functional and nonfunctional requirements exist. In addition, the MCGPO model allows to automatically select a trade-off among the objectives

according to the customers' preferences. MCGPN model is fast for generating satisficing solutions.

Stochastic models are developed to compose service queries under uncertainty. In these models, nested compound work flow and service workload are considered. The algorithms directly utilize prior information of the potential of Concept Service (CS) network matrix to generate a transition matrix. This offers a fast and convenient approach for real time service composition to capture the uncertainty and dynamic phenomena.

Finally, possible applications of service composition in product design, global manufacturing, supply chain and resource allocation are discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am deeply indebted to Dr. Soundar Kumara for his advice and guidance. He is very supportive and encouraging. He is not only a dissertation advisor but also a mentor in many aspects of my Ph.D study. He exposed me to many interesting research topics such as Cyber security, Business intelligence, Revenue management and Service computing.

I am also thankful to Dr. Reka Albert for guiding me in network science, and in paper and book chapter writing, most of which are included in this dissertation. I am thankful to Dr. Dongwon Lee for revising my first paper on web service composition. I am thankful to Dr. Tao Yao, for teaching me stochastic processes, giving me suggestions on conference presentations, and introducing me to other Chinese scientists in service area. I am lucky to have his help.

I am also thankful to Dr. Raj Mittra for his support during the first year of my Ph.D study. He is such a wise and rightful man. I have benefited so much while working with him. I learnt how to deal with many aspects in one's academia career. Except his professional accomplishments, he is doing a lot of charity and sponsoring several scholarships. He has been a successful role model for many scientists.

I am thankful to Dr. Susan Xu, and Dr. A. Ravi Ravindran for teaching me knowledge in their expertise and sharing their research experience. Two service composition models in this dissertation are derived and extended from the knowledge that I learnt in the courses they taught me .

I want to thank Mr. Juan Ernesto de Bedout and Mr. Guillermo Pinochet for their financial support during the last two years of my study. Thank them for giving me the

chance to grow both professionally and personally. I also want to thank my helpful and efficient colleagues in Kimberly-Clark.

I want to thank my parents, who brought me into the world, raised me up and gave me love. I am grateful to have the wonderful parents who always give me the freedom to do what I want. I am blessed to have a very lovely sister, who is very supportive and care about my health (She graduated from a medical school in China).

I am really grateful to my husband, Xiaoling Yang, for his warm support all the way through my Ph.D study. Without him, I couldn't have finished the program in about three and a half years. Thank him for giving me love, joy, support and motivation. Thank my parents-in law for having raised my husband so well. Thank them for their encouragement and support.

I would like to express my sincere thanks to all my wonderful friends at Penn State for their support.

EPIGRAPH

This dissertation is dedicated to those who work on Service Science, those who participate in global collaboration, and those who believe in freedom and ability of mankind.

Chapter 1. Introduction

Proliferation of e-commerce and the growth of information technology have fueled the use of distributed cyber business processes. At the same time, the entire global economy is being catalyzed by utilizing these distributed e-business processes. Studying service as a science is increasingly important in the global economy. In service science, people, machines, products and software are both service providers and service consumers. This dissertation deals with service as a science in general and with specific emphasis on web services. Web service composition enforces to reuse automatic web services to build different kinds of service complexes, which can meet varieties of business applications. Web service composition helps build web-based applications by composing as well as integrating existing web services.

Though service science and service composition have assumed enormous importance, rigorous mathematical foundations of service science are still lacking. In this dissertation, the mathematical theory of service composition is established as a major contribution. Next, a unique concept service (CS) network matrix and a set of service composition algorithms are developed as the second contribution. Further, a set of multi-criteria goal programming models by considering both the optimal and satisfycing solutions based on Quality of Service (QoS) is the third contribution. Finally, dynamic

programming models based on concept service (CS) network matrix considering nested work flow and work load capacity of service providers is addressed in this dissertation.

1.1 Problem Definition

The web service composition problem aims at identifying a set of web services (and work-flow therein) such that the composition of those web services can satisfy users' goals as much as possible. Therefore, a natural objective is to identify the best composition of web services that optimizes the customer's criteria (e.g., cost, execution time which is the process time of a service, reliability, etc.). Several important definitions are given to describe services and web services:

Definition 1-1 Service provider: Any software, machine, people, product etc. that can provide useful function to others can be called as a service provider.

Definition 1-2 Customer: The party which needs a service from a service provider is called a customer of the service provider.

Definition 1-3 Service: A set of functions that a service provider offers to a customer is called a service.

Definition 1-4 Web service (Internet based service): A service provided via WWW or Internet is called a web service.

Definition 1-5 Concept: A concept is a cognitive unit of meaning. Concepts are introduced to describe services in this dissertation.

Definition 1-6 Service network: A network formed where the nodes are services and concepts and the links among nodes are the information flows among the nodes.

Definition 1-7 Service composition: Service composition is a process to select a set of services and their flow sequence to perform a task which cannot be finished by any individual service.

A service is a set of related activities that produce a function to customers. Web services belong to services. However, only those services using data flow as inputs and outputs can be developed as web services. If we only study the business processes composed by web services, we call it a web service composition problem (WSC); otherwise we call it a general service composition problem. Most published works on service composition study WSC problems. A web service is described by pre-conditions (inputs) and post-conditions (outputs).

Definition 1-8 Input: A pre-condition of a service is defined as an input of the service.

Definition 1-9 Output: A post-condition or result of a service is defined as an output of the service.

The relationship among web service composition and general service composition is the following:

$$Web\ Service\ Composition \subset General\ Service\ Composition$$

In order to illustrate service planning, let us consider the following scenario: Olivia wants to travel to Washington, D.C., from State College this Saturday. She would like to return from D.C. on next Friday. She has business meetings on Monday and Tuesday. If the weather is good, she would like to watch baseball game on Thursday. If

the weather is bad then she would like to go to the theater on both days to watch a play. On Thursday she would like to go to a Mexican restaurant for dinner.

She may need to accomplish the following tasks in order to generate a plan:

(1) Book a hotel in Washington D.C.;

(2) Rent a car in Washington D.C.;

(3) Check the weather for Thursday afternoon;

(4) If the weather is good in the afternoon of Thursday, check schedule and venue for the baseball game;

(5) If the weather is bad, buy a ticket for the music performance;

(6) Reserve a seat in a Mexican restaurant for dinner.

Each of these tasks can be accomplished by invoking the appropriate web service. The invocation of the web services required to generate a plan is termed as the web service composition problem. Figure 1-1 shows the service composition process.

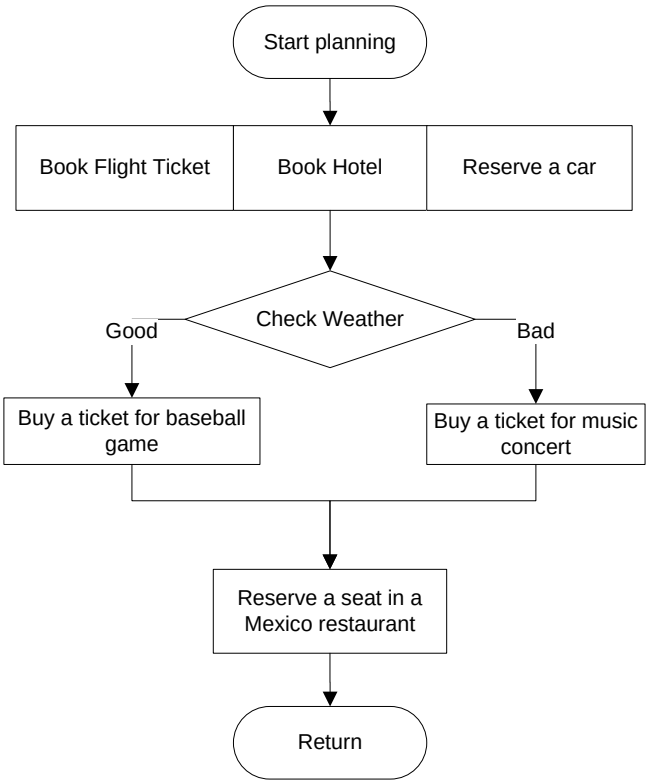


Figure 1-1 An example of a web service composition

1.2 Research Objectives

The research objective is to develop novel methods and to apply tools and concepts from other scientific disciplines into service computing so that the distributed business processes can be fused together to form compound services in a timely and accurate manner. Service composition and service networks are two important issues in service computing. Service composition research issues my include planning algorithms, network analysis, semantics, monitoring, exception handling, re-planning and security as shown in Figure 1-2. The service composition planning algorithms are the foundation of

service computing. There are offline planning (static) and online (dynamic) planning algorithms. A dynamic model overcomes the static assumption on services in service composition. The service composition problem deals with mathematical theory, optimization, and dynamics of service composition. Finally, the network analytics is useful for planning, and make recommendations to service providers and service brokers. This dissertation addresses these issues while building a theoretical basis for service composition.

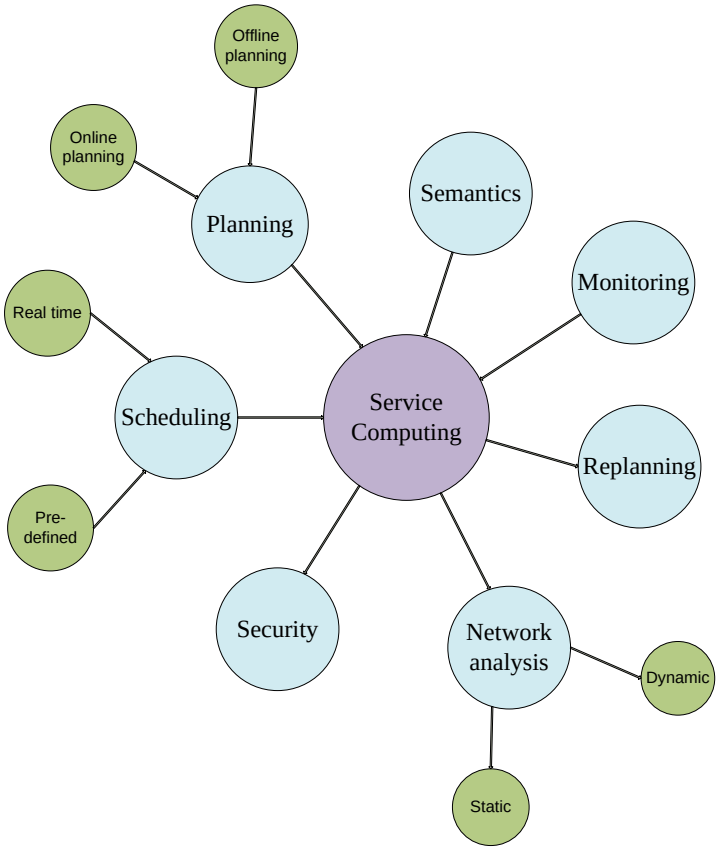


Figure 1-2 Potential research issues in service computing

1.3 Research Problems

Service Science, a new field is in search of a rigorous mathematical foundation. A mathematical theory of service science is needed for establishing service science. Service networks, as a large-scale system of systems, can use network mining techniques to identify the hidden information in it. Only little introductory work is reported on this problem in literature. Optimization algorithms on service networks are needed to be studied further. The Quality of Service (QoS) constraints needs to be considered flexibly in service composition. It is known that the environment of service networks changes from time to time. How to capture these uncertainty and dynamic phenomena in service networks is another important research problem.

Four major research problems addressed in this dissertation are: (1) Establishing a mathematical theory of service science; (2) Identifying hidden information in service networks treating the system of service systems as a large scale network; (3) Solving the service composition problem as an optimization problem yet flexibly considering the QoS constraints and users' preferences; (4) Capturing uncertainty and dynamics in service networks.

1.4 Uniqueness and Contributions

The contributions of this dissertation include:

(1) For the first time, a mathematical theory of service computing via mapping service operators over topology space is developed to provide a foundation for service

science to become a rigorous discipline. Mutuality/ Duality theory of service space and concept space is established.

(2) The Concept Service (CS) network matrix leads to the development of design concepts for service networks. The measures from network science are used to identify important services and concepts, which can be used in the network design as well as in preprocessing for service composition. A set of service composition algorithms is developed based on CS network matrix.

(3) The multi-criteria goal programming models (MCP, MCGPO, MCGPN) are developed to cater to the offline-planning problem in service composition. Compared with previous work in literature, these optimization models are unique as they are able to handle the Quality of Service constraints in a flexible manner. MCP model can generate an optimal solution if there are solutions that satisfy all the customer's functional and nonfunctional requirements. In addition, the MCGPO model allows to automatically select a trade-off among the objectives according to the customers' preferences. MCGPN model is fast for generating satisficing solutions.

(4) Stochastic models are developed to perform online service composition under uncertainty. This algorithm directly utilizes the prior information in the potential of Concept Service (CS) network matrix. It is a fast and convenient model for real time service composition to capture the uncertainty and dynamic phenomena.

1.5 Organization of the Dissertation

The characterizations of service composition are introduced in Chapter 2. A general summary of service composition techniques and planners is given in Chapter 3. The mathematical foundation of service computing is established in Chapter 4. Further, a unique concept service (CS) network matrix and a set of service composition algorithms are developed in Chapter 5. In Chapter 6, a set of multi-criteria goal programming models are introduced as a service composition method, which considers both the optimal and compromising QoS attributes. In Chapter 7, dynamic programming models are introduced to consider work load capacity of service providers and nested work flows. In Chapter 8, several applications of service science are explored. Finally, in Chapter 9, future work and conclusions drawn are reported based on the theoretical and application developments in this dissertation. In each chapter, only the representative equations are numbered to avoid unnecessary cumbersome.

Chapter 2. Service Composition: Characterization of Important Aspects

This chapter lays the groundwork for the research reported in this dissertation. The details of flow patterns, semantic aspects, and Quality of Service (QoS), and user preferences are discussed. Definitions of important terms used in service composition literature are studied.

2.1 Flow Pattern

In service composition problems, there are four types of service flow patterns, namely Sequential, Switch, Parallel and Iterative. In Fig. 2(a), Service 1 is the predecessor of Service 2, so these two services are sequential. In Fig. 2(b), either Service 1 or Service 2 is adequate for the whole compound service, so the relationship between Service 1 and Service 2 is a switch (also called "OR Parallel" in this context), with a probability p_1 for using Service 1, and probability p_2 for using Service 2. In Fig. 2(c), both Service 1 and Service 2 are needed to provide input information for the successors, so the two services are parallel (also called "And Parallel" in this context). In Fig. 2(d), Service 1, 2, and 3 are in a loop, so it is named iterative flow pattern. In the example shown in section 1.1, the task of booking an air ticket and the task of booking a hotel in Washington D.C. are sequential, since the number of nights needed in a hotel depends on the departure date from State College. The tasks of booking a hotel and renting a car can be executed in parallel. Buying a ticket for a baseball game and checking the schedule of

the Space Museum are conditional tasks, and the flow pattern is a switch. There is no iterative flow pattern in this plan.

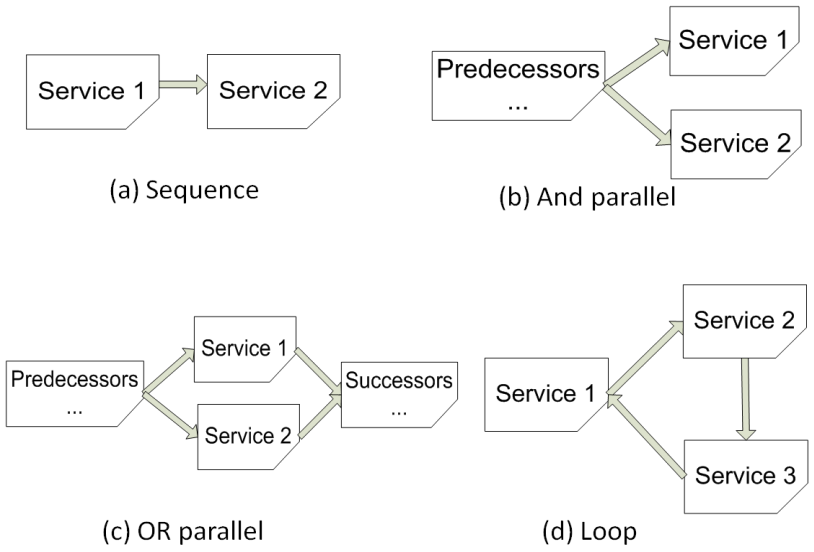


Figure 2-1 Four basic service flow patterns

The data structure of web services is shown in Figure 2-1. Web services are stored in a hierarchical form and are classified into different porttypes; one service may have multiple operations. Each web service operation includes a set of inputs and a set of outputs. Two web service operations can communicate with each other if one's output(s) matches the others' input(s). In order to accomplish the service composition, each web service must specify the properties of the service, such as where it is located (URL), how it is invoked (inputs), and what it does (outputs). To guarantee the successful interaction of web services, the service providers must describe and publish the web services in a specified standard(s), such as WSDL, BPEL4WS, WS-Choreography, and OWL-

S(DAML-S) (*http://www.w3.org/TR/wsdl20*) to guarantee that the services can exchange information among them.

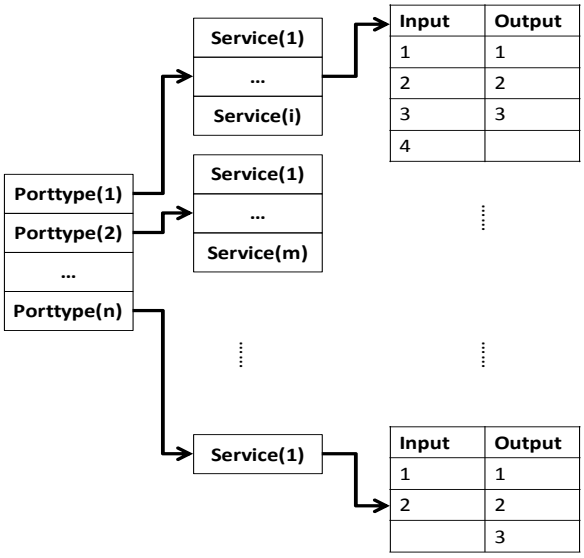


Figure 2-2 Data structure of web service database

2.2 Semantic Aspects

It is assumed that all the input and output attributes of web services are described in standard semantic concepts. So, the web services are semantically interpreted in ontology and registered in service registries such as, for instance, UDDI (http://www.uddi.org). A Semantic Service Matching Algorithm is used to compare the concepts derived from processing ontology and the requirements from the users with the concepts presented in format of published services. Next, a popular semantic matching algorithm in service composition is introduced.

Each concept has its sub-classes and super-classes. The service matching algorithm searches the Web Service Description file and the ontology file to check the

semantic relationship between every pair of concepts. Not only the explicit relations but also the implicit relations between concepts are derived through reasoning in this algorithm. In Table 2-1, there are four basic types of relationships between any two concepts: (i) Concept 1 is the same as Concept2; (ii) Concept1 is an ancestor of Concept2; (ii) Concept1 is an offspring of Concept2; (iv) Concept1 has no relationship with Concept2.

Table 2-1 Types of similarity match between two concepts (C_1, C_2)

Description in words	**Semantic Description**
C_1 exactly matches C_2	$C_1 = C_2$
C_1 partially matches C_2	$C_1 \subset C_2$
C_1 over matches C_2	$C_1 \supset C_2$
C_1 fails to match C_2	No relation

A score can be assigned differently according to the real situations between the pairs of concepts.

$$match(C_1, C_2) = \begin{cases} Score1 & if\ C_1 = C_2 \\ Score2 & if\ C_1 \subset C_2 \\ Score3 & if\ C_1 \supset C_2 \\ 0 & elsewhere \end{cases},$$

Here $Score1 \geq Score2 > Score3$. $C_1 = C_2$ means concept C_1 and concept C_2 are exactly the same; $C_1 \subset C_2$ means concept C_1 is a subclass of concept C_2; $C_1 \supset C_2$ means concept C_1 is the parent class of concept C_2.

Definition 2-1 Concept domination: Concept C_1 dominates concept C_2 if concept C_1 has more information than concept C_2 does.

In general, if concept C_1 is a subclass of concept C_2, concept C_1 has more information than concept C_2 does.

Lemma 2-1 If concept C_1 is a subclass of concept C_2, e.g. $C_1 \subset C_2$, then C_1 dominates C_2.

In order to check whether a service S_1 can be used as another service S_2, we need to compare the input and output concepts. It is preferred that service S_1 needs no more input information than service S_2 does, and generates no less output information than service S_2 does.

Definition 2-2 Service domination: Service S_1 dominates service S_2, if service S_1 needs no more input information than service S_2 does, and generates no less output information than service S_2 does.

Let input represents the total input concept set of service S_1 and service S_2, and output represents the total output concept set of service S_1 and service S_2. The following lemma holds.

Lemma 2-2 Service S_1 dominates service S_2 if $C_i^{S_1} \supseteq C_i^{S_2}$, $i \in input$, and $C_i^{S_1} \subseteq C_i^{S_2}$, $i \in output$.

When a query R is posed, service S can be used as a candidate to fit the request R, if service S dominates the request R. The usability score of an existing web service S to query R can be defined as:

Definition 2-3 Usability score:

$$Usability\ score(R,S) = \sum_{i \in input} match(C_i^R, C_i^S) + \sum_{i \in output} match(C_i^S, C_i^R)$$

The above formula shows us how to calculate the usability score of a service to a query. The usability score of a service to a request consists of the match scores of all the input concepts and output concepts of the request. All the input and output matching scores between the request and the service are summed up into the usability score of a service to the requested service. All candidate services can be sorted according to their usability scores. The service with the highest usability score wins. The candidate services can also be further evaluated according to the Quality of Service (QoS). The disadvantage of the Semantic Matching Algorithm is that it is only used to match a query with existing services, instead of doing service composition.

2.3 Quality of Services (QoS)

Li et.al. (Li, et. al., 2008) proposed the QoS Filtering Algorithm. Recently user preferences also have drawn more attentions in the service-oriented research field (Raman, 2002).. However, some traditional automated service composition approaches cannot meet the personalized user requirements due to the following reasons:

Common knowledge cannot characterize the personalized requirements. For example, if the user wants to go to Washington D.C. from State College, taking a flight minimizes time but driving is also a viable option. In case the user is afraid of flying, or annoyed with the frequent delays airlines have or finds the cost of flying too high, flying

is not the best option. In fact, the users' requirements are changeable and unpredictable in different environments. Therefore it is impossible to forecast all personalized requirements in advance.

Users' requirements not only include the hard constraints, but also soft-constraints. In the traditional approaches, only a user's initial state is considered as a constraint to implement planning, for example, the user may have 5000 dollars budgeted for the trip and she cannot pay more than that. In addition, if the user wants to travel in a modality that both minimizes travel time and offers the least cost, there may be no way to satisfy both of her preferences at the same time. Thus, she has to make a compromise to select a feasible conveyance. This kind of soft-constraint is usually fuzzy and negotiable. It is important to realize that the user-centered service composition should pay much more attentions to flexible user preferences for improving users' satisfaction. In addition, the soft-constraints will also improve the success rate of service composition if compromising feature is considered.

In general, the following five quality of service (QoS) criteria can be considered in service composition: (1) execution price, (2) execution duration, (3) reputation, (4) reliability, and (5) availability.

A web service can contain more than one operation. The operations can be different in terms of functionality and QoS. And the service selection technique can also be applied in operation selection. We can treat each operation as an individual service.

Based on Zeng et.al.'s work (Zeng et.al., 2004), the QoS definition of services can be defined as follows.

Definition 2-4 Price: Given a service S, the price $C(s)$ is the fee that a service requester has to pay for invoking service S.

Definition 2-5 Execution duration: Given a service S, the execution duration $D(s)$ measures the expected delay between the moment when a request is sent and the moment when the results are delivered. The execution duration is computed using the expression $D(s)=T_{process}(s)+T_{trans}(s)$, meaning that the execution duration is the sum of the processing time $T_{process}(s)$ and the transmission time $T_{trans}(s)$. The transmission time is estimated based on past executions of the service, i.e., $T_{trans}(s)=\frac{\sum_{i=1}^{n} T_i(s)}{n}$, where $T_i(s)$ is one of the past transmission times, and n is the number of executions of this service observed in the past. S. was executed in history.

Definition 2-6 Reliability: The reliability $Q(s)$ of a service s is the probability that a request is correctly responded within the maximum expected time frame indicated in the web service description. Reliability is a measure related to the successful execution of a service. The value of the reliability is computed from data of the past invocations using the expression $Q(s)=\frac{N_{successful}(s)}{N(s)}$, where $N_{successful}(s)$ is the number of times that the service s has been successfully delivered within the maximum expected time frame, and $N(s)$ is the total number of invocations of the service.

Definition 2-7 Availability: The availability $A(s)$ of a service is the probability that the service is accessible. The value of the availability of a service s is computed using the following expression $A(s) = T_a(s) / T(s)$, where $T_a(s)$ is the total amount of time when service S is available during the total $T(s)$ amount of operation period.

Definition 2-8 Reputation: The reputation $R(s)$ of a service S is a measure of its trustworthiness. It mainly depends on the comments from users' experiences of using this service. Different users may have different opinions on the same service. The value of the reputation is defined as the average rank given to the service by n users: $\frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the users' ranking on a service's reputation, n is the number of times the service has been graded.(derived from (Zeng et.al., 2004)).

Definition 2-9 Quality of a service: The quality a service S is defined by the following expression:

$$q(s) = (C(s), D(s), Q(s), A(s), R(s))$$

The method for computing the value of the quality criteria is not unique. Other computational methods can be designed to fit the needs of specific applications. The service selection approaches presented in the following sections are independent of these computational methods. Table 2-2 provides one way of computing the QoS of a compound service (Zeng et.al., 2004).

Table 2-2 Computing the QoS of a compound service S($S_1, S_2, \ldots, S_n$))

Criteria	Equation
Price	$C(s) = \sum_{i=1}^{n} C(s_i)$
Duration	$D(s) = C_{\max}(\mathrm{S})$
Reliability	$Q(s) = \prod_{i=1}^{N} e^{R(s_i)}$
Availability	$A(s) = \prod_{i=1}^{N} e^{A(s_i)}$
Reputation	$R(s) = \sum_{i=1}^{n} \frac{1}{n} R(s_i)$

In Table 2-2, $C(s_i)$ is the price of service S_i. $i = 1, 2, \ldots, n$. $C_{max}(s)$ is the makespan of the service complex. $Q(s_i)$ is the reliability of service S_i, $i = 1, 2, \ldots, n$. $A(s_i)$ is the availability of service S_i, $i = 1, 2, \ldots, n$. $R(s_i)$ is the reputation of service S_i, $i = 1, 2, \ldots, n$.

Let us assume that preferences upon a concept form a totally ordered set. In most of the cases, it is difficult to find a plan which can satisfy all the requirements. In some research work (Raman et.al., 2002), a global utility function is either implicitly or explicitly assumed to implement the trade-off between different preferences. The utility-based approaches may not be convenient for users. For example, it can be difficult for a user to determine the weight that can be assigned to cost and performance. The theory of Pareto dominance can be used to select relatively better plans from feasible plans.

Definition 2-10 Pareto dominance: Let A and B be the two candidate plans for a planning problem, and ($p_1, p_1, \ldots, p_n$) consists of a set of preference criteria. Let $p_i(A), and\ p_i(B)$ be the preference value of plan A and plan B for the preference criterion p_i. If $p_i(A) \geq p_i(B)$ for any i, the plan A Pareto-dominates the plan B.

2.4 User Preferences and Performance Metrics

User preferences play an important role in business activities, and they are recently getting more and more attentions in the web service research field, especially in personalized e-services. In business environments, the number of web services has been growing continuously, and many similar compound services can be constructed based on the existing services with differences in terms of functionality, culture, QoS and Data, as Li et.al. proposed in (Li et.al., 2008).

Functional differences: Similar services may have different functions in detail. For example, two travel services, (book an air ticket, reserve a hotel room, rent a car) and (book an air ticket, reserve a hotel room) both provide the function of trip planning, however, the first service has one more sub-function (rent a car) compared to the second service.

Cultural differences: Similar services may refer to different cultural actions. For example, one service requires users to pay before delivery while another service charges after delivery. Apparently, these two services need different preconditions, thus they can be applied to different users.

QoS differences: Similar services may deliver different QoS features. For example, a service using one algorithm may be more efficient than a service using another algorithm; however the former may have less privacy than the latter.

Data differences: Similar services' preconditions may be different in terms of some data constraints. For example, a specific travel service can only serve those users with certain destinations and transportation options.

Definition 2-11 Density of satisfaction:

$DP = N_v / N_t$, where N_v represents the total number of user preference vectors that are satisfied, and N_t represents the total number of preference vectors.

Definition 2-12 Satisfaction degree:

$SD = 1 - \frac{\sum_{i=1}^{n} rd(v_{Ai}) - \sum_{i=1}^{n} rd_{min}(v_{Ai})}{\sum_{i=1}^{n} rd_{max}(v_{Ai})}$, where $\sum_{i=1}^{n} rd_{max}(v_{Ai})$ and $\sum_{i=1}^{n} rd_{min}(v_{Ai})$ respectively represent the Max and Min of relaxation degree of the candidate plans, and $\sum_{i=1}^{n} rd(v_{Ai})$ represents the relaxation degree of the selected plan.

Chapter 3. Service Composition: General Literature Review

In this chapter a review of service composition techniques is undertaken. Popular planning systems and software in literature are documented and critically compared.

3.1 Techniques

The two main categories of service composition methods are: 1) Heuristic methods, and 2) Mathematical Programming methods. Chan et al. (Chan et.al., 2007) discussed AI planning methods under classical planning, Neoclassical planning, Heuristic Control and AI Planning under uncertainty categories. In the Mathematical programming methods, Integer Programming and Dynamic Programming are the main ones. Within each sub-category of these composition methods, there are smaller classes of methods for service composition planning.

As for mathematical programming approaches, in 1999, Vossen et al. (Vossen et.al., 1999 and 2000) and Kautz (Kautz, 1999) initiated the ILP-based approach to AI(Artificial Intelligence)-Planning problems. Zeng et al. (Zeng et.al., 2004) reported a multiple criteria Linear Integer programming model that can be used for a general case of Web Service Composition. Gao et al. (Gao et al., 2005) model service composition by using Integer Programming, which provides a way for capacity planning and load control to be addressed by the service provider. In 2008, Rao et al. (Rao et al., 2008) applied Linear Logic theorem proving to the Web Service Composition problem. This approach considered not only functional attributes but also non-functional ones in composing web

services, which is the same objective as our integer linear programming based methodology. This approach assumes that core services are already selected by the user; however, their functionality does not completely match the user's requirement. Therefore, the approach allows partial decision maker involvement in the solution procedure.

Heuristic methods are popular in the current planners, as they can find an acceptable solution in a large decision domain in a timely manner. In this chapter, the Semantic Matching Algorithm and Semantic Casual Link Matrices will be described in detail since they consider the semantic issue in planning, which is an important issue in web service composition. The mathematical programming methods find the optimal plans for customers' queries. Some mathematical programming methods only consider the Quality of Service (QoS) issue in the domain consisting of the services with the same functions; some other mathematical programming methods consider both functional and QoS issues for heterogeneous services.

In addition, miscellaneous service planning works were produced in the past decade. In 2003, Sirin et al. (Sirin et al., 2003) proposed a prototype version of a semi-automatic method for web service composition. Their method provides possible web services to users at each step of the composition through matching web services based on functional properties as well as filtering out based on non-functional attributes. In this method, users are involved in the composition process. In 2004, Xiao et al. (Xiao et al., 2004) proposed an automatic mapping framework based on XML Document Type Definition structure. In 2005, Huang et al. (Huang et al., 2005) proposed a progressive

auction based market mechanism for resource allocation that allows users to differentiate service value and ensure resource acquisition latency. Allocation rule, pricing rule, bidding strategy were explored in this work. Hwang et al. (Hwang et al., 2008) proposed a dynamic web service selection model to determine a subset of web services to be invoked at runtime so as to successfully compose the required web service. Pacific et al. (Pacific et al., 2005) presented an architecture and prototype implementation of performance management of cluster-based web services which can allocate server resources dynamically. Oh et al. (Oh et al., 2006 and 2007) present AI planning-based frameworks which enable automatic composition of web services, and developed a novel web service benchmarking tool. In 2007, Zhang et al. (Zhang et al., 2007) proposed an architectural framework which enables technology for business service analyzers to compose and adapt heterogeneous services by pattern identification. Oh et al. (Oh et al., 2008) applied large-scale network techniques in web service analysis. Montagut et al. (Mongtagut et al., 2008) addressed the security features on executing distributed web services. Lufei et al. (Lufei et al., 2008) proposed an adaptive secure access mechanism as well as an adaptive function invocation module in mobile environments. Phan et al. (Phan et al., 2008) discussed a similarity-based SOAP multicast protocol to reduce bandwidth and latency in web services with high-volume transactions. The secure information handling in web service platform was studied by Wei et al. (Wei et al., 2008).

Table 3-1 The classification of web service composition techniques

Class	Sub-class	Detailed Sub-class
Heuristic Methods	1. Deterministic Planning methods	1. State-space based planning 2. Plan-space based planning 3. Graph based planning 4. Propositional satisfiability planning 5. Constraint Satisfaction planning 6. Domain-Independent Heuristic control planning 7. Rule based control planning 8. Hierarchical Task Network planning 9. Situation/event Calculus planning 10. Logic based planning 11. Temporal planning 12. Planning with Resources 13. Model Checking based planning 14. Case-based Planning 15. Agent-based planning 16. Plan merging and rewriting 17. Abstraction Hierarchies 18. Semantic Matching Algorithm 19. Semantic Casual Link Matrices
	2. Planning under uncertainty	1. Case-based Planning with uncertainty 2. Domain Analysis in time 3. Planning based on Markov Decision Processes (Also see Dynamic Mathematical Programming) 4. Multi-agent based planning
Mathematical Programming	1. Integer programming (Multi-criteria or Single criterion)	1. Single Objective 2. Multi Objective Programming 3. Multi Objective Goal programming
	2. Markov Dynamic programming	4. Classical Dynamic Programming 5. Nested Dynamic Programming 6. Discounted Dynamic Programming

The advantages of heuristic planning lies in that (1) the methods are usually simple and easy to be implemented; (2) they can generate a solution quickly if the

parameters can properly be adjusted. The disadvantage of the heuristic methods is that they cannot guarantee an optimal solution.

The advantages of mathematical programming models are that the models can guarantee the optimality of the solution. The disadvantage of mathematical programming model is that the computational complexity is high. Many optimization problems are NP-hard.

Service composition merely using QoS filtering has high computational complexity when there are a large number of feasible plans, since it is needs an exhaustive search before drawing a conclusion. Moreover, when there is more than one QoS criterion in the planning problem and there is no feasible plan dominating all the other plans, it is difficult to determine which plan is the best for the user. In order to alleviate this problem, a network mining based approach is explored in the dissertation.

Finally, in the current techniques, concepts and services are not treated equally important in service composition problem. Although there are graph based planning method, the idea of concept and service network does not appear. In this dissertation, the concept service (CS) network is firstly introduced to solve service composition problems.

Chan *et.al.* (2005) compared AI planning methods upon domain, searching, scalability, extendibility, concurrency and *et.al*. We extend it and compare the web service planning methods in Table 3-2.

Table 3-2 Techniques of service planning

(Legend: N : No; Y: Yes; SA : Semi-automatic; A : Automatic; NM : Not mentioned; W: Weak; G : Good.)

Properties Criteria	**Markov Decision process**	**Multi-criteria Integer programming**	**Domain independent heuristic**	**HTN Planning**	**Situation Calculus**	**Planning as Model Checking**
Search Mechanism	Policy guided	Simplex method	Heuristic guided	Forward search	Situation based	Breadth-first search from the goal state toward the initial state
Domain independence	Domain-specific	Domain-independent	Domain independent	Domain-specific or domain-configurable	Domain independent	Domain-specific
Domain Complexity	$O(\|S\|^2)$ S is the state space.	O(2^{n*L}) n is the number of services, L is the number of steps.	No proper measurement exist	O$\left(\left(\frac{n}{k}\right)!\,k\right)$	(1) Linear time in plan generation; (2) Polynomial time in combining actions	O(n!) n is the number of services
Scalability	Lack of scalability	Lack of Scalability	Not feasible	G	Reasonable	Not good
Extendibility	Y	Y	NM	Y	NM	Yes (consider the context of execution).
Applicability	Translate the policy into workflow in BPEL4-WS	Obtain semantic information from OWL	Mapping OWL-S to PDDL	Translate OWL-S into planning operators	Translate OWL-S to PDDL to situation calculus	translate BPEL4WS to planning operators
Composition constructs	Sequential, iteration and conditional	NM	Sequential and parallel	Sequential, parallel and conditional	Sequential, iteration and conditional	Sequential, iteration and conditional
Nondeterminism	Y	N	NM	ND-SHOP2 supports non-determinism	Y	Y.
Concurrency	Y	Y	Y	Y	Y	Y
Plan monitoring	NM	N	NM	NM	NM	Y
Plan failure detection and recovery	NM	NM	NM	Y	NM	Y

3.2 Current Planners

Digiampietri *et.al* (2006; 2007) gave a survey on popular planners. The survey is extended utilizing new criteria in Table 3-3.

Table 3-3 Planners

(Legend: N: No; Y: Yes; SA: Semi-automatic; A: Automatic; NM: Not mentioned; W: Weak; G: Good)

Planner	Technique	Domain complexity	Extended goals	Sensing	Concurrency	Nondeterministic actions	Extra Memo
SWORD	Rule based	NA	N	NA	Y	N	Don't use standards
FF	State based	PDDL 2.1 level 1	N	N	N	N	Use standards
IPP	Graph-plan based	PDDL/ ADL without complex preconditions and goals	N	N	N	N	--
SGP	Graph-plan based	PDDL/ADL, without complex negations in preconditions	N	Y	N	Y	--
STAN4	Graph plan, State based	The STRIPS + Equality subset of PDDL	N	N	N	N	--
VHPOP	POP based	PDDL 2.1 level 1 and 3	N	N	N	N	--
BLACKBOX	SAT, Gragh-plan	PDDL/STRIPS with restrictions	N	N	N	N	--
LGP	SAT based	PDDL 2.1 levels 1,2,3	N	N	N	N	--
SHOP2	HTN based	PDDL/ADL with metrics and time	Y	Y	N	Yes in YoYo planner (extended SHOP2)	Use standards, good scalability
Golog	High level programming execution	Situation Calculus	Y	Y	NM	Y	Use standards, and offer concurrency
MIPS	Model Check.	PDDL/STRIPS +	Support CTL	N	N	N	--
MBP	Model Check	PDDL 2.1 + extensions	Y	Y	NM	Y	Use standards, offer concurrency, and good scalability
TLPlan	Temporal	ADL + metrics	NM	N	N	N	--
TALPlanner	Temporal	PDDL 2.1 (or TAL)	Y	Y	NM	NM	--
Mcllraith's	Situation calculus	PDDL	Y	Y	Y	NM	Generate templates

							based user's preference
Paolucci's	HTN	OWL-S	NM	NM	Y	Y	Replanning function available
Pistore's	Model checking	Complex protocols	NM	NM	NM	Y	Performanc e is bad due to space complexity
ASTRO	Model checking	BPEL4-WS	NM	NM	NM	Y	Interaction between users and process

SHOP2 is a well known planner. It uses first-order-logic definitions of variables, constants, functions, predicates, terms, atoms, conjuncts, and unifiers. A plan is represented as a list of operator instances. One of the shortcomings of SHOP2 is its inability to deal with non-determinism. Subsequently, ND-SHOP2 planner (Kuter, 2004), a non-deterministic version of SHOP2, was developed. Kuter (2004) found that the way ND-SHOP2 solves non-deterministic cases is not efficient if search strategies cannot cut down the search space. Finally, a novel algorithm, YoYo, is proposed to solve non-determinism under complete observability. It combines the search-control strategies in HTNs with Planning of symbolic Model-Checking.

Paolucci (Paolucci, 2003) made use of RETSINA, a planner that is based on the HTN planning paradigm. The RETSINA planner uses OWL-S descriptions. Similar to SHOP2, RETSINA interleaves planning and execution. The author claimed that by executing the information-gathering services and re-planning, unexpected situations can be handled.

Golog (Levesque, 1997) is a logic programming language based on situation calculus. It allows to execute complex actions in dynamic environments. The semantics of Golog is defined by marco-expansion using an action and a situation calculus formula that indicates the possibility of reaching the next state from the current state by executing a sequence of actions.

McIlraith et.al. (Mcllraith and Fadel, 2002; Mcllraith and Cao, 2002) extended Golog for automatic web service composition. Service composition is done via the use of general templates. These templates can be modified based on user preferences. Later, a modified version of ConGolog (Concurrent Golog) is made. And extensions were added to the ConGolog interpreter so that it has the ability to implement sensing actions at the request from external functions. Narayanan and McIlraith translated OWL-S processes to situation calculus. OWL-S processes describe the atomic and complex actions of the web service as well as the compound services. All these descriptions are translated to PDDL (Planning Domain Description Language) then to situation calculus. Golog is still responsible for finding the suitable composition plan and executing it.

Pistore et.al. (Pistore et.al., 2004; Pistore et.al., 2005; Trainotti, 2006) employed Planning with Model Checking to automatically compose web services and synthesize the monitoring of components during planning. This is able to deal with unpredictable external services. The main drawback of this approach is its performance, which makes it unusable for complex protocols which result in a complex state space.

ASTRO toolset4 (Trainotti et.al., 2005; McDermott, 2002) is a set of tools that extend existing platforms for web service design and execution with automated composition and execution monitoring. ASTRO is comprised of five components: WS-gen, WS-mon, WS-verify, WS-console and WS-animator. WS-gen is responsible for generating the automated composition. It takes a BPEL4WS specification and translates it into an intermediate file that represents a Planning Checking problem. WSYNTH takes the intermediate file and generates a plan, translating it back to BPEL4WS executable code. WS-mon is responsible for call monitor code for the composition process which will get deployed to the monitor framework. During run-time the monitor framework executes the monitors associated with a given instance of the process being executed. WS-verify is responsible for verification to ensure the properties of service to hold. WS-console uses the active BPEL console, and WS-animator plugs in Eclipse. And they provide easy interaction between the users and the planning process.

Based on Table 3-2 and Table 3-3, it can be summarized that: (1) OWL-S is most used in applying AI planning to web service composition. (2) The domain complexity of estimated-regression planning, Model Checking and MDP model limits the scalability to large-scale problems. In web service composition, the problem domain tends to be very large. Therefore, these techniques are not applicable for large web service composition. (3) There is no existent planner based on estimated-regression and situation calculus that supports an extended goal in web service composition. (4) Plan monitoring, plan failure detection and plan recovery are not supported by most planners. Automatic web service

composition should not only generate the plan automatically, but it should be able to monitor the plan during execution and perform recovery when failures occur. (5) Estimated-regression planning (Zhang et.al., 2004) is suitable for web service composition without non-deterministic actions. (6) HTN planning (enhanced HTN planning by Zhang) is good to tackle recursive and conjunctive tasks. The algorithm combines HTN methods with partial-order planning (POP) for web service composition. (6) The way ND-SHOP2 solves non-deterministic cases is not efficient if search strategies cannot cut down the search space. (7) All the heuristic planners cannot guarantee an optimal solution.

Chapter 4. Mathematical Foundation of Service Composition

Rigorous mathematical foundations for service science are still lacking. In this dissertation, for the first time in service science filed, a serious attempt to this end is made. In this chapter, the mathematical foundation of service composition is first explored. Service composition problem is described formally by taking concepts as points in topological space, and services as maps over the space to develop the mathematical theory of service composition. After problem definition, three novel methods of service composition, namely (1) Concept-Service network model (CS), (2) Multi-criteria goal programming model, and (3) Real time dynamic model, are explored.

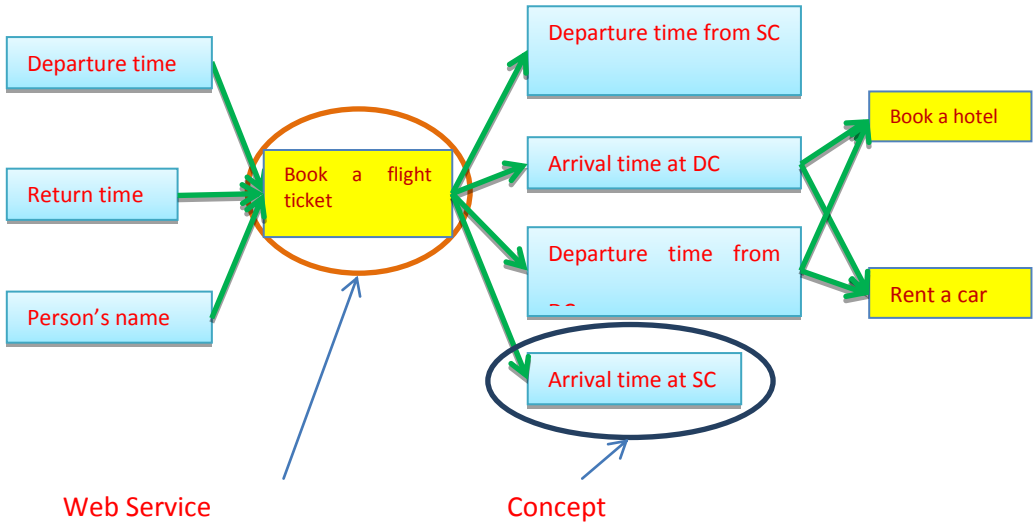


Figure 4-1 A network of web services and concepts

Two important elements of service are: (1) concepts and (2) service operation. Figure 4-1 describes an example of trip planning service. Departure time, return time and person's name are the input concepts for the service operation: book a flight ticket, and arrival and departure times of the round trip are the output concepts.

4.1 Concept Space

Let n be the number of concepts in a category. Let C denote the set of all the n concepts, and Ω denote the collection of all the subsets of C. Therefore, Ω together with empty set is a ***topological space***, satisfying the following axioms:

(1) The empty set and C are in Ω;

(2) The union of any collection of sets in Ω is also in Ω;

(3) The intersection of any finite collection of sets in Ω is also in Ω.

Each concept is a ***point*** in the topological space. The collection of concept sets, Ω is called a ***topology*** of C. The sets in Ω are ***open sets***, and their complements in C are ***closed sets***. All the concept sets in Ω are both open and closed.

Further, let $X = (x_i)_n$ be a $n \times 1$ vector, where $x_i = 0 \; or \; 1$. X can represent a combination of n concepts. A particular concept i, denoted by x_i, If $x_i = 1$, x_i is active; and if $x_i = 0$, x_i is inactive. The collection with all the combinations of n concepts is called Ω, Then $\{\Omega, \Phi\}$ is which is the power set, is the topological space. The cardinality of Ω is 2^n. Further vector X can be normalized as $\hat{X} = \frac{X}{\|X\|_1}$, so that

$\widehat{x_i} \in [0,1] and\ \sum_{i=1}^{n} \widehat{x_i} = 1$. Define semi-vector space including zero vector as ${R_+}^n \subset R^n$. The space of concept vectors, ${R_+}^n$, needs to satisfy the following seven axioms to be a semi-vector space:

(1) Associativity of addition: $X_1 + (X_2 + X_3) = (X_1 + X_2) + X_3,\ \forall X_1, X_2, X_3 \in {R_+}^n$

(2) Commutativity of addition: $X_1 + X_2 = X_2 + X_1, \forall X_1, X_2 \in {R_+}^n$

(3) Identity element of addition: let $\Theta = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, X + \Theta = X, \forall\, X \in {R_+}^n$

(4) Identity element of scalar: $1\ X = X,\ \forall\, X \in {R_+}^n$

(5) Distributivity 1 of scalar multiplication: $\alpha(X_1 + X_2) = \alpha X_1 + \alpha X_2$, $\forall X_1, X_2 \in {R_+}^n, \alpha, \beta \geq 0$

(6) Distributivity 2 of scalar multiplication: $(\alpha + \beta) = \alpha X + \beta X, \forall X \in {R_+}^n$, $\alpha, \beta \geq 0$

(7) Compatibility of scalar multiplication: $\alpha(\beta)\mathrm{X} = (\alpha\beta)X, \forall X \in {R_+}^n, \alpha, \beta \geq 0$

In this context all the seven axioms are satisfied, so, the concept vector space meets the definition of semi-vector space.

After a concept vector is defined, let us further define services as operators over concept vectors in ${R_+}^n$.

Let m be the number of services in the domain. Define ***service operator*** $f_i, i = 1, \dots, m$ mapping from ${R_+}^n$ to ${R_+}^n$, e.g. $f_i: {R_+}^n \to {R_+}^n, i = 1, \dots, m$.

By definition, the service: book a flight ticket in Figure 4-2 can be treated as mapping from a vector of concepts to another vector of concepts.

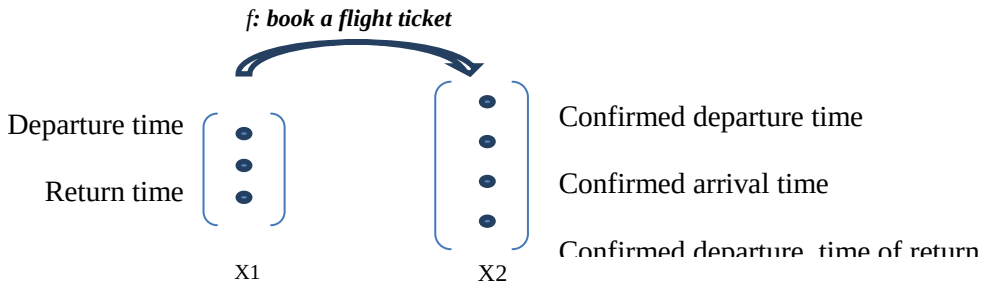


Figure 4-2 Mathematical representation of mapping over concept space

In the environment of service composition, only the Boolean status (active or inactive) of a concept is considered, e.g., whether the element corresponding to the concept is greater than zero or not after normalization. In another word, we only use the sign of each element instead of the quantity of them in a concept vector.

Definition 4-1 $\forall X_1, X_2 \in {R_+}^n,\ X_1 \geq X_2$, if and only if, the following condition holds:

For $\forall\ i$, $X_{2i} > 0 \Rightarrow X_{1i} > 0$, X_{1i}, and X_{2i} are the i^{th} element in X_1, and X_2 respectively.

For example: $\begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix} > \begin{pmatrix}1\\0\\0\\1\\1\end{pmatrix}$, where $n = 5$

For each service f_i, there are two concept combination vectors $X_i, X_i' \in {R_+}^n$, $s.t.$ $f_i(X_i) = X_i'$. So a service f_i can be defined as

$$f_i(X) = \begin{cases} X_i' + X & if\ X \geq X_i \\ X & if\ X < X_i \end{cases}, \text{ for } \forall X \in {R_+}^n \qquad 4\text{-}1$$

For example, if $f\begin{pmatrix}1\\0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}1\\0\\1\\0\\1\end{pmatrix}$, then $f\begin{pmatrix}1\\1\\1\\1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\1\\0\\1\end{pmatrix} + \begin{pmatrix}1\\1\\1\\1\\0\end{pmatrix} = \begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix}$, and

$$f\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}$$

Definition 4-2 Potential of a concept vector X:

For $\forall\ X \in B^n$,

$$\Psi(X) = \underbrace{\sum_{i=1}^{m} f_i(\sum_{j=1}^{m} f_j(\dots \sum_{l=1}^{m} f_l(X)))}_{m} \qquad 4\text{-}2$$

is called the ***potential of the concept vector*** X.

For example, if $n = 5, m = 2$, and there are two services f_1, f_2, for a concept vector $X \in \Omega$, $\Psi(X) = f_1\big(f_1(X) + f_2(X)\big) + f_2\big(f_1(X) + f_2(X)\big)$.

Definition 4-3 Spanning vector: If $\Psi(X) = I$, where $I = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$, e.g., X has the potential of I, X is called a ***spanning vector*** of Ω.

Obviously, $I = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ satisfies $\Psi(X) = I$. So I is a the largest spanning vector of Ω. In general, we are only interested in these spanning vectors $X < I$.

Definition 4-4 Minimum spanning vector: If $\Psi(X^*) = I$, and $\nexists\, X < X^* \in \Omega$, and $\Psi(X) = I$, X^* is a ***minimum spanning vector*** of Ω.

There may be more than one minimum spanning vectors of Ω, for example, both $X_1^* \; and \; X_2^*$ ($X_1^* \neq X_2^*$) can be minimum spanning vectors of Ω, e.g., $X_1, X_2 \in \Omega$, $\Psi(X_1) = I$, $\Psi(X_2) = I$, and $X_1 \not\leq X_2 \; and \; X_2 \not\leq X_1$.

Definition 4-5 ***Service composition***: For a query with a given input vector X_0 and a needed output vector X_1, noted as $q(X_0, X_1)$, find a set of services and the operating sequence among them, noted as $H(F)$, so that $H(F)(X_0) \geq X_1$, where F is a collection of all the services. $H(F)$ represents a selection among all the services and the operating sequence among these selected.

Theorem 4-1 Existence of a service composition solution

For a query with a given input vector X_0 and a needed output vector X_1, noted as $q(X_0, X_1)$, if $\Psi(X_0) \geq X_1$, there is at least one solution of service composition to $q(X_0, X_1)$ in Ω.

Proof:

(1) A service composition exists for the query $q(X_0, X_1)$, e.g., $\exists\ H(F)$, s.t. $H(F)(X_0) \geq X_1$. Since $\Psi(X_0)$ is the combination of all the services, and $H(F)(X_0)$ is a combination of some services, $\Psi(X_0) \geq\ H(F)(X_0) \geq X_1$;

(2) $\Psi(X_0) \geq X_1$. This tells us that at least one service composition for the query $q(X_0, X_1)$ exists, which is the combination of all the services.

Now, let us study another extreme case.

Definition 4-6 minimum spanning vector: If $\Theta = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$ satisfies $\Psi(X) = I,\ \Theta$ is the minimum spanning vector of Ω, since for $\forall\ X \in \Omega,\ \Theta \leq$ X.

Thus, there must be at least one service f^* which does not need any input concept to generate its output concept(s), this service f^* can be called as a ***creator*** of Ω. If a space has a creator, the system defined via Ω and its service operator family $F = \{f_1, f_2, \ldots, f_m\}$ is ***self-sufficient***. However, f^* may or may not exist in reality.

Corollary 4-1 If a system is self-sufficient, there is a least one solution for any query to the system.

Next let us define addition of services.

Definition 4-7 *Addition of services*: Since services $f_i$and f_j needs some concept vector to process, let us define service addition as follows:

$$\forall X \in \Omega,\ \left(f_i + f_j\right)(X) = f_i(X) + f_j(X) \quad \text{4-3}$$

Definition 4-8 *Nestedness (Multiplication) of services* $f_i . f_j$:

$$\forall X \in \Omega, \left(f_i . f_j\right)(X) = f_i(f_j(X)) \quad \text{4-4}$$

Theorem 4-2 Addition of Service Operators

(1) Associativity of addition: for $\forall X \in \Omega, \left[\left(f_i + f_j\right) + f_k\right](X) = [f_i + (f_j + f_k)](X)$;

(2) Commutativity of addition: for $\forall X \in \Omega, \left(f_i + f_j\right)(X) = (f_j + f_i)(X)$

(3) Identity element of addition: There exists an element of Θ, e.g. Zero service, such that $(f + \Theta)(\mathrm{X}) = f(X), \forall X \in \Omega$.

Theorem 4-3 Multiplication of Service Operators

(1) Associativity of multiplication: for $\forall X \in \Omega, \left(f_i . f_j\right) . f_k(X) = f_i . (f_j . f_k)(X)$; 4-5

(2) No commutativity of multiplication: for $\forall X \in \Omega, \left(f_i . f_j\right)(X) \neq (f_j . f_i)(X)$ 4-6

(3) Scalability: for $\forall X \in \Omega, (\alpha f_i)(X) = \alpha f_i(X) = f_i(\alpha X)$, for $\forall \alpha > 0$ 4-7

Let $\mathbb{Z}$ be the collection of all the subsets of entire service set F. It is easy to prove that the union of $\mathbb{Z}$ and null set $\emptyset$, $\{ \mathbb{Z}, \emptyset \}$ also forms a topological space.

4.2 The Service Space and Mutuality/Duality theory

In the previous section, a collection of concept sets together with null set is defined as topological space and further a semi-vector space is developed. In this section, let us study the dual space — the service space, which is also a topological space and a semi-vector space.

Let m be the number of services in a category. let us denote the set of all the m services as F, the collection of all the subsets of F as $\mathbb{Z}$. $\mathbb{Z}$ together with empty is a ***topological space***, satisfying the following axioms:

(1) The empty set and F are in $\mathbb{Z}$;

(2) The union of any collection of sets in $\mathbb{Z}$ is also in $\mathbb{Z}$;

(3) The intersection of any finite collection of sets in $\mathbb{Z}$ is also in $\mathbb{Z}$

Each service is a ***point*** in the topological space. The collection of service sets, $\mathbb{Z}$ is called a ***topology*** of F. The sets in $\mathbb{Z}$ are ***open sets***, and their complements in F are ***closed sets***. All the service sets in $\mathbb{Z}$ are both open and closed.

Further, let $Y = (y_i)_n$ be a $n \times 1$ vector, where $y_i = 0 \; or \; 1$. Y can represent a combination of m services. A particular service i, denoted by y_i, if $y_i = 1$, y_i is active/usable; and if $y_i = 0$, y_i is inactive/unusable. The collection with all the combinations of m services is $\mathbb{Z}$, then the power set $\{\mathbb{Z}, \Phi\}$ is the topological space. The cardinality of $\mathbb{Z}$ is 2^m. Further vector Y can be normalized via $\hat{Y} = \frac{Y}{\|Y\|_1}$, so that $\hat{y_i} \in [0,1]$ and $\sum_{i=1}^{n} \hat{y_i} = 1$. Define semi-vector space including zero vector as ${R_+}^m \subset R^m$. It is proven in the previous section that the space of concept vectors, ${R_+}^m$, satisfies the following seven axioms and is a semi-vector space:

(1) Associativity of addition: $Y_1 + (Y_2 + Y_3) = (Y_1 + Y_2) + Y_3, \forall \, Y_1, Y_2, Y_3 \in {R_+}^m$

(2) Commutativity of addition: $Y_1 + Y_2 = Y_2 + Y_1, \forall \, Y_1, Y_2 \in {R_+}^m$

(3) Identity element of addition: let $\Theta=\begin{pmatrix}0\\ \vdots \\ 0\end{pmatrix}$, $Y+\Theta=Y, \forall\, Y \in {R_+}^m$

(4) Identity element of scalar: $1\,Y=Y,\ \forall\, Y \in {R_+}^m$

(5) Distributivity 1 of scalar multiplication: $\alpha(Y_1+Y_2)=\alpha Y_1+\alpha Y_2$, $\forall Y_1, Y_2 \in {R_+}^m, \alpha, \beta \geq 0$

(6) Distributivity 2 of scalar multiplication: $(\alpha+\beta)=\alpha Y+\beta Y, \forall\, Y \in {R_+}^m$, $\alpha, \beta \geq 0$

(7) Compatibility of scalar multiplication: $\alpha(\beta)\mathrm{Y}=(\alpha\beta)Y, \forall Y \in {R_+}^m, \alpha, \beta \geq 0$

In this context all the seven axioms are satisfied, so, the service space meets the definition of semi-vector space.

After a service combination set is defined as a service vector, the concepts can be further defined as operators over service vectors in ${R_+}^m$.

Let m be the number of services in the domain. Define ***concept operator*** $g_i, i=1, \ldots, n$, mapping from ${R_+}^m$ to ${R_+}^m$, e.g. $g_i: {R_+}^m \to {R_+}^m$, $i=1, \ldots, n$.

A service vector Y represents which a set of services. These elements with $y_i > 0$ means that service i can be processed.

By definition, the concept set $g_i = \{$person' s name, departure time, return time$\}$ in Figure 4-3 can be treated as mapping from a vector of services to another vector of services. This concept set can be used by services {book a flight, book a hotel} at the

beginning, after using these services, more concepts will be generated {arrival time at the venue, departure time from the venue}, and a more broad set of services can be used, for example {book a flight, book a hotel, schedule pick up, schedule sightseeing}

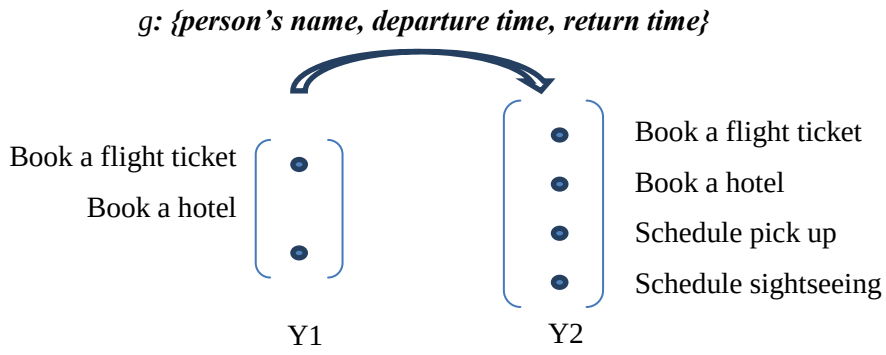


Figure 4-3 Mathematical representation of mapping over service space

In the environment of service composition, only the Boolean property (activated or deactivated) of a service is considered, e.g., whether the element corresponding to the service is greater than zero or not after normalization. In another word, we only care about the sign of each element instead of the quantity of them in a service vector, which is the same as what we cared about concept vector in the previous section.

Definition 4-9 $\forall\, Y_1, Y_2 \in {R_+}^m,\ Y_1 \geq Y_2$, if and only if, the following condition holds:

For $\forall\ i$, $Y_{2i} > 0 \Rightarrow Y_{1i} > 0$, Y_{1i}, and Y_{2i} are the i^{th} element in Y_1, and Y_2 respectively.

For example: $\begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix} > \begin{pmatrix}1\\0\\0\\1\\1\end{pmatrix}$, where $m = 5$

For each an operator of concept set g_i, there are two service vectors $Y_i, Y_i' \in {R_+}^m$, $s.t.\ g_i(Y_i) = Y_i'$. So a ***concept operator*** g_i can be defined as

$$g_i(Y) = \begin{cases} Y_i' & if\ Y \geq Y_i \\ g(Y) & if\ Y < Y_i \end{cases}, \text{ for } \forall Y \in {R_+}^m \qquad 4\text{-}8$$

For example, if $g\begin{pmatrix}1\\0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}1\\0\\1\\0\\1\end{pmatrix}$, then $g\begin{pmatrix}1\\1\\1\\1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\1\\0\\1\end{pmatrix}$, and $g\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}$ cannot be simply decided here.

Definition 4-10 *Addition of concept operators*: Since concept operators $g_i$and g_j needs some service vector to be processed, let us define the addition of concept operators: for $\forall\ Y \in \mathbb{Z},\ \left(g_i + g_j\right)(Y) = g_i(Y) + g_j(Y)$

Definition 4-11 *Nestedness (Multiplication) of concept operators* $g_i.g_j$:

$$\forall\ Y \in \mathbb{Z}, \left(g_i.g_j\right)(Y) = g_i(g_j(Y)) \qquad 4\text{-}9$$

Theorem 4-4 Addition of concept operators

(1) *Associativity* of addition: for $\forall\ Y \in \mathbb{Z}$,

$$\left[\left(g_i + g_j\right) + g_k\right](Y) = [g_i + (g_j + g_k)](X); \qquad 4\text{-}10$$

(2) *Commutativity* of addition: for $\forall\ Y \in \mathbb{Z}$,

$$(g_i + g_j)(X) = (g_j + g_i)(Y) \quad 4\text{-}11$$

(3) *Identity element* of addition: There exists an element of Θ, called Zero concept operator such that

$$(g + \Theta)(\mathrm{Y}) = g(Y), \forall\, Y \in \mathbb{Z} \quad 4\text{-}12$$

Theorem 4-5 Multiplication of concept operators

(1) Associativity of multiplication: for $\forall\, Y \in \mathbb{Z}$, $(g_i . g_j)$.

$$g_k(X) = g_i . (g_j . g_k)(X); \quad 4\text{-}13$$

(2) No commutativity of multiplication: for $\forall\, Y \in \mathbb{Z}$,

$$(g_i . g_j)(Y) \neq (g_j . g_i)(Y) \quad 4\text{-}14$$

(3) Scalability: for $\forall\, Y \in \mathbb{Z}$,

$$(\alpha g_i)(Y) = \alpha g_i(Y) = g_i(\alpha Y), \text{ for } \forall\, \alpha > 0 \quad 4\text{-}15$$

Theorem 4-6 Mutuality /Duality of concept space and service space

$\forall\, f \in {R_+}^m, g \in {R_+}^n,$ both f and g are vectors, and operators at the same time. f is a service vector and service operator; and g is a concept vector and concept operator.

(1) $f(g)$ will generate a set of new concepts, denoted as the vector as g'.

(2) $g(f)$ will generate a set of new services f' that can use g'.

So, it is possible to have a media in order to let concept vectors and service vectors working together. This will form the foundation of service computing.

In this chapter, a mathematical foundation for service composition is established to map the real word problem with topological space problems. Theorems on space

definition, operators, existence of solutions and convergence are given. In the next chapter, a concept service (CS) network matrix is introduced and various of service composition algorithms are explored.

Chapter 5. Concept Service (CS) Network Matrix and Service Composition

Based on the mathematical foundation in Chapter 4, let us explore how to fuse concept operators and service operators together based on the mutuality/duality theory of concept space and service space. A concept-service (CS) network matrix is formulated in this dissertation. A set of service composition methods based on CS network matrix is developed. Before CS network matrix and the service composition algorithms based on it are discussed in detail, a literature of Causal Link Matrices (CLM) is undertaken.

5.1 Related Literature: Semantic Casual Link Matrices

Lecue and Leger (Lecue et.al., 2006) developed the Causal Link Matrices (CLM) for web service composition in 2006. The method of CLM was proposed as a composition model for a finite set of web services. The method pre-calculates and stores all semantic connections in a matrix, which make the later composition easier and faster. Moreover, this method has good scalability because adding or deleting a service can easily be treated as adding or deleting a column in the causal link matrix. The CLM method is based on functional level composition of web services.

Input and output parameters of web services are concepts referred to ontology. Finding a semantic similarity between two parameters is similar to finding matching between two concepts. The connection between two services is measured by the matching score between an input parameter of one service and an output parameter of

another service. The CLMs contribute to the process of web service composition by describing web services according to the dependency of the semantic links. A causal link is defined as a triple $< s_1, Link\ score(S_1, S_2), s_2 >$, where S_1 and S_2 are two web services in a set of available web services S_{Ws}. The concept set Out_{S_1} consists of output parameters of the service S_1 and the concept set In_{S_2} consists of input parameters of the service S_2. The function $Link\ score$ is the function of semantic similarity described in Section 2.1 between the concept sets Out_{S_1} and In_{S_2} . A causal link $< s_1, Link\ score(Out_{S_1}, In_{S_2}), s_2 >$ indicates that

i) S_1 precedes S_2, if $Link\ score(Out_{S_1}, In_{S_2}) > 0$ and no web service is between S_1 and S_2;

ii) S_1 does not precedes S_2, if $Link\ score(Out_{S_1}, In_{S_2}) = 0$

$$Link\ score(S_1, S_2) = \sum_{all\ concept\ pairs} match\ (C_{Out_{S_1}} C_{In_{S_2}})$$

Where $match\ (C_{Out_{S_1}} C_{In_{S_2}})$ is the similarity score of concepts $C_{Out_{S_1}}$ and $C_{In_{S_2}}$.

Definition 5-1 *Valid Causal Link* (Lecue et.al., 2006) :

A causal link $< s_1, Link\ score(Out_{S_1}, In_{S_2}), s_2 >$ is valid, if

$Link\ score(Out_{S_1}, In_{S_2}) >$ 0.

A causal link matrix contains all valid links because causal links help to detect semantic link between web services. Indeed all valid causal links between Web services are explicitly represented with a value pre-computed by the $Link\ score()$ function. The

causal link matrix aims at storing all those valid causal links in advance. The more valid causal links there are, the more possibility it generates a functional composition problem.

Definition 5-2 *Causal Link Matrix (CLM)*

The set of $p \times q$ CLMs is defined as $M_{(p\times q)}$. $r_{i,i\in\{1,\dots,p\}}$ are labelled by I$nput(S_{Ws})$, the inputs parameters of services in S_{Ws}. Columns $c_{j,j\in\{1,\dots,q\}}$are labelled by $(Output(S_{Ws}) \cup \beta) \subseteq \mathcal{T}$, where $Output(S_{Ws})$ is the set of output parameters. $\beta \subseteq \mathcal{T}$ is the set of concepts of requested output . Each entry $m_{i,j}$ of a CLM $\mathcal{M}$atrix is defined as a set of pairs $(sy, Link\ score\left(Out_{S_y}, c_j\right))$ such that $r_i \in \mathcal{T} \cap In(s_y) \subseteq Input(S_{Ws})$ is the i^{th} row, and $c_j \in \mathcal{T} \cap (Input(S_{Ws}) \cup \beta)$ is the j^{th} column. Here, $Out(s_y)$ is the set of output parameters of the web services s_y whereas $In(s_y)$ is its set of input parameters. β contains the set of goals, described as concepts in a terminology $\mathcal{T}$. Those concepts have to be reached in order to meet the goal. The variable *Link score* refers to the degree of match. Once all web services are semantically chained according to the causal link criteria, the web service composition problem is mapped to a network problem.

The key contribution of the causal link matrix is to form a semantic network model relevant for a web service composition. This model allows performance analysis of compound services with a concrete view of causal links and their implicit semantic dependency. The causal link matrix aims to compute web services according to a semantic similarity between their Output/Input concepts. Thus the CLM stores all possible interactions between all the known web services. Furthermore, the casual link

matrix allows to dynamically plan in real scenarios, since insertion and deletion of web services is easy. The causal link matrix is able to prepare a suitable context for further planning with the purpose of obtaining complete, correct, consistent and optimal plan in terms of performance of the compound services.

The CLM checks all web service pairs with semantic connections in a domain. The causal link score allows the early detection of infeasible, feasible and candidate links between web services. In case of more complex composition, more than one web service needs to be chained with a service in order to produce input parameters of this service. This can be realized through parallel regression search or by the application of further mathematical programming. These techniques are acceptable in case the entry degree in the CLM is greater than 1.

Consistency is a necessary condition for a solution plan. A CLM contains explicitly all valid causal links between web services. A plan refinement algorithm is necessary afterwards to guarantee a correct, consistent solution. The plan refinement follows a backward chaining strategy from a goal to initial state. The solution plan set from CLM can be larger than what we need, so a post process is necessary to find an optimal, consistent, correct and complete plan at the last step.

The flexibility of web service composition models is a criterion for evaluating the methods. Models need to be as robust as possible in order to evolve in a volatile environment, in terms of the alteration and modification in the environment, such as addition, deletion, and the upgrading of a web service. In CLMs, a modification is

supported by a causal link matrix revision in an element, in a row or in a column. So the updated systems are partially supported by the previous model of composition in CLMs. The flexibility of the model allows the users to apply a dynamic process of web service discovery.

The drawback of Lecue and Leger's CLM algorithm is that it loses the potential to do further operations on the matrix. The reason is that: (1) The element of the matrix consists of both services and score; (2) The matrix element contains implicitly more than one web service related to the row. All these features add difficulty in utilizing the matrix fully and efficiently.

5.2 Concept Service (CS) Network Matrix

In this section, concept service (CS) network matrix is introduced based on the mutuality/duality theory in Chapter 4 to overcome the disadvantages of casual link matrix.

Web services and the parameters/ concepts can form a network if we take services and parameters (concepts) as nodes, and treat the information flow among concepts and services as arcs. In order to simplify the model, OR parallel is not considered at the beginning of modeling. OR parallel can be considered at the end, after network analysis. Figure 5-1 shows that web service has parameter definition of input flows and output flows.

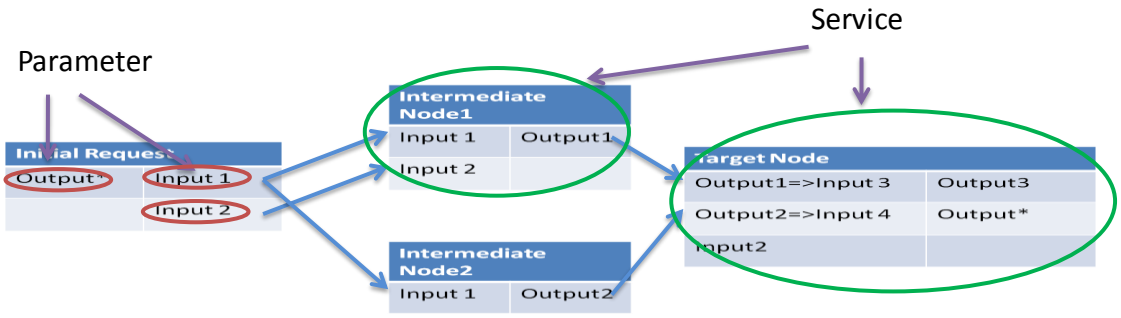


Figure 5-1 Inner structure of web services

The steps for building the Concept-Service network includes: (1) Index the services and concepts; (2) Generate the relationship among concepts and services in a matrix, let us call it Concept-Service (CS) network matrix. Let this matrix be *M*. In matrix M, (a) if a concept is one of the input concepts that a service needs, use value 1 as the corresponding element in M; (b) if a concept is one of the output concepts of a service, use value 1 as the corresponding element in M; (b) all other elements are Zero.

$$M_{(n+m)\times(n+m)} = \left[\begin{array}{ccc|ccc} M_{11} & \cdots & M_{1n} & M_{1(n+1)} & \cdots & M_{1(n+m)} \\ \cdots & \ddots & \cdots & \cdots & \ddots & \cdots \\ M_{n1} & \cdots & M_{nn} & M_{n(n+1)} & \cdots & M_{n(n+m)} \\ \hline M_{(n+1)1} & \cdots & M_{(n+1)n} & M_{(n+1)(n+1)} & \cdots & M_{(n+1)(n+m)} \\ \cdots & \ddots & \cdots & \cdots & \ddots & \cdots \\ M_{(n+m)1} & \cdots & M_{(n+m)n} & M_{(n+m)(n+1)} & \cdots & M_{(n+m)(n+m)} \end{array}\right] \quad 5\text{-}1$$

In M:

$M_{ii} = 1\,,\ i = 1,2,\ldots,(n+m)$ which means any service or concept can reach themselves.

$M_{ij} = 0\,,\ i = n+1,\ldots,n+m; j = n+1,\ldots,n+m$ which means there is no direct relation in the first matrix.

Let $C_{n\times n} = \begin{bmatrix} M_{11} & \dots & M_{1n} \\ \dots & \ddots & \dots \\ M_{n1} & \dots & M_{nn} \end{bmatrix}$ be the incidence matrix of concepts derived from OWL ontology.

Let $I_{n\times m} = \begin{bmatrix} M_{1(n+1)} & \dots & M_{1(n+m)} \\ \dots & \ddots & \dots \\ M_{n(n+1)} & \dots & M_{n(n+m)} \end{bmatrix}$ be the incidence matrix of concepts to services from WSDL.

Let $O_{m\times n} = \begin{bmatrix} M_{(n+1)1} & \dots & M_{(n+1)n} \\ \dots & \ddots & \dots \\ M_{(n+m)1} & \dots & M_{(n+m)n} \end{bmatrix}$ be the incidence matrix of services to concepts from WSDL.

Let $S_{m\times m} = \begin{bmatrix} M_{(n+1)(n+1)} & \dots & M_{(n+1)(n+m)} \\ \dots & \ddots & \dots \\ M_{(n+m)(n+1)} & \dots & M_{(n+m)(n+m)} \end{bmatrix}$ be the incidence matrix of services. Initialize, $M_{ij} = 0$, $i = m+1, \dots, m+n; j = m+1, \dots, m+n$, that is $S_{n\times n} = (0)_{n\times n}$ in *M* initially.

$$\text{Thus, } M_{(n+m)\times(n+m)} = \begin{bmatrix} C_{n\times n} & I_{n\times m} \\ O_{m\times n} & S_{m\times m} \end{bmatrix} \qquad 5\text{-}2$$

The following algorithm shows how to get each element in $C_{n\times n}$, $I_{n\times m}$, $O_{m\times n}$and $S_{m\times m}$.

Algorithm 5-1 Network generation

```
Main function MatrixGen(W,G)
    For any service in W, do
        Index it in A;
    For any concept in G, do
    Index it in A;
    Initialize ElementScore = 0;
    For any concept C in G and any service S in W do
        ElementScore =ElementCtoS(C, S);
    For any service S in W and any concept C in G do
        ElementScore =ElementStoC(S, C);
    For any two concepts C1 and C2 in G do
        ElementScore =ElementCtoC(C1, C2);
    For any two services S1 and S2 in W do
        ElementScore=0;
End

Function ElementCtoS(C, S)
    Parse W, if I is the input set of S, and C ∈ I, then
    Score =1;
End
Function ElementStoC(S, C)
    Parse W, if O is the output set of S, and C ∈ O, then
    Score =1;
End
Function ElementCtoC(C1,C2)
    Score=Score(C1,C2);
End
```

In Algorithm 5-1, W is the WSDL file, and G is the OWL file. (W, G) includes the total information of the WSDL file and OWL ontology. A is a vector. ElementScore is a temporary variable representing the utility of the relation. C is any concept in G. ElementCtoS( ) is a sub-function to compute the utility score of the link from a concept to a service. ElementStoC( ) is a sub-function to compute the utility score of the link from a

service to a concept. ElementCtoC( ) is a sub-function to compute the utility score of the link from a concept to a concept. Score(C1, C2) is the semantic score of the link C1 to C2.

In this work the semantic match score among concepts are derived based on the work of Li and Horrocks (2004). The following definition is used to calculate the score in ElementCtoC( ).

$$Score(C1, C2) = \begin{cases} 1 & if\ C1 = C2 \\ 1 & if\ C1 \subset C2 \\ 0 & if\ C1 \supset C2 \\ 0 & Otherwise \end{cases} \quad 5\text{-}3$$

Next, Concept-Service network matrix can be normalized column by column individually. $M = [M_1, M_2, \dots, M_{(n+m)(n+m)}]$ becomes $[\frac{M_1}{\|M_1\|_1}, \frac{M_2}{\|M_2\|_1}, \dots, \frac{M_{(n+m)}}{\|M_{(n+m)}\|_1}]$, where $M_1, M_2, \dots, M_{(n+m)(n+m)}$ are the column vectors in M. After normalizing the columns, furthermore, the rows can also be normalized. If the L_1norm of any row vector in M is greater than 1, I divide each elements of the row by the L_1 norm of this row; otherwise, the row is kept the same. Let us still denoted this normalized CS network matrix as M. Thus, M has the following properties:

Properties of M:

$$(1)\ \sum_{i=1}^{(m+n)} M_{ij} \le 1,\ j = 1,2,\dots,m+n; \qquad 5\text{-}4$$

$$(2)\ \sum_{j=1}^{(m+n)} M_{ij} \le 1,\ i = 1,2,\dots,m+n. \qquad 5\text{-}5$$

Definition 5-3 for any matrix A and B, if $A_{ij} > 0 \Rightarrow B_{ij} > 0$, it can be called that $A \le B$ in sign.

Corollary 5-1

(1) If $A \le B$ in sign, then $B \ge A$ in sign.

(2) If $A \le B$ in sign, and $A \ge B$ in sign, then $A = B$ in sign.

Lemma 5-1 If both A and B satisfies properties (1) and (2), AB also satisfies both properties (1) and (2).

Lemma 5-2 Property of CS network matrix series

If CS network matrix M has properties (1) and (2), the matrices $M^i\ i = 1,2,\dots$ also have properties (1) and (2).

Theorem 5-1 Convergence of CS network matrix series:

If M satisfies properties (1) and (2), $M^i, i = 1,2,3\dots$ converges to zero, e.g., $\lim_{i\to\infty} M^i = 0$.

Proof:

$$(1)\ \sum_{i=1}^{(m+n)} M_{ij} \le 1,\ j = 1,2,\dots,m+n \Longrightarrow \|M\|_1 = \max_j\left\{\sum_{i=1}^{(m+n)} M_{ij}\right\} \le 1$$

$$(2)\ \sum_{j=1}^{(m+n)} M_{ij} \le 1,\ i = 1,2,\dots,m+n \Longrightarrow \|M\|_\infty = \max_i\left\{\sum_{j=1}^{(m+n)} M_{ij}\right\} \le 1$$

Because of $\|M\|_1$ and $\|M\|_\infty$ are equivalent, let us use $\|M\|_1$ to prove the theorem of convergence.

Since $\left\|M^i\right\|_1 \leq (\|M\|_1)^i$,

$0 \leq \lim_{i\to\infty}\left\|M^i\right\|_1 \leq \lim_{i\to\infty}(\|M\|_1)^i = 0$

So, $\lim_{i\to\infty}\left\|M^i\right\|_1 = 0$

Since $M_{ij} \geq 0$, and , $\lim_{i\to\infty}\left\|M^i\right\|_1 = \max\{column\ sums\ in\ M^i\} = 0$

$$\lim_{i\to\infty} M^i = 0 \qquad 5\text{-}6$$

Next let us study the details in the CS network matrix series.

$$M^2{}_{(n+m)\times(n+m)} = \begin{bmatrix} C^2{}_{n\times n} + I_{n\times m}\mathrm{O}_{m\times n} & C_{n\times n}I_{n\times m} \\ \mathrm{O}_{m\times n}C_{n\times n} & \mathrm{O}_{m\times n}I_{n\times m} \end{bmatrix} \qquad 5\text{-}7$$

$$M^3{}_{(n+m)\times(n+m)} =$$

$$\begin{bmatrix} C^3{}_{n\times n} + I_{n\times m}\mathrm{O}_{m\times n}\mathrm{C}_{n\times n} + C_{n\times n}I_{n\times m}\mathrm{O}_{m\times n} & C^2{}_{n\times n}I_{n\times m} + I_{n\times m}\mathrm{O}_{m\times n}I_{n\times m} \\ \mathrm{O}_{m\times n}C^2{}_{n\times n} + \mathrm{O}_{m\times n}I_{n\times m}\mathrm{O}_{m\times n} & \mathrm{O}_{m\times n}C_{n\times n}I_{n\times m} \end{bmatrix} \qquad 5\text{-}8$$

Multiplying by M, the algorithm will converge due to the convergence theorem.

Let $G_t = \sum_{j=1}^{t} M^j$

G_t is the entire CS network obtained via augmenting the initial CS network M by the new networks generated at each step.

$$M^{t}{}_{(n+m)\times(n+m)}$$

$$= \begin{bmatrix} \sum_{\sum_{i=0}^{t}(k_i+2l_t)=t} C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{\mathrm{m\times n}})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} & \sum_{\sum_{i=0}^{t}(k_i+2l_t)=t-1} C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} I_{n\times m} \\ \sum_{\sum_{i=0}^{t}(k_i+2l_t)=t-1} \mathrm{O}_{m\times n}C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} & \sum_{\sum_{i=0}^{t}[2l_i+(2+k_i)g_i]=t} (\mathrm{O}_{m\times n}I_{n\times m})^{l_1}(\mathrm{O}_{m\times n}C^{k_1}{}_{n\times n}I_{n\times m})^{g_1} \dots (\mathrm{O}_{m\times n}I_{n\times m})^{l_t}(\mathrm{O}_{m\times n}C^{k_t}{}_{n\times n}I_{n\times m})^{g_t} \end{bmatrix}$$

$$t = 1,2,3, \dots$$

Where $M^{t}{}_{(n+m)\times(n+m)}$ represent the new arcs generated in the t^{th} step.

The total graph at the t^{th} step can be represented as

$$G_t = \sum_{j=1}^{t} M^{j}$$

$$= \sum_{j=1}^{t} \begin{bmatrix} \sum_{\sum_{i=0}^{t}(k_i+2l_t)=t} C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{m\times m}(I_{m\times n}\mathrm{O}_{\mathrm{n\times m}})^{\mathrm{l_t}} & \sum_{\sum_{i=0}^{t}(k_i+2l_t)=t-1} C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} I_{n\times m} \\ \sum_{\sum_{i=0}^{t}(k_i+2l_t)=t-1} \mathrm{O}_{m\times n}C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} & \sum_{\sum_{i=0}^{t}[2l_i+(2+k_i)g_i]=t} (\mathrm{O}_{m\times n}I_{n\times m})^{l_1}(\mathrm{O}_{m\times n}C^{k_1}{}_{n\times n}I_{n\times m})^{g_1} \dots (\mathrm{O}_{m\times n}I_{n\times m})^{l_t}(\mathrm{O}_{m\times n}C^{k_t}{}_{n\times n}I_{n\times m})^{g_t} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^{t}\sum_{\sum_{i=0}^{t}(k_i+2l_t)=t} C^{k_1}{}_{n\times n}(I_{n\times m}O_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} & \sum_{j=1}^{t}\sum_{\sum_{i=0}^{t}(k_i+2l_t)=t-1} C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} I_{n\times m} \\ \sum_{j=1}^{t}\sum_{\sum_{i=0}^{t}(k_i+2l_t)=t-1} \mathrm{O}_{m\times n}C^{k_1}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_1}} \dots C^{k_{t-1}}{}_{n\times n}(I_{n\times m}\mathrm{O}_{m\times n})^{\mathrm{l_t}} & \sum_{j=1}^{t}\sum_{\sum_{i=0}^{t}[2l_i+(2+k_i)g_i]=t} (\mathrm{O}_{m\times n}I_{n\times m})^{l_1}(\mathrm{O}_{m\times n}C^{k_1}{}_{n\times n}I_{n\times m})^{g_1} \dots (\mathrm{O}_{m\times n}I_{n\times m})^{l_t}(\mathrm{O}_{m\times n}C^{k_t}{}_{n\times n}I_{n\times m})^{g_t} \end{bmatrix}$$

Now, let us define the spectrum of a network.

Definition 5-4 *Spectrum of a network* is the spectrum of its incidence matrix.

Next let us study the spectrum of the networks in each step.

$$G_t = \sum_{j=1}^{t} M^j \tag{5-9}$$

Let $\lambda(M)$ be the set of Eigen values of M. For a $\lambda_1 \in \lambda(M)$, let $Mx_1 = \lambda_1 x_1$, where λ_1 is a Eigen value of matrix M, and x_1 is a Eigen vector corresponding to λ_1.

$$G_t x_1 = \sum_{j=1}^{t} M^j x_1$$

$$\therefore\ G_t x_1 = \lambda x_1 \Leftrightarrow \sum_{j=1}^{t} M^j x_1 = \lambda x_1$$

If the Eigen vector does not change, we have

$$\sum_{j=1}^{t} M^j x_1 = \sum_{j=1}^{t} M^{j-1} M x_1 = \sum_{j=1}^{t} M^{j-1} \lambda_1 x_1 = \sum_{j=1}^{t} \lambda_1 M^{j-1} x_1 = \ldots = \sum_{j=1}^{t} \lambda_1^j x_1$$

$$\sum_{j=1}^{t} \lambda_1^j x_1 = \lambda x_1,\ \text{for}\ \forall\ x_1 \neq 0 \Leftrightarrow \sum_{j=1}^{t} \lambda_1^j = \lambda$$

$$\therefore \lambda = \sum_{j=1}^{t} \lambda_1^j \tag{5-10}$$

Theorem 5-2 Network spectrum

When the network evolves with $G_t = \sum_{j=1}^{t} M^j$, the spectrum of the service network increases as $\lambda = \sum_{j=1}^{t} \lambda_1^j$.

When the spectrum increases the linking information also increases in the network which can be visualized in the Section 3.7. It is obviously that the linking

information will not decrease in G. Now, let us study when the network spectrum will not increase.

$$\sum_{j=1}^{t} M^{j} x = \lambda_1 x$$

left multiply x_1^T to the boths sides of the equation

$$x_1^T \sum_{j=1}^{t} M^{j} x = x_1^T \lambda_1 x$$

$$\sum_{j=1}^{t} x_1^T M^{j} x = x_1^T \lambda_1 x$$

$$\sum_{j=1}^{t} (x_1^T M^{j}) x = (\lambda_1 x_1)^T x$$

$$\because Mx_1 = \lambda_1 x_1$$

$$\therefore \sum_{j=1}^{t} (x_1^T M^{j}) x = (Mx_1)^T x$$

$$\sum_{j=1}^{t} (x_1^T M^{j}) x = x_1^T M^T x$$

$$x_1^T \sum_{j=1}^{t} M^{j} x = x_1^T M^T x, \text{ for } \forall x_1$$

$$\sum_{j=1}^{t} M^{j} x = M^T x, \text{ for } \forall x$$

$$\therefore \sum_{j=1}^{t} M^{j} = M^T$$

Since M is not a zero matrix (a matrix with all zero elements),

$\therefore M = I$

Next let us give a definition on the sum of the CS network matrix series.

Definition 5-5 The *potential of a Concept Service (CS) network M* is defined by

$G^*_{(m+n)\times(m+n)} = \left[\frac{C^*_{m\times m}}{O^*_{n\times m}} \middle| \frac{I^*_{m\times n}}{S^*_{n\times n}}\right]$, where,

$$G^* = \lim_{t\to\infty} \sum_{i=1}^{t} M^i = (I - M)^{-1} - I. \quad 5\text{-}11$$

From the proof above, the following statement can be drawn.

Theorem 5-3 Spectrum of the CS network matrix potential

The spectrum of the service network $G_t = \sum_{j=1}^{t} M^j$ increases in all other CS network matrix *M* other than $M = I$.

Since $\lim_{i\to\infty} M^i = 0$, the potential of CS network matrix converges.

Theorem 5-4 Convergence of the potential of CS network matrix

The potential of CS network matrix $G^t = \sum_{i=1}^{t} M^i,\ t = 1,2,\ldots$ converges, e.g., $\lim_{t\to\infty} G^t = \lim_{t\to\infty} \sum_{i=1}^{t} M^i = G^*$.

Corollary 5-2 Stopping criteria

For $\forall\, \epsilon > 0, \exists\, t,\ \ s.t. \|M^t - M^{t-1}\|_1 \le \epsilon$, So, $G^* = \sum_{i=1}^{t} M^i$ is the final concept-service (CS) network matrix.

$$G^*_{(m+n)\times(m+n)} = \left[\frac{C^*_{m\times m}}{O^*_{n\times m}} \middle| \frac{I^*_{m\times n}}{S^*_{n\times n}}\right] = (I - M)^{-1} - I \quad 5\text{-}12$$

The potential of a network G^* is full of information, and can be used in service composition of any query. The potential of Concept Service (CS) network matches the

format of Katz index (Katz, 1953) in Sociometric analysis, which is described in Chapter 7.

5.3 Find Spanning Concept Vectors

There are several ways to find a spanning concept vector in the final concept-service network matrix G^*. The easiest way is to start with $O^*_{m\times n}$. The algorithm is as follows:

Algorithm 5-2 Find spanning concept vector

Step 0: Initialization: set A=ϕ. $B = \phi$;

Step 1: For each column $j = 1, \dots, m$, find an element $O^*_{ij} > 0$.

If $i \notin A$, put service number i in set A.

Step 2: For each number i in A,

Check the input concepts that service i needs in the initial input sub-array I in the CS network matrix M,. If the concept numbers are not in B, put the concepts in B.

Step 3: Let vector $X = (x_i)_n$, where $x_i = \begin{cases} 1 & if\ i \in B \\ 0 & if\ i \notin B \end{cases}$.

The vector obtained above satisfies $\Psi(X) = I$.

There may be many spanning vectors that can be found via the above algorithm. If there is no other vector $X' < X$, s.t. $\Psi(X') = I$, then X is a minimum spanning concept vector.

5.4 Find Service Composition

Given a known input concept vector X_0 and required output concept vector X_1, a service composition problem is to find the a set of services and their operating sequence $H(F)$, so that $H(F)(X_0) >= X_1$.

5.4.1 Existence Theorem of Service Composition

It is good to know that whether a service composition solution exists or not for any query. We judge whether a service composition exist according to **Theorem 4-1 Existence of Service Composition** introduced in Section 5.2: For a given query $q(X_0, X_1)$, a service composition exists if and only if $\Psi(X_0) \geq X_1$. From the theorem, the follow corollary can be derived.

Corollary 5-3 Existence of Service Composition

For a given query $q(X_0, X_1)$, a service composition exists if and only if $\sum_{i\in\{i|X_{0i}>0\}} G^*{}_{ij} > 0$, for $\forall\, j \in \{j|X_{1j} > 0\}$, where G^* is the potential of CS network matrix.

To proof this, we need show that $\sum_{i\in\{i|X_{0i}>0\}} G_{ij} > 0$, for $\forall\, j \in \{j|X_{1j} > 0\}$ equals to $\Psi(X_0) \geq X_1$ in **Theorem 4-1.**

5.4.2 Check Existence of Service Composition

based on the existence theorem of service composition, let us further explore whether a query $q(X_0, X_1)$ has a solution or not. The algorithm for checking solution existence is given in the following.

Algorithm 5-3 Check existence of service composition

Step 0: Initialization: set A=ϕ. $B = \phi$;

Step 1: For each element $X_{i0} > 0, i = 1, \dots, n$,

For each row i in $C^*_{m\times m}$, find an element $C^*_{ij} > 0$.

If $i \notin A$, put service number i in set A.

Step 3: Let $\Psi(X_0) = X$.

where $X = (x_i)_n$, where $x_i = \begin{cases} 1, & if\ i \in A \\ 0, & if\ i \notin A \end{cases}$.

Step 4: If $X \geq X_1$, service composition for query $q(X_0, X_1)$ exists;

If $X < X_1$, service composition for $q(X_0, X_1)$ does not exist.

Next, let us explore how to further find the service composition for a given query $q(X_0, X_1)$, if the solution exists.

5.4.3 Find a Service Composition based on CS Network Matrix

In this section, service composition methods based on the potential CS network.

Algorithm 5-4 Find service composition based on CS network matrix

Step 0: Check the existence of solution. If the solution exists, go to step 1; otherwise, quit.

Step 1: Let set $A = \phi, B = \phi, C = \phi, and\ D = \phi$

% check the services that can use the given input concepts both directly and indirectly

for each element x_{i0} in X_0,

if $x_{i0} > 0$,

for $j = 1, \dots, m$

if $I_{ij}^* > 0$, and $j \notin A$

put j in A.

% check the services that can generate the required output concepts both directly and indirectly

for each element x_{i1} in X_1,

if $x_{i1} > 0$,

for $j = 1, \dots, m$

if $O_{ji}^* > 0$, and $j \notin B$

put j in B.

% check the services that use the given input concepts directly

for each element x_{i0} in X_0,

if $x_{i0} > 0$,

for $j = 1, \dots, m$

if $I_{ij} > 0$, and $j \notin C$

put j in C.

% check the services that use the given input concepts directly

for each element x_{i1} in X_1,

if $x_{i1} > 0$,

for $j = 1, \dots, m$

if $O_{ji} > 0$, and $j \notin D$

put j in D.

Step 2: Draw the composition network

$E = C$

l=1

for each service element i in E

for j=1,…,m

if $j \in B\ and\ S_{ij} > 0$ in the S matrix in M^l

connect service $i\ and\ j$.

if $j \notin D$

Put j in set E

$l = l + 1$

However, it is not guaranteed that the solution is optimal with respect to criteria specified by the user or otherwise. In order to find the best composition regarding cost, reliability and delivery time, we need to further examine all the shortest possible paths in the graph found above. If there are many feasible paths, comparing all the possibilities one by one is time consuming. Mathematical programming based service composition algorithms will be designed in Chapter 6.

5.5 CS Matrix considering QoS (Quality of Service)

Many service composition problem take QoS into account. In order to consider quality of service (QoS) in service science, let us further define QoS operators.

Theoretically, many QoS measures can be considered; however in this section, let us take price, reliability and delivery time as examples. Let $\mathcal{F}$ represent the set of service operators.

(1) price as QoS

Definition 5-6 The *price* p_i of the service $f_i \in \mathcal{F}$ is a mapping from $\mathcal{F}$ to R^1, e.g., $p_i: \mathcal{F} \longrightarrow R^1$.

$$\text{For } \forall\, X \in \Omega,\ p_i(f_i)(X) = p_i \qquad 5\text{-}13$$

Definition 5-7 $p_i \oplus p_j$:

$$\left(p_i \oplus p_j\right)(f)(X) = \left[p_i(f) \oplus p_j(f)\right](X) = p_i(f)(X) \oplus p_j(f)(X),\ \forall X \in \Omega. \qquad 5\text{-}14$$

(2) reliability as QoS

Definition 5-8 The *reliability* of the service $f_i \in \mathcal{F}$ is a mapping from $\mathcal{F}$ to R^1, e.g., $R_i: \mathcal{F} \longrightarrow R^1$.

$$\text{For } \forall\, X \in \Omega,\ R_i(f_i)(X) = R_i \tag{5-15}$$

Definition 5-9 $R_i \oplus R_j$:

$$\big(R_i \oplus R_j\big)(f)(X) = \big[R_i(f) \oplus R_j(f)\big](X) = R_i(f)(X) \oplus R_j(f)(X),\ \forall X \in \Omega. \tag{5-16}$$

(3) delivery time as QoS

Definition 5-10 The *delivery time* of the service $f_i \in \mathcal{F}$ is a mapping from $\mathcal{F}$ to R^1, e.g., T_i: $\mathcal{F} \longrightarrow R^1$.

$$\text{For } \forall\, X \in \Omega,\ T_i(f_i)(X) = T_i \tag{5-17}$$

Next, let us define additive theorem of the operators.

Definition 5-11 $T_i \oplus T_j$**:**

$$\big(T_i \oplus T_j\big)(f)(X) = \big[T_i(f) \oplus T_j(f)\big](X) = T_i(f)(X) \oplus T_j(f)(X),\ \forall X \in \Omega. \tag{5-18}$$

Another difference of the algorithm considering QoS from algorithm in the previous section is that the matrix multiply and addition operations. The new matrix multiply and addition definitions are developed in this section to calculate the network composition considering QoS.

(1) for the price of services

Definition 5-12 *matrix multiply on cost* $A \odot_p B$:

$$A \odot_p B = \left(\min\nolimits_k \{a_{ik} + b_{kj}\}\right) \tag{5-19}$$

Definition 5-13 *matrix addition on cost* $A \oplus_p B$:

$$A \oplus_{\mathrm{p}} B = \begin{pmatrix} \begin{cases} \min\{a_{ij}, b_{ij}\} & if\ a_{ij} b_{ij} > 0 \\ \max\{a_{ij}, b_{ij}\} & if\ a_{ij} b_{ij} = 0 \end{cases} \end{pmatrix} \tag{5-20}$$

(2) for the reliability of services

Definition 5-14 *matrix multiply on reliability* $A \odot_R B$:

$$A \odot_{\mathrm{R}} B = (\min_k\{a_{ik}b_{kj}\}\ ) \tag{5-21}$$

Definition 5-15 *matrix addition on reliability* $A \oplus_R B$:

$$A \oplus_{\mathrm{R}} B = (\max\{a_{ij}, b_{ij}\}\ ) \tag{5-22}$$

(3) for the delivery time of services

Definition 5-16 *matrix multiply on delivery time* $A \odot_T B$:

$$A \odot_{\mathrm{T}} B = (\min_k\{a_{ik} + b_{kj}\}\ ) \tag{5-23}$$

Definition 5-17 matrix addition on delivery time $A \oplus_T B$:

$$A \oplus_{\mathrm{T}} B = \begin{pmatrix} \min\{a_{ij}, b_{ij}\} & if\ a_{ij}b_{ij} > 0 \\ \max\{a_{ij}, b_{ij}\} & if\ a_{ij}b_{ij} = 0 \end{pmatrix} \tag{5-24}$$

Next, let us take the price of service for example to show how this algorithm works.

$$M_{(m+n)\times(m+n)} = \begin{bmatrix} C_{\times n} & I_{n\times m} \\ \mathrm{O}_{m\times n} & S_{m\times m} \end{bmatrix}$$

The price information of services is contained in $I_{n\times m}$. Let $I'_{ij} = I_{ij}p_j$, where p_j is the price of service S_j, $j = 1,2,\dots,m$; $i = 1,2,\dots,n$

As for the positive elements in $C_{n\times n}$ and $O_{m\times n}$, we need to make them as small as possible so that they don't affect the price of services much, but can still be positive. Let $\alpha = \frac{1}{100}\min_{\mathrm{j}}\{p_j\}$, and

$$C'_{n\times n} = \alpha C_{n\times n} \tag{5-25}$$

$$O'_{m\times n} = \alpha O_{m\times n} \tag{5-26}$$

$$\text{Thus, } M'_{(m+n)\times(m+n)} = \begin{bmatrix} \alpha C_{n\times n} & (I_{ij}p_j)_{n\times m} \\ \alpha \mathrm{O}_{m\times n} & S_{m\times m} \end{bmatrix} \tag{5-27}$$

Normalize each column of M' by $\left(\frac{M_{ij}}{\|M_j\|_1}\right)_{(m+n)\times(m+n)}$

Different from the matrix operation defined in Euclidean space, the QoS operators defined above can be used to calculate the following potential of CS network M.

$$G_t = \sum_{j=1}^{t} M^j$$

The stopping criteria is still the same.

Stopping criteria: For $\forall\, \epsilon > 0, \exists\, t,\ \ s.t. \|M^t - M^{t-1}\|_1 \leq \epsilon$, So, $\ G^* = \sum_{i=1}^{t} M^i$ is the final concept-service network matrix.

$$G^*_{(n+m)\times(n+m)} = \left[\begin{array}{c|c} C^*_{n\times n} & I^*_{n\times m} \\ \hline O^*_{m\times n} & S^*_{m\times m} \end{array}\right]$$

Algorithm 5-5 Check the cost range of service composition for a query $\mathbf{q(X_0, X_1)}$

Step 0: For any concept j, where $x_{1j} > 0$,

for any concept i, where $x_{0_i} > 0$,

find $p_j = \min_i\{C^*_{ij}\}$,

Step 1: The ***cost of the optimal service composition*** for the query falls into the following range:

$$\max_j\{\min_i\{C^*_{ij}\}\} \leq P^*\big(q(X_0, X_1)\big) \leq \sum_{j\ \text{where } x_{1j}>0} \min_i\{C^*_{ij}\}$$

Algorithm 5-6 the reliability of the service composition for a query $\mathbf{q(X_0, X_1)}$

Step 0: For any concept j, where $x_{1j} > 0$,

for any concept i, where $x_{0_i} > 0$,

find $R_j = \max_i\{C^*_{ij}\}$,

Step 1: The reliability of the optimal service composition for the query falls in to the following range:

$$\min_j\{\max_i\{C_{ij}^*\}\} \le R^*\big(q(X_0, X_1)\big) \le \sum_{j \atop where\ x_{1j}>0} \max_i\{C_{ij}^*\}$$

Algorithm 5-7 the delivery time of service composition for a query $\mathbf{q(X_0, X_1)}$

Step 0: For any concept j, where $x_{1j} > 0$,
for any concept i, where $x_{0_i} > 0$,
find $T_j = \min_i\{C^*{}_{ij}\}$,

Step 1: The delivery time of the optimal service composition for the query falls in to the following range:

$$\max_j\{\min_i\{C_{ij}^*\}\} \le T^*\big(q(X_0, X_1)\big) \le \sum_{j \atop where\ x_{1j}>0} \min_i\{C_{ij}^*\}$$

So, the best possible case for the QoS metric will be the left hand size of each inequality. The following algorithm finds a service composition under QoS requirements:

Algorithm 5-8 Find a service composition considering QoS

Step 0: Check the existence of solution. If the solution exists, go to step 1; otherwise, stop.
Step 1: Let set $A = \phi, B = \phi, C = \phi, and\ D = \phi$
% check the services that can use the given input concepts both directly and indirectly
for each element x_{i0} in X_0,
if $x_{i0} > 0$,
for $j = 1, \dots, m$
if $I_{ij}^* > 0$, and $j \notin A$
put j in A.
% check the services that can generate the required output concepts both directly and indirectly
for each element x_{i1} in X_1,
if $x_{i1} > 0$,
for $j = 1, \dots, m$
if $O_{ji}^* > 0$, and $j \notin B$

put j in B.
% check the services that use the given input concepts directly
for each element x_{i0} in X_0,
if $x_{i0} > 0$,
for $j = 1, \dots, m$
if $I_{ij} > 0$, and $j \notin C$
put j in C.
% check the services that use the given input concepts directly
for each element x_{i1} in X_1,
if $x_{i1} > 0$,
for $j = 1, \dots, m$
if $O_{ji} > 0$, and $j \notin D$
put j in D.

Step 2: Draw the composition network
Trace the network for the left hand size of the QoS inequality.

$$\max_j\{\min_i\{C_{ij}^*\}\}$$

for the match in I^* and O^*, such that
$I_{ik}^* = \max_j\{\min_i\{C_{ij}^*\}\}$, and $O_{hj}^* = \max_j\{\min_i\{C_{ij}^*\}\}$
let $E=\{k,h\}$
$l=1$
for each service element i in E
for j=1,…,m
if $j \in B \; and \; S_{ij} > 0$ in the S matrix in M^l
connect service $i \; and \; j$.
if $j \notin D$
Put j in set E
$l = l + 1$

As we can see, the potential of Concept Service (CS) Network matrix can provide us information about the existence of solution, service composition set, and range of QoS of optimal solution. The method is fast and converges with complexity of $O((m+n)^2)$. However, it cannot provide us the optimal solution directly. We can further use network analytics to characterize the CS network.

5.6 Example of Concept Service (CS) Network

In this section, an example of Concept Service (CS) Networks is shown to demonstrate the mathematical theory of service composition.

Let $n = 9$, and $m = 5$. Initially, let $C = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$O = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_{14\times14} = \begin{bmatrix} C_{9\times9} & I_{9\times5} \\ \mathrm{O}_{5\times9} & S_{5\times5} \end{bmatrix}$$

Next, let us normalize $M_{14\times14}$ row by row.$M = \begin{bmatrix} \frac{M_1}{\|M_1\|_1} \\ \frac{M_2}{\|M_2\|_1} \\ \vdots \\ \frac{M_{14}}{\|M_{14}\|_1} \end{bmatrix}$.

$$M = \begin{bmatrix} \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 & \frac{1}{5} & 0 & 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let $\varepsilon = 0.001$, the algorithm stops after 12 steps, e.g. $\|M^{12} - M^{13}\|_\infty \leq \varepsilon$.

$$G^*_{14\times14} = \left[\frac{C^*_{9\times9}}{O^*_{5\times9}} \middle| \frac{I^*_{9\times5}}{S^*_{5\times5}}\right]$$

G* =

	C1	C2	C3	C4	C5	C6	C7	C8	C9	S1	S2	S3	S4	S5
C1	0.36	3.84	0.00	0.73	0.42	4.83	0.81	0.00	0.10	0.27	1.85	0.00	0.64	0.14
C2	0.00	9.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.78	0.00	0.00	0.00
C3	0.00	0.00	9.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.78	0.00	0.00
C4	0.00	6.44	0.00	1.00	0.67	0.00	1.39	0.00	0.16	0.00	3.11	0.00	1.00	0.22
C5	0.00	6.00	0.00	0.00	0.33	0.00	3.78	0.00	0.33	0.00	3.11	0.00	0.00	0.44
C6	0.00	0.00	0.00	0.00	0.00	14.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C7	0.00	0.00	0.00	0.00	0.00	0.00	14.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	14.00	0.00	0.00	0.00	0.00	0.00	0.00
C9	0.00	0.00	0.00	0.00	0.00	0.00	12.33	0.00	1.00	0.00	0.00	0.00	0.00	0.67
S1	0.45	3.13	0.00	0.91	0.36	6.16	0.64	0.00	0.09	0.09	1.50	0.00	0.55	0.12
S2	0.00	9.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.44	0.00	0.00	0.00
S3	0.00	0.00	9.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.44	0.00	0.00
S4	0.00	7.56	0.00	0.00	0.67	0.00	1.72	0.00	0.17	0.00	3.67	0.00	0.00	0.22
S5	0.00	0.00	0.00	0.00	0.00	0.00	12.67	0.00	1.00	0.00	0.00	0.00	0.00	0.33

From C^*, it can be seen that one of the spanning vectors of the CS network is $X_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$

X_1 is a spanning vector because . $\Psi(X_1) = I$; $X_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ is not a spanning vector since

$\Psi(X_2) < I$. In fact, we can show that X_1 is the minimum spanning vector.

For the query $q(X_3, X_4)$, where $X_3 = \begin{pmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{pmatrix}$, and $X_4 = \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix}$, there is no composition solution, since $C^*_{1,8}$=0, and $C^*_{3,8}$=0, or because $X_3 < X_1$ and X_1 is the minimum spanning vector.

For query $q(X_1, X_6)$, where $X_1 = \begin{pmatrix}1\\0\\1\\1\\0\\0\\0\\1\\1\end{pmatrix}$, and $X_6 = \begin{pmatrix}0\\0\\0\\0\\1\\0\\1\\0\\0\end{pmatrix}$, there is at least one composition solution, since $\Psi(X_1) = I$, $C^*_{1,5}$>0, and $C^*_{1,7}$>0.

Next, let us find the service composition for $q(X_1, X_6)$.

From I^*: since $I^*_{1,1} > 0$, $I^*_{1,2}$>0, $I^*_{1,4} > 0$, $I^*_{1,5} > 0$, $I^*_{3,3} > 0$, and $I^*_{9,5} > 0$, we have $A = \{S_1, S_2, S_3, S_4, S_5\}$

From O^*: since $O^*_{1,5} > 0$, $O^*_{1,7}$>0, $O^*_{4,5} > 0$, $O^*_{4,7} > 0$, and $O^*_{5,7} > 0$, we have $B = \{S_1, S_4, S_5\}$

So, $A \cap B = \{S_1, S_4, S_5\}$

From I, since $I_{1,1} > 0$, $I_{1,4} > 0$, $I_{3,3} > 0$, and $I_{9,5} > 0$, $C = \{S_1, S_3, S_4, S_5\}$

So, $A \cap B \cap C = \{S_1, S_4, S_5\}$

From O, since $O_{4,5} > 0$, and $O_{5,7} > 0$, $D = \{S_4,\ S_5\}$

So $A \cap B \cap D = \{S_4, S_5\}$

Then $A \cap B = \{S_1, S_4, S_5\}$ is a composition solution. Further we can check that $A \cap B \cap C \cap D$ $\{S_4, S_5\}$ is a smaller composition solution in this problem.

Assume that G^* is a Concept Service (CS) matrix with QoS as cost. From C^*, we can estimate the range of the solution cost is

$$\max_j\{\min_i\{C_{ij}^*\}\} \le P^*\big(q(X_0, X_1)\big) \le \sum_{\substack{j \\ where\ x_{1j}>0}} \min_i\{C_{ij}^*\}$$

So,

$$P^*\big(q(X_0, X_1)\big) \ge \max\{min\{0.42,0.67\}, \min\{1.81, 1.39, 14.00, 12.33\}\}$$

$$P^*\big(q(X_0, X_1)\big) \le min\{0.42,0.67\} + \min\{1.81, 1.39, 14.00, 12.33\}$$

Thus, $1.81 \le P^*\big(q(X_0, X_1)\big) \le 2.23$

5.7 Network Analytics

Next, network analytics can be further performed on service networks.

5.7.1 Related Literature: Characterization of Networks

The structure of networks conveys rich information useful for inference. The last decade has seen a proliferation of topological metrics. Let us discuss some of the important ones here. The order of a network is the total number of nodes (also called vertices), and its size is the total number of links (also called edges) in a network. The degree of a node is the number of links connecting the node to its neighbors. The incoming (in-degree) and outgoing (out-degree) sum up to the degree of a node. The degree distribution is a 2 dimensional graph showing the frequency of nodes with

different degrees in the network. The network density is the ratio between network size $\boldsymbol{m}$ and the maximum possible number of links. One of the most important measures that has been explored is the distance. The distance between two nodes is the length of the shortest path between them, i.e. the minimum number of links that one needs to follow when going from one node to the other. The shortest path can be found through Dijkstra's algorithm. The average path length of a network is the average value of distance between any pair of nodes in the network.

Table 5-1 Metrics for networks (Source: Complex Networks: An Engineering View, Cui et al., 2010)

Metric	**Math equation (in undirected graph)**
Order	*The number of nodes n*
Size	$m = \sum_i \sum_j a_{ij}$, where $a_{ij} = \begin{cases} 1, & node\, i\, and\ j\, are\, linked \\ 0, & otherwise \end{cases}$
Node degree	$d = \sum_i a_{ij}$, where $a_{ij} = \begin{cases} 1, & node\, i\, and\ j\, are\, linked \\ 0, & otherwise \end{cases}$
Density	$\delta = \frac{2m}{n(n-1)}$, where m is the number of arcs in the network and n is the number of nodes in the network.
Average path length	$l = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij}$, where d_{ij} is the distance between node i and node j, and n is the number of nodes in the network.
Diameter	$D = \max\{d_{ij}\}$, where d_{ij} is the distance between node i and node j, and n is the number of nodes in the network.
Clustering coefficient of a node	$C_i = \frac{2t_i}{k_i(k_i-1)}$, where t_i is the number of triangles centered at node i, and k_i is the number of degrees of node i.
Clustering coefficient network	$C = \frac{1}{n} \sum_i C_i$, where C_i is the clustering coefficient of node i.

Betweeness	$B_k = \sum_{k \in p(i,j)} 1/n_{ij}$, where n_{ij} is the number of path from node i to node j.
Proximity ratio	$\mu = \frac{\frac{C}{l}}{\frac{C_{rand}}{l_{rand}}}$, where C is the clustering coefficient of the network, and l is the average path length of the network; C_{rand} is the clustering coefficient of a random network and l_{rand} is the average path length of a random graph.
Efficiency	$E_{global} = \frac{1}{n(n-1)\sum_{i \neq j} \frac{1}{d_{ij}}}$, where d_{ij} is the distance between node i and node j
Mixing coefficient	$r = \frac{\sum_i \left(dg_i - \bar{dg} \right)\left(dn_i - \bar{dn} \right)}{\sqrt{\sum_i \left(dg_i - \bar{dg} \right)^2 \left(dn_i - \bar{dn} \right)^2}}$, where dg_i is the degree of node i, and dn_i is the first order mean degree of the neighbors of node i. The mixing coefficient is the standard devotion.
Modularity index	$Q = \sum_i \left(e_{ii} - a_i \right)^2$, where e_{ii} is the fraction of edges in subgraph i; a_i is the fraction of edges between this subgroup and all other subgraph.

The diameter of the network is the longest distance between any pair of nodes in a network. The clustering coefficient of a node is the number of triangles centered at the node divided by the number of triples centered at the node. The clustering coefficient of a network is the arithmetic mean of the clustering coefficients of all the nodes. The betweenness centrality quantifies how much a node is between other pairs of nodes. Let us define the ratio between the clustering coefficient and the average path length as the CP ratio. The proximity ratio of a network is the CP ratio between this network and a random network. This property captures the extent of a network's small-worldness. The efficiency of a network is the communication effectiveness of a networked system (global)

or of a single node (local). The mixing coefficient is the Pearson correlation between the degrees of neighboring nodes. The modularity index measures the topological similarity in the local patterns of linking. Table 5-1 summarizes the most commonly used metrics and their equations.(Baggio et al., 2010) show a similar table.

The current network metrics characterize the network at three levels: micro, global, and mesoscopic levels. The properties of individual nodes or edges such as degree, centrality and node rank functions are at the micro level; the properties of the entire network, such as degree distribution, diameter and clustering coefficient are at the global level; the detection of cohesive groups, e.g. communities is at the mesoscopic level. The recently introduced concept of "stochastic block structures" (Reichardt, 2010) is also at the mesoscopic level and it is shown that communities are a special case of this concept in a static network. Stochastic block structures can be applied in studying the evolution of complex networks.

5.7.2 Related Literature: Network Models

Before the computerization of data acquisition and sharing allowed the mapping of real-world networks, several abstract prototypical networks were studied.

The mathematicians Erdös and Rényi analyzed *Random networks* based on a set of nodes (ER model). In this network, the links are added into a network consisting of n nodes and with no links among them. The link between any pair of nodes is placed with a probability of p. The degree distribution of the network follows Poisson distribution with the average node degree of $< k > = np$, thus $P(k) \approx \frac{<k>^k e^{-<k>}}{k!} = \frac{(np)^k e^{-(np)}}{k!}$. For a large node set, there is a giant connected component if the average number of links per

node is larger than 1. The average path length l in this giant connected component scales as $\ln(n)$, for example $l \propto \ln(n)$. Further, $l = \ln(n - < k >)$. In this resulting network, most of the nodes will have a degree of $< k >$. When $p < 1/n$, almost all the vertices in the network belong to isolated trees. Cycles of all orders in the random network appear at $p \approx 1/n$.

Regular networks include rings, lattices, trees, stars and complete graphs. A *ring* is a connected graph in which a node is linked exactly with two other nodes. A *lattice* is a graph in which the nodes are placed on a grid and the neighbors are connected by an edge. A one dimensional lattice is like a chain. A *tree* is a connected graph containing no circles. A *star* graph is a tree in which every node is connected to the root. In a *full (complete)* graph there is an edge between all pairs of nodes. The *Graph Atlas* (Read and Wilson, 1998) contains all undirected graphs with up to seven nodes. Regular networks are often used in materials science and in parallel computing in computer science.

One of the most important network models is the *small-world network*. The *small-world* concept describes the fact in a network that regardless of size there is a relatively short path between any two nodes. The small-world phenomenon was first mentioned by the writer Frigyes Karinthy, in Hungary, in 1929. 30 years later, it became a research problem named "contact and influence" (Kochen and Pool, 1978) posed the question "What is the probability that two strangers will have a mutual friend?", "When there is no mutual friend, how long would the chain of intermediaries be?" In the seminal work of Watts and Strogatz, the small world concept was expanded to mean networks with low average shortest path length and a high clustering coefficient, which therefore are in a sense situated between regular and random networks. The (Watts and Strogatz, 1998)

model (WS model) starts from a ring lattice with vertices and k edges per vertex, and rewires each edge at random with probability p. This construction tunes the graph from regular graph (p=0) to a random graph (p=1). Small word networks ($0 < p < 1$) have high clustering coefficient and a low average path length. The average path length of this network scales in $\log_{<k>} n$. Here $< k >$ is the average out-degree and n is the number of nodes. When pairs are selected uniformly at random, they are connected by a short path with high probability. Many real world networks exhibit this property. Random networks also have small average distances, however they are not clustered. Small-world networks usually appear in social sciences.

Many systems in real world are dynamic and the order of the networks (number of nodes) grows over time. The concept of a scale-free network was introduced by Barabási and Albert (BA model) in 1999. The term scale-free represents the property of a function that changing the variable in a scale results in changing the function value in scale as well, *e.g.*, $f(ax) = bf(x)$. The power-law form $f(x) = \alpha x^{-\beta}$ is the only form that satisfies the scale-free property. In a scale-free network, the number of nodes n is expected to change over time. In BA model, the network dynamics is described by introducing new nodes into an existing network. When a vertex is linked in, it tends to link with higher probability to a vertex that already has a large number of edges, which is the so called *preferential attachment*. The probability for a new node to be attached to an existing node is proportional to how many links the existing node already has. $\prod(k_i) = \frac{k_i}{\sum_j k_j}$. The resulting network has a degree distribution that follows $p(k) \propto k^{-\alpha}$. Many networks in real world have the scale-free property, especially in social and biological sciences.

5.7.3 Service Networks

The characteristics of web service network are:

(1)The maximum distance between two nodes may be infinity, which means that the two nodes have no relationship with each other.

(2) The minimum degree of a service is 2; the minimum degree of a concept is 1.

(3) There could be some circuits.

(4) There may be some completely connected sub-graphs in the network.

(5) The network is directed, but not symmetrical.

(6) The connectivity of the graph is the most informative measure in the web service network, since it is the basis to composite individual web services into compound services.

The network analysis includes:

(1) Find the components and bridge information in the network. This can help the service brokers to judge the effects of a catastrophe upon some queries in the network. In Figure 5-2, nodes $7, 12, 8,$ and 3 belong to component 1 (marked in pink); nodes $1, 2, 4, 5, 6$ and 11 are in component 2 (marked in blue); and nodes $9, 10$ and 13 form component 3 (marked in yellow). And arc$(4, 3)$ is the bridge from the component 2 to component 1. So, arc $(4, 3)$ is important, if we need to integrate component 1 and component 2 in order to form compound services that satisfy some customers' requests.

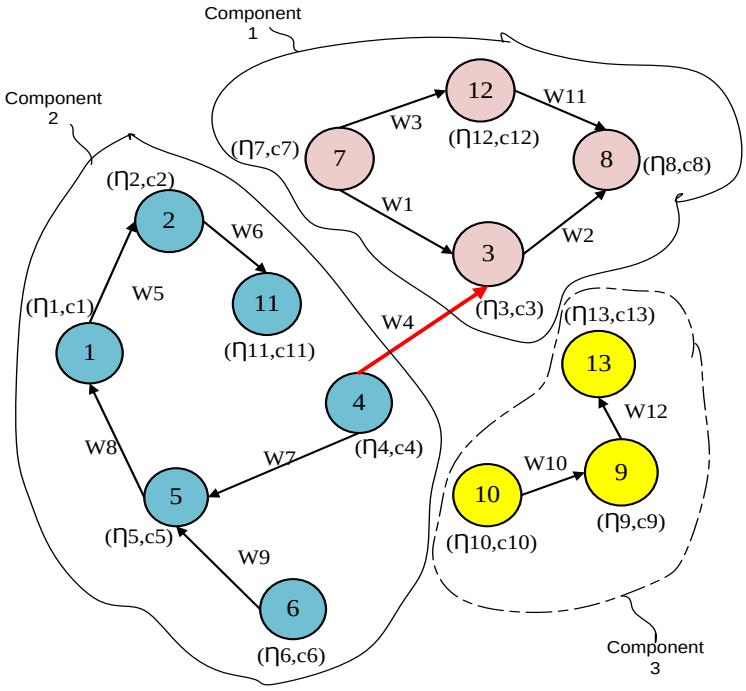


Figure 5-2 A small network of web services

(2) Calculate the centrality for services. This can help the system to find out the important services.

(3) Analyze the distance between the services. The information can be used for service computing.

(4) Utilize the network dynamics and analyze the characteristics mentioned above. This can help identify the hidden information from the network. The dynamics can include edge growth, node growth or both. In web service composition, the edge growth dynamics is most informative, which is the focus in the next section.

Next, let us study the edge growth in CS network. H to use Boolean dynamic network to find the potential of CS network is developed. A Boolean dynamic network is

a network with the status of its nodes and edges changes in time, and both nodes and edges in it can only have two states: on and off (or 1 and 0). In the CS network, the procedure of calculating the CS matrix potential is the same as growing the edges via the following dynamics.

$$e_{ij}(t) = \begin{cases} 1 & \text{if } l_{ij} \le t \\ 0 & otherwise \end{cases} \quad 5\text{-}28$$

where $e_{ij}(t)$ is the edge from node i to node j at step t, and l_{ij} is the shortest path length from node i to node j. This dynamics shows that whether an edge $e_{ij}(t)$ exists or not depends on whether the shortest path length from node i to node j is equal to or less than t or not. The above edge growth dynamics can be further studied by the following boolean dynamics:

$$N_j(t+1) = \begin{cases} N_i(t) & \text{if } e_{ij}(t) = 1 \\ N_j(t) & \text{if } e_{ij}(t) = 0 \end{cases} \quad 5\text{-}29$$

$$e_{kj}(t+1) = \begin{cases} 1 & \text{if } N_j(t+1) = 1 \\ 0 & \text{if } N_j(t+1) = 0 \end{cases} \quad 5\text{-}30$$

Equation (5-29) shows how to update the node status in a network. $N_j(t)$ is the status of node j at time t. $N_j(t) = 1$ means the node j is on, and $N_j(t) = 0$ means the node j is off. Equation (5-30) shows how to update the edge status in a network. $e_{ij}(t)$ is the status of the edge between node i and node j. $e_{ij} = 0$ means the edge between node i and node j does not exist at time t; $e_{ij} = 1$ means the edge between node i and node j is on at time t. **Algorithm 5-9** shows how to calculate the potential of CS network via Boolean dynamic network.

Algorithm 5-9 Calculate CS network potential by Boolean networks

Main function **CS-Boolean(E,V)**

For any node k, $k = 1, 2, \dots, m+n$ do

$$N_k(1) = 1, \qquad N_j(1) = 0, \qquad j \neq k;\ j = 1,2, \dots, m+n$$

$$t = 1$$

Repeat

Upgrade node status using Equation()

Upgrade edge status using Equation()

$$t \leftarrow t + 1$$

Until any node $N_j(t+1) = N_j(t)$, $j = 1,2, \dots, m+n;\ j \neq k$

End for

End

The node status in Equation (5-29) is an intermediate step for updating edge status in Equation (5-30). We are only interested in edge growth in CS network. An example service network with 45 concept nodes and 35 service nodes is constructed. To generate the initial network, we assume that the connectivity probability among nodes is $p = 0.01$; the connectivity probability between concepts and web services is $p = 0.04$. And there is no arc among web services. No self-loop is allowed in the network. Service network is like a bipartite- network but not a bipartite networks, because though the majority of the links are between concept nodes and service nodes, there is still connection among concept nodes and service nodes. (see Figure 5-3). In this network, there are some isolated concept nodes not used by any service, which means that they are not important for the network in reality.

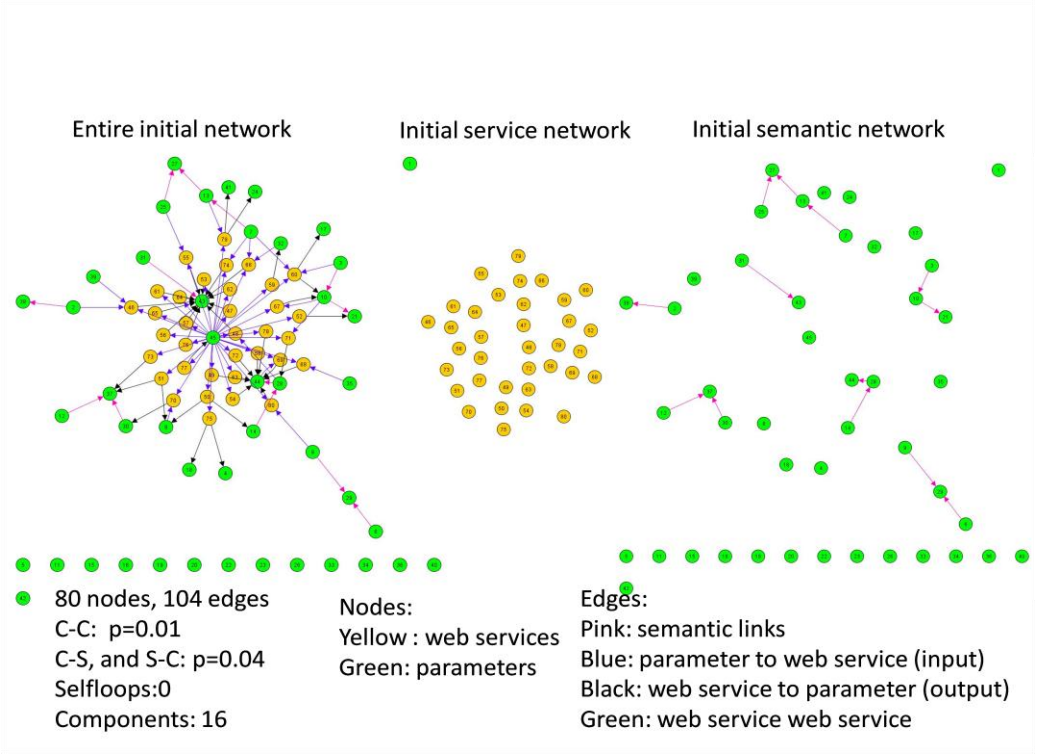


Figure 5-3 Initial service network generation

Use Centrality Analysis to the initial CS network, let us see the result in Figure 5-4. The node 44 is the mostly often used input concept in the network and 43 and 45 are the most often used output concepts in the network.

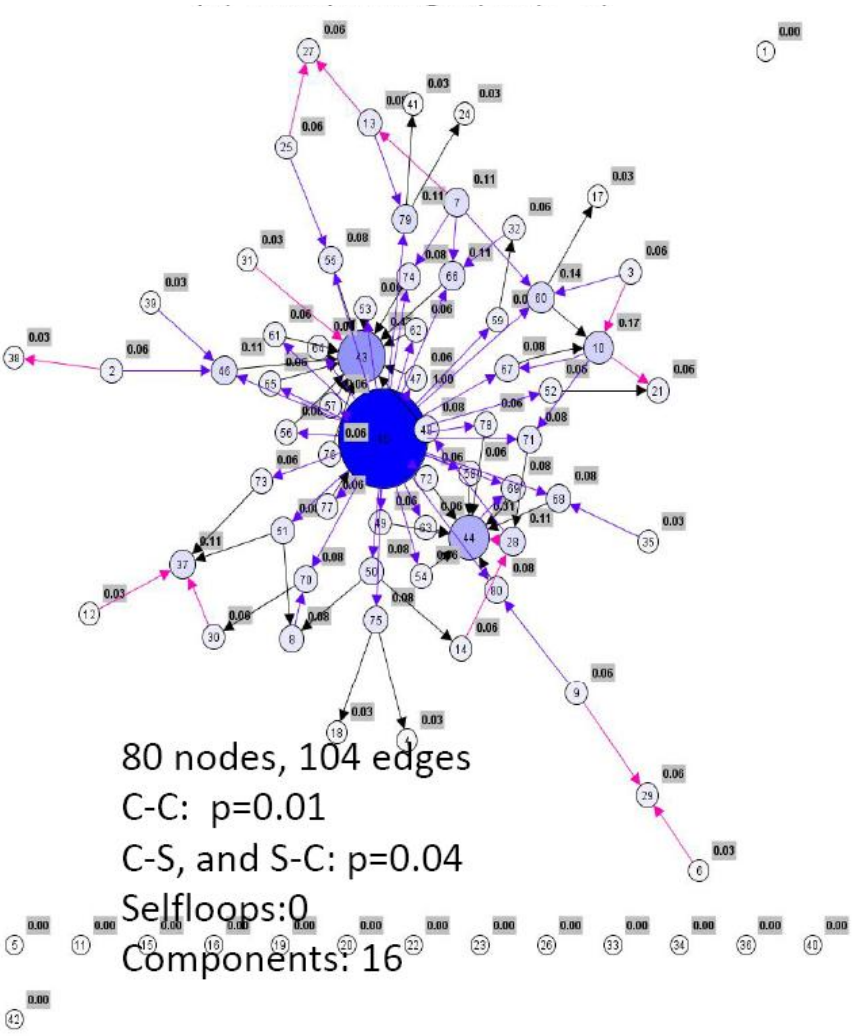


Figure 5-4 Centrality analysis of the network

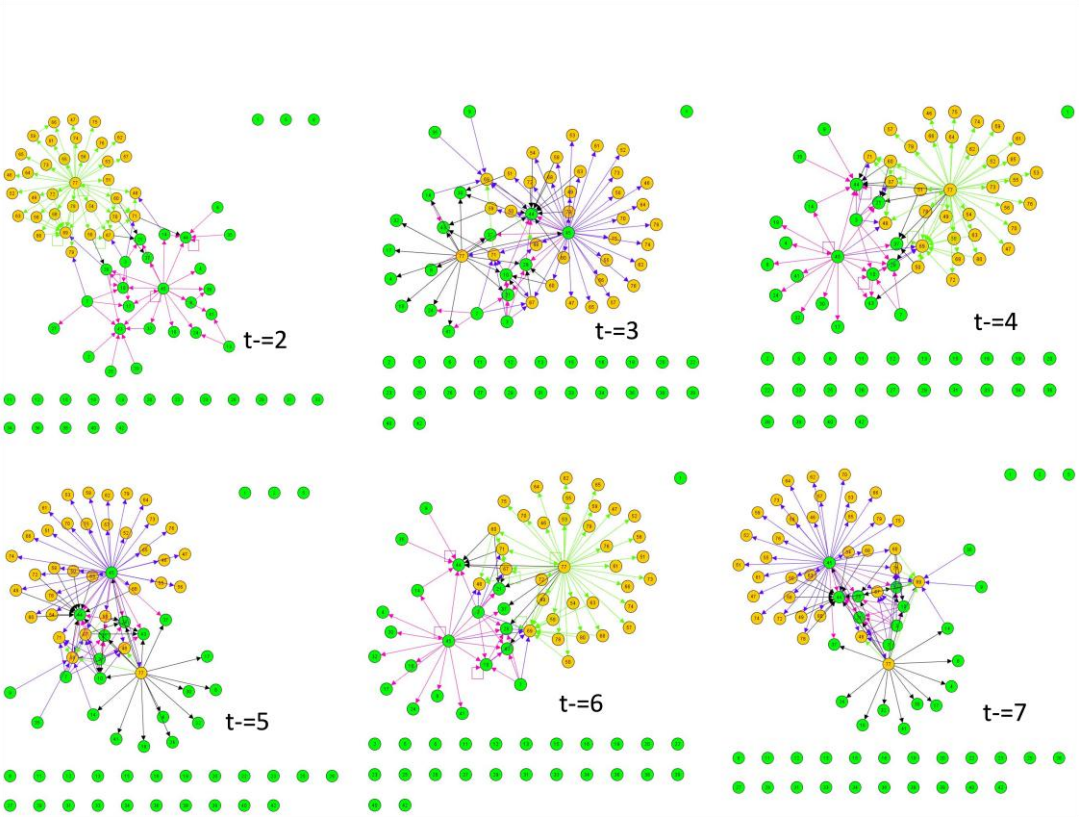
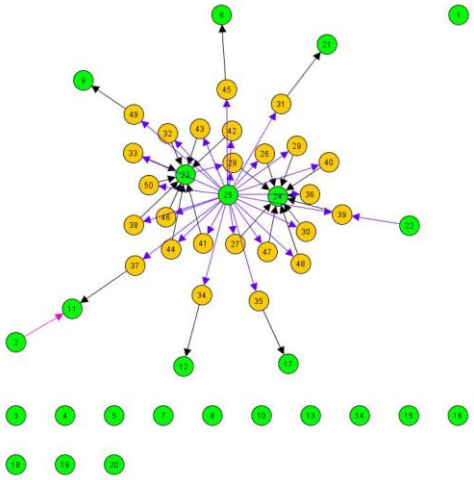
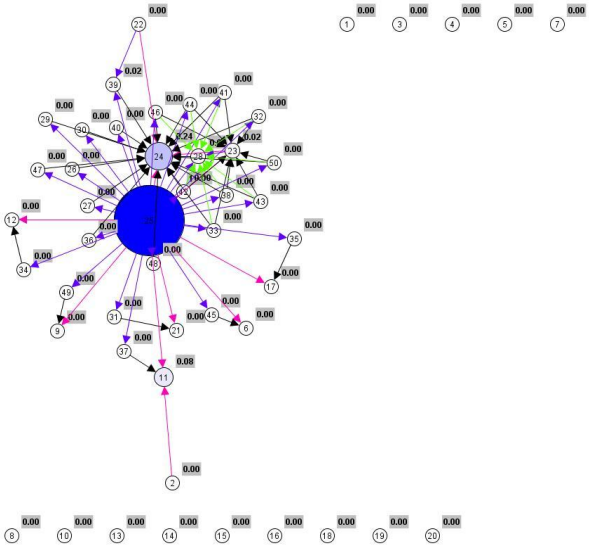


Figure 5-5 Edge growth in the network within 7 steps

In Figure 5-5, the edge growth at each step of t=1,2,3,4,5,6,7 is shown. The picture only shows the new edges added in the network at that particular step. It is noticed that self-loops occurs at step t=2. This means that there are cycles of length 2 in the initial network. According to the experiments run for this type of network (web service network), the network will not converge within a finite number of steps. However, if there is no cycle of any length in the original network, the web service network will be able to converge with finite times of propagations. For example, the experiment on the network in Figure 5-5 converges at step t=5, since there are no cycles in the initial network.



(a) The network at step 5



(b) Degree centrality analysis of the network at step 5

Figure 5-6 A sample converges with 5 propagations

From Figure 5-5, it can be seen that the web services start to have links among them at t=2. And more semantic links are built among the concepts after a few steps. And in each step there are new edges built between web services and concepts. Figure 5-6 (a) is the CS network at step $t = 5$, and Figure 5-6 (b) is the centrality analysis of the network at step $t = 5$.

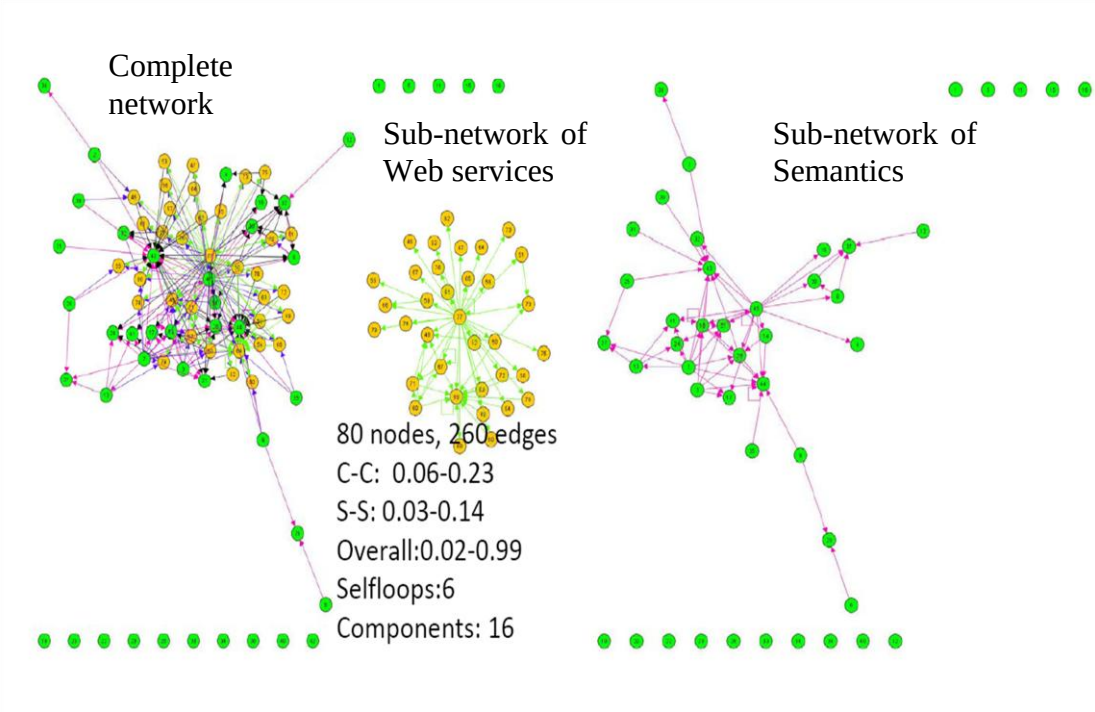


Figure 5-7 Network at step t=7

At the step t=7 shown in Figure 5-7, the web services are connected, and the semantic network are well developed. The number of edges increased from 104 to 260. The connection probability among concepts is between 0.06 and 0.23, and the connection probability among services is between 0.03 and 0.14. The overall connection probability of nodes is between 0.02 and 0.99. There are a few very popular concepts and services in the network. The number of self loops is 6. The number of isolated sub graphs doesn't decrease in this case.

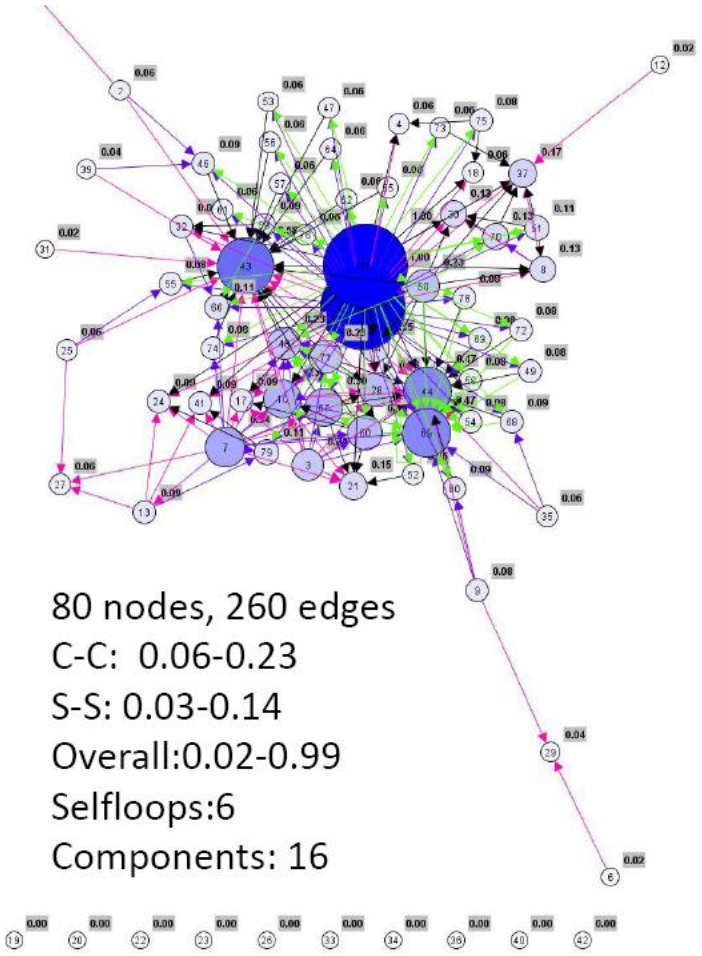


Figure 5-8 Centrality analysis of the network at step t=7

As we can see from Figure 5-8, popular web services and concepts are: node 77, node 43, node 44, node 45, node 69, node 7, *etc.* which cannot be seen from the original network, although, node 43, node 44 and node 45 are already shown to be popular. So, the network after evolution as shown can expose hidden information in it. And it is useful in service computing, robustness analysis and even rescue from catastrophe in the future.

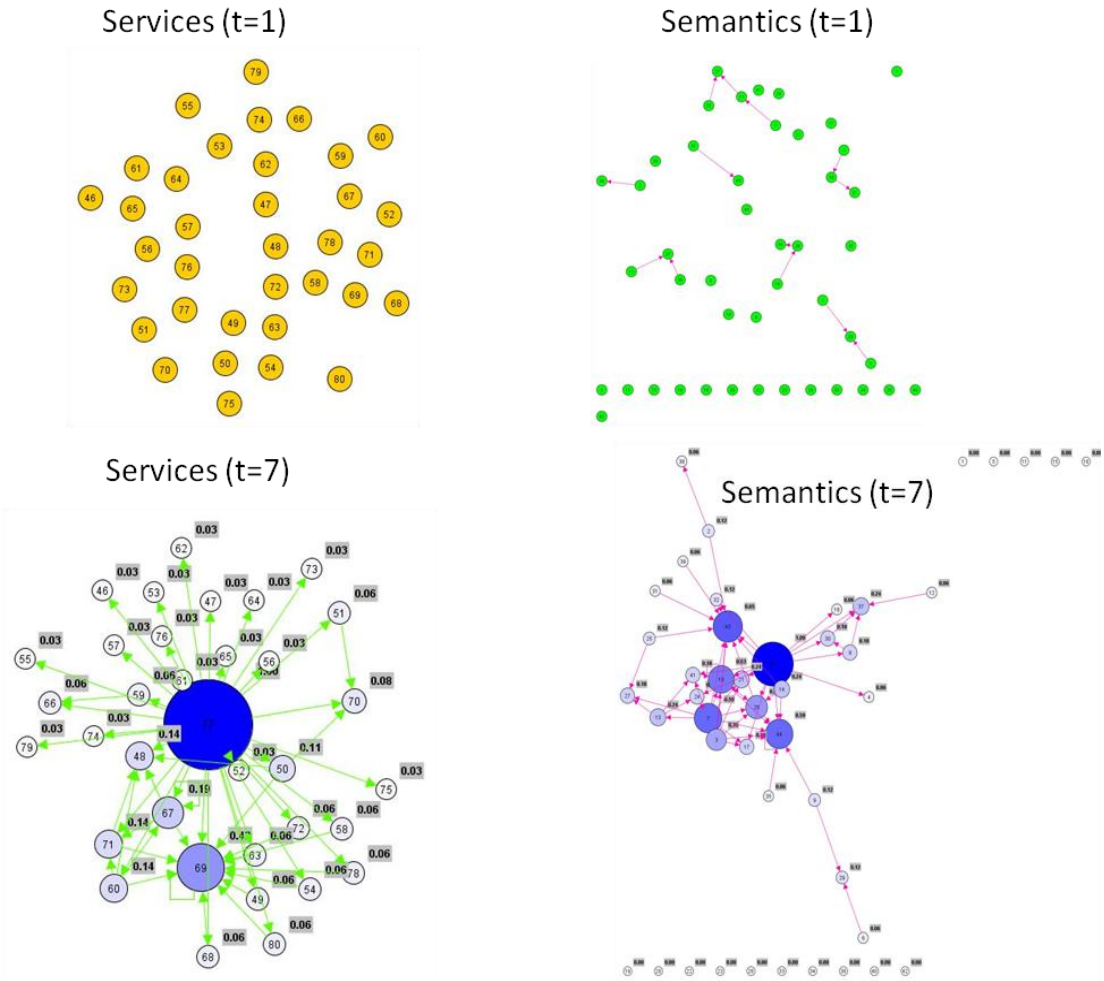


Figure 5-9 Compare the sub-network of services and the sub-network of semantics

Figure 5-9 shows the comparison of CS networks between step $t = 1$ and step $t = 7$. It is shown that a network can grow in the number of edges. From the sub-graph of services separated, we can see that many relations among services are built in step $t = 7$. The semantic connection among concepts is sparse in the initial network, and the hidden semantic relations among concepts are well developed and identified in the network at step t=7. In the meantime, new popular services and concepts occur. These nodes that are

the predecessors of the nodes with a high out-degree have the potential to be popular when the network evolves.

So, analyzing network evolution can help us to predict the long-run network structure conveniently. How accurate the prediction is depends on how accurately the Boolean rules describe the real world problem.

The service composition algorithms above are fast and Concept Service (CS) networks are informative. However, it is not accurate enough for some composition problems because it cannot differentiate between the follow patterns of "OR parallel" and "And parallel."

5.8 Conclusions and Future work

In this chapter, a concept service (CS) network matrix is explored and a set of service composition algorithms are introduced based on it. In the CS network (matrix M), nodes represent web services and concepts. The directed edges represent the input relations (matrix I), output relations (matrix O), semantic relations (matrix C) and service relations (matrix S) between web services and concepts. The potential of concept service (CS) network matrix can provide us information about the existence of solution, service composition set, and range of QoS of optimal solution. The method can converge with complexity of $O((m+n)^2)$. The network structure analysis can help the system to make recommendations to customers and service providers. It shows how robust the current network is when catastrophic events such as node and link failures occur. And these network structures can be utilized in web service composition.

In the future, the node growth of Concept Service (CS) networks, e.g., how new services and concepts can be added into the existing network will be useful to explore. Currently, the web service network is under development worldwide. When a service network is growing, it is reasonable to assume that: (1) The nodes with a higher capacity $\boldsymbol{c_i}$ have the potential to be used by more services in the network; (2) The nodes with the same classification η_i are more likely to have a higher correlation between any two of them; (3) Two nodes sharing concepts are more correlated with each other; (4) The arcs appearing in the composition results more frequently have a stronger impact on the network, and contribute more to the correlation between the two end services that connect them. In this type of network, the correlation score between two services can be calculated by the following:

$$r_{ij} = \left[1 - \frac{\sum_{all\ arcs\ on\ the\ path\ i \to j} \frac{1}{w_{pq}}}{\sum_{all\ arcs\ in\ G} \frac{1}{w_{pq}}}\right] * \eta_{ij}$$

$$where\ \eta_{ij} = \begin{cases} 1, & if\ \eta_i = \eta_t \\ \dfrac{1}{(\eta_i - \eta_j)^2}, & if\ \eta_i \neq \eta_t \end{cases}$$

In the future, it is also useful to analyze the network evolution when new nodes are added by preferential attachment. Start with a network with an initial network with m_0 nodes and e_0 edges. Each node has an attribute η_i which depends on the type of the node, and an attribute c_i which indicate the capacity of the service provider; Add a new node with m number of edges which connects to different existing nodes in the network;

the preferential attachment is: $\Pi(k_i, t) = \begin{cases} \frac{k_i * c_i}{\sum_j k_j c_j}, & if\ \eta_i = \eta_t \\ \frac{\frac{1}{(\eta_i - \eta_t)^2} k_i c_i}{\sum_j \frac{1}{(\eta_i - \eta_t)^2} k_j c_j}, & if\ \eta_i \neq \eta_t \end{cases}$ In this case, the preferential attachment of the new node to an existing node does not only depend on the degree of the existing node, but also depends on the similarity between the type of the new node to the type of existing nodes. A new node will be more likely to be connected with the existing node that has the similar type with itself and with high degree. This model is similar to the weighted preferential attachment model, but the weights change depending upon the relationship between the new node and the existing nodes. In addition to the type (for example semantics) of the node, the capacity of the nodes also plays a critical role in defining this relationship. The number of nodes and edges change in time. After T time steps, the number of nodes will be $m_0 + t$, and the number of edges will be $e_0 + m * t$.

Next, the service composition problem can be further formulated as mathematical programming in order to find a set of services and the relation among them (Chapter 6), and to study the uncertainty in service networks, bandwidth of the links, and the capacity of web services in real world(Chapter 7).

Chapter 6. Service Composition based on Multi-criteria Goal Programming

Based on the mathematical foundation of service computing established in Chapter 4, we can quickly find composition solutions for a query using the potential of Concept Service (CS) network matrix. However, the service composition problem needs optimization in order to find the best set of services and the relation among them, denoted as $H(F)$, such that the cost of the solution can be minimized, the reliability of the solution can be maximized, and the delivery time of the solution can be minimized.

$$Min\ P(H(F)(X_0)) \tag{6-1}$$

$$Max\ R(H(F)(X_0)) \tag{6-2}$$

$$Min\ T(H(F)(X_0)) \tag{6-3}$$

$$\text{s.t.} \quad H(F)(X_0) \geq X_1 \tag{6-4}$$

where $q(X_0, X_1)$ is a given query, $P(H(F)(X_0))$ is the cost of the service composition; $R(H(F)(X_0))$ is the reliability of the service composition; $T(H(F)(X_0))$is the delivery time of the service composition. It will be preferred if other QoS attributes can also be optimal in the solution. This mathematical optimization problem will be discussed in this chapter.

In order to overcome the drawback of inaccuracy of the Concept Service (CS) network models, a multi-criteria goal programming method is studied in this chapter. Moreover, it combines routing and Quality of Service (QoS) filtering together, while many traditional mathematical models need to have Heuristic methods to search for feasible solutions before utilizing the multi-criteria mathematical model.

6.1 Related Literature: Integer Programming Model

We need to decide three types of things in order to formulate the mathematical model, and they are variables, objective functions and constraints. Zeng *et. al.* (2004) gave a mathematical programming approach to maximize or minimize the value of the objective functions by adjusting the values of variables while enforcing the constraints.

After collecting QoS information, a quality vector is computed for each of the candidate web services, and based on these quality vectors, the system selects one of the candidate web services by applying a Multiple Criteria Decision Making (MCDM). This selection process is based on the weight assigned by the user to each criterion, and a set of user-defined constraints. To illustrate the local optimization approach, five quality dimensions are discussed: 1. $price$, 2. $duration$, 3. $availability$, 4. $reliability$, and 5. $reputation$. Given a task t_j in a compound service, there is a set of candidate web services $S_j = \{s_{1j}, s_{2j}, \dots, s_{nj}\}$ that can be used to execute this task. By merging the quality vectors of all these candidate web services, a matrix $Q = (Q_{i,j};\ 1 \le i \le n, 1 \le j \le 5)$ is built, in which each row Q_j corresponds to a web service s_{ij} while each column corresponds to a quality dimension. There are six steps need to be taken care of in Zeng *et al.*'s multi-criteria mathematical programming approach to web service composition problem.

Step 1. Decompose the query into tasks: In Zeng *et al.*'s paper, the first step is to decompose the query to tasks. The next step is to construct the multi-criteria mathematical programming.

Step 2. Variable definition: First, for every web service s_i that can be used to execute a task t_j, the integer variables are y_{ij}, such that y_{ij} is 1 if service s_{ij} is selected for executing task t_j; otherwise, it is 0. Let x_j denote the expected start time of task t_j (if t_j is executed).

Step 3. Scaling: Scaling is important in multi-criteria mathematical programming. Without scaling, these criteria with high values (for instance, price $500, and execution time 2500 seconds) will dominate these criteria with low values (for instance, reliability 0.7, availability 0.89 and reputation 5), so the solution will be always favorable to these criteria with large values.

Step 4. Weighting: Different criteria need to be assigned weight so that they can be compared. The following formula is used to compute the overall quality score for web service s_{it} (the i^{th} service which can finish task t.): $Score(s_{i,t}) = \sum_{k=1}^{5}(V_{i,k} * w_k)$, $i = 1,2, \dots, n$; for task t, where $w_j \in [0,1]$ and $\sum_{j=1}^{5} w_j = 1$. w_j represents the weight of criterion j. As stated before, decide the weights w_j to their own preference on QoS criteria. Different users may give different weights to the same criterion.

Step 5. Construct constraints: For each task t_j , there is a set of web services S_j that can be assigned to it. However, for each task t_j, It is only allowed to select one web service to execute this task. Given that y_{ij} denotes the selection of web service s_{ij} to execute task t_j , the following constraint must be satisfied: $\sum_{i \in S_j} y_{ij} = 1, \forall\, j \in A$, where A is the set of tasks in the state chart.

Let x_j denote the expected start time of task t_j, p_{ij} denotes the execution duration of task t_j when assigned to service $s_{i,j}$, and p_j denotes the expected duration of task t_j knowing which service has been assigned to it. Also, let $t_j \rightarrow t_k$ denote the fact that task t_k is a direct successor of task t_j. So $\sum_{i \in S_j} y_{ij} = 1, \forall\, j \in A; x_k - (p_j + x_j) \geq 0, \forall\, t_j \rightarrow t_k, j, k \in A;$ and $D - (x_j + p_j) \geq 0, \forall\, j \in A$. The first constraint means that the every task has to be executed by some service, since one and only one of these services will be selected to execute task t_j. The second constraint indicates that if task t_k is a direct successor of task t_j, then the execution of t_k must start after task t_j has been completed. Constraint. $t_j \rightarrow t_k$ means job t_j is a direct predecessor of job t_k. The third constraint indicates that the execution of a compound service plan is completed only when all the tasks in the plan are completed. So, the make span of this service complex is greater than the completion time of any times in the plan. Let $z_{ij} = \begin{cases} 1, & if\ Z_{ij}\ is\ on\ the\ critical\ path; \\ 0, & if\ Z_{ij}\ is\ not\ on\ the\ critical\ path \end{cases}$ The relationship between the duration of an execution plan and the duration of the critical services of the plan is captured by $D = \sum_{j \in A} \sum_{i \in S_j} p_{ij} z_{ij}$. Similarly, assuming that variable c_{ij} represents the price of web service s_{ij}, The total price of a plan is $\sum_{j \in A} \sum_{i \in S_j} c_{ij} y_{ij} \leq B$, where B is the budget of the customer.

Step 6. Construct objective: Assuming that variable r_{ij} represents the reputation of web service s_{ij} , the following expression indicates the overall reputation of an execution plan: $Rep = \sum_{j \in A} \sum_{i \in S_j} r_{ij} y_{ij}$. The availability and the reliability are with nonlinear aggregation functions. In order to capture them in the IP problem, Zeng *et. al.*

linearize them using a logarithmic function. The other four objectives can be constructed differently.

The advantage of this multi-criteria mathematical programming model is that it solves the optimal solution according users' preference on QoS. The drawback of the model proposed by Zeng *et.al.* lies in that it assumes the query from the users can be explicitly partitioned into tasks, and that we know how many services and what services can be assigned to this task, so that a pre-checking process is needed prior to the mathematical modeling. This process adds complexity to the model. The model only optimizes users' utility on QoS, instead of taking both QoS and functional requirements into the mathematical composition model explicitly. Zeng *et.al.*'s model forces hard QoS constraints instead of soft ones, which means the model does not allow comprising solutions in case that some of the QoS constraints cannot be satisfied at the same time.

In modern business environments, the ability to provide a user with personalized services has become pivotal factor for enterprises. And users' preferences are also drawing more and more attention in the service-oriented research field. However, most of the traditional automated service composition approaches cannot meet of the personalized user requirements in a flexible manner according to the real situations. The users' requirements not only include the hard constraints, but also include the soft-constraints. Consider the travel planning example in Chapter 1, it is necessary for the customer to compromise among her preferences so that she can obtain a satisfactory solution instead of no solution. This kind of soft-constraint is usually negotiable according to the tradeoff. In addition, the soft-constraints will also improve the success rate of service composition

thanks to the negotiable feature. Goal programming solves the problem in accordance with the customers' preferences. The customers' requirements on aggregating price, reliability of service, and total service time are considered as goals. Both preemptive and non-preemptive goal programming models are built and tested. In this chapter, compromising solutions are considered when there is no solution that can satisfy all the requirements from the customer exists. In the mean time, it incorporates the functional constraints in the model. So, it does not require any pre-processing of plans before using the proposed model. This chapter discusses the theoretical development and application of the models in detail.

In this dissertation, three different scenarios of multi-criteria mathematical programming models are explored under a framework of network based analysis in web service composition. This work takes care of the issues pertaining to inputs and outputs matching of web services and Quality-of-Service (QoS) at the same time. Six multi-criteria programming models are explored to select the desirable service composition in a variety of categories in accordance with customers' preferences in three different scenarios: (1)Optimal, (2)Compromise optimal, and (3)Acceptable. This set of multi-criteria models have both advantages and disadvantages compared with each other, and can be used as different solvers in the network based service composition framework. The proposed regular multi-criteria programming (MCP) models is used in Scenario (1): Optimal. The proposed multi-criteria goal programming for optimal composition (MCGPO) and multi-criteria goal programming for non-optimal solution (MCGPN) models are designed for Scenarios (2):Compromise optimal and (3): Acceptable respectively. And they can find a compromise composition based on the trade-off of

customer's preference on the QoS goals in case that the optimal composition satisfying both functional and QoS constraints does not exist in the network.

6.2 Prerequisite and Framework

It is assumed that web services are all based on the standard WSDL (Web Services Description Language), and exchange information interactively via SOAP (Service Oriented Architecture Protocol)(W3C 2003). To solve the multi-criteria web service composition problem, three different multi-criteria goal programming models are designed to generate customized solution. For each of the three models, we can solve them either by preemptive method proposed by Wadhwa et al (2007) or non-preemptive method. This models proposed in this chapter not only is able to handle the functional aspects of service composition issue in general, but also has the ability to tailor the service in accordance with the preferences of the customers in terms of QoS attributes.

It is assumed that the services defined here are the smallest units of operations. i.e., a single service cannot be subdivided any further. Services are stored hierarchically in the UDDI registry center, and are categorized by a service interface in WSDL documents that contains the types, PortType, operations and binding elements. A web service can have multiple inputs as well as outputs. Singh and Huhns (2005)introduced varieties of topics on service computing including standards, enterprise architectures, semantics, social and economical service selection and security.

The network of web services can be built based on the WSDL and OWL ontology. Cui et al. (2009) proposed a service composition framework based on large scale networks as shown in Figure 6-1(a) and Figure 6-1(b). The nodes represent web services.

There is a link between two web services if they share some common input or output attributes. Communities can help in focusing a query on a serch region.

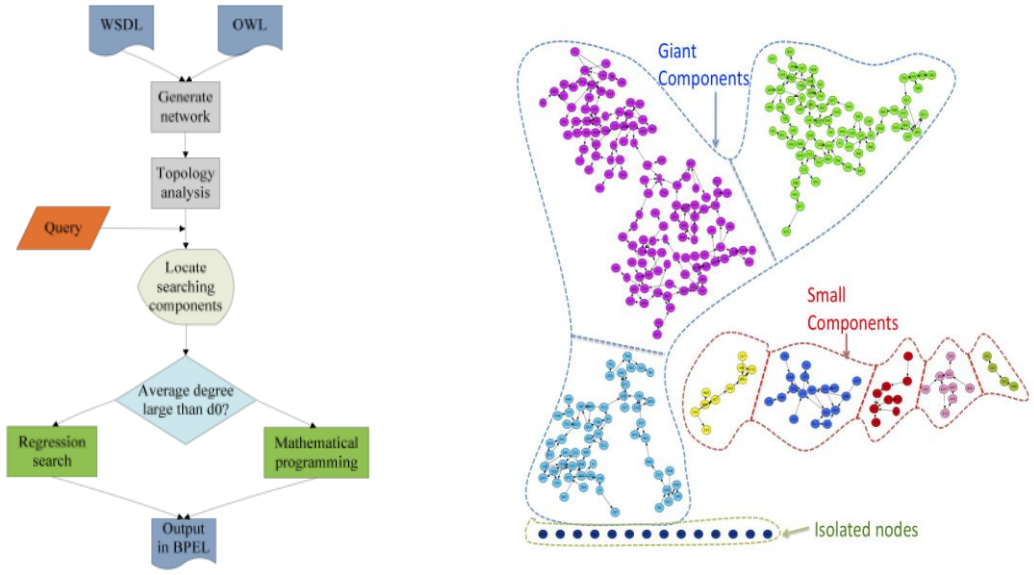


Figure 6-1 Composition process: (a) Composition framework, (b) An example of service networks

Quality of Services (QoS) metric is used to evaluate services. The cost, service execution time, reliability etc. are considered as QoS elements. The model presented in this chapter can handle many number of QoS metric elements. Cost, execution time and reliability are used to demonstrate the model. Hence, the QoS vector for a service can be expressed as: "quality of service(*s*) = f(cost(*s*), service execution time(*s*), reliability(*s*))." Cost is the expense for purchasing a service or services; service execution time is the process time of a service or services; reliability is a measure of successfully running a service or a set of services.

The mathematical programming methods — namely, MCP (Multi-criteria Programming), MCGPO (Multi-criteria Goal Programming model for Optimal

composition) and MCGPN (Multi-criteria Goal Programming model for Non-optimal composition) ─ is designed to accommodate service composition problems with flexible functional and QoS criteria.

6.3 Mathematical Modeling

Web service composition problem is modeled as a multi-criteria program. Especially, goal programming solves the problem in accordance with the customers' preferences and can find a compromise solution when the regular multi-criteria programming model does not have a solution. Customers' requirements on aggregating price, reliability of service, and total service time are considered as goals. Both preemptive and non-preemptive goal programming models are built and tested.

6.3.1 Attributes of Services

In this section, we consider a service as the smallest unit of operation. i.e., an operation can be treated as a single service; therefor services include individual services and operations in the context. Services are stored hierarchically in the UDDI registry center, and are categorized by a service interface in WSDL documents that contains the types, PortType, operations and binding elements. A service interface document can reference another service interface document using an import element. The service interface document is developed and published by the service interface provider.

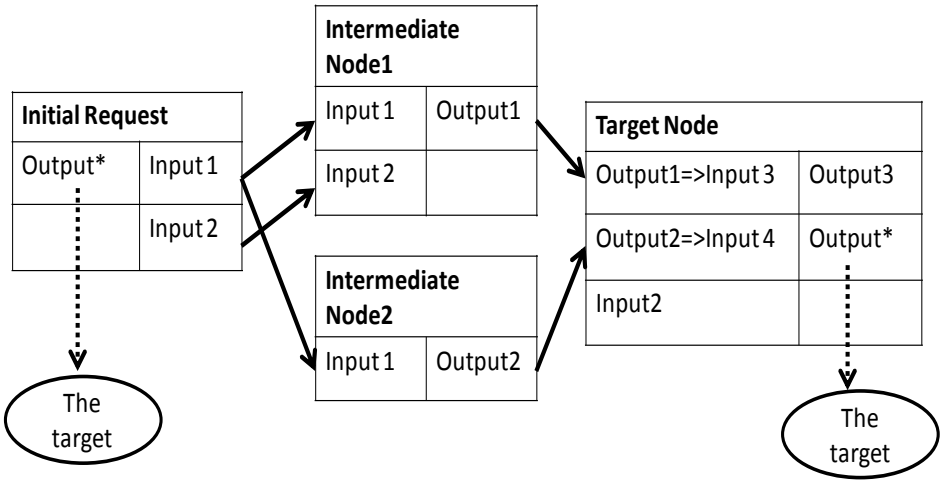


Figure 6-2 An example of service composition

Service implementation document is created and published by the service provider. The roles of the service interface provider and service provider can either be logically separable, or be the same business entity. Each service can contain single or multiple inputs and outputs, as shown in Figure 6-2.

A web service can have multiple inputs as well as outputs. When the service needs to be activated, all of its inputs need to be available. In the meantime, some extra outputs that are not useful for the future composition may be generated as by-products when the service is initiated. After predetermining the maximum number of iterations, the searching procedure can be modeled by a linear program.

6.3.2 Definition of Parameters and Variables of the Model

Let us define the entire list of variables and parameters in this section. Quality of Services (QoS) metric is used to evaluate services. In the context, cost, execution time and reliability are used to demonstrate the model. Hence, the QoS vector for a service can be expressed as: "quality of service(s) = f(cost(s), service execution time(s),

reliability(s))." Cost is the expense for purchasing a service or services; service execution time is the process time of a service or services; reliability is a measure of successfully running a service or a set of services. The entire list of variables used is shown in Table 6-1.

Table 6-1 Definition of variables and parameters

Variable	Definition
Z	a set of web services
I	a set of input attributes of the web services
O	a set of output attributes of the web services
m	the number of services in Z
n	the number of attributes for the services in set Z
L	the maximal number of composition levels
Z_{lj}	web service that is currently available in the database; $Z_{lj} \in Z; j = 1, 2, \ldots, m; l = 1, 2, \ldots, L$
I_{ij}	the i^{th} input attribute of service $Z_j; i = 1, \ldots, n; j = 1, 2, \ldots, m$
O_{ij}	the i^{th} output attribute of service $Z_j; i = 1, \ldots, n; j = 1, 2, \ldots, m$
p_j	the fixed price for acquiring the service from $Z_j; j = 1, 2, \ldots, m$
t_j	the execution time of service $Z_j; j = 1, 2, \ldots, m$
f_j	the failure rate of service $Z_j; j = 1, 2, \ldots, m$
q_j	the reliability of service $Z_j; j = 1, 2, \ldots, m$
C_0	the maximum total cost that the customer is willing to pay for the services
T_0	the maximal total execution time that the customer allows to accomplish the entire process of services
Q_0	the minimal reliability that the customer allows for a service in the composition
Q_1	the minimal overall reliability that the customer allows for the entire service complex, where $Q_1 > Q_0$

In general, the number of attributes in the input set I and the number of attributes in the output set O are different. However, it is reasonable to let $n = max\{h, k\}$ be the total number of attributes since most of the attributes are the inputs of some services and the outputs of other services at the same time. Generally speaking, all the

attributes can be inputs in some services and outputs in other services. Thus, it can be proved that $I = O$ approximately, in large scale service networks. In order to define the reliability score for web services, the following definition is given. The reliability of a service is a function of failure rate of the service.

If the failure rate of service Z is f, the reliability score of the service is defined as: $q(f) = -log(f)$, where $0 < f \leq 1$ and $0 \leq q(f) < +\infty$.

Here, it is noted that reliability measure $q(f)$ can be expressed in terms of the failure rate f. This technique is useful to convert the nonlinear objective function (7) into linear objective function (4), which simplifies the problem. LP(linear programming) solvers can be used to solve the model. Next, we need to specify a depth level of composition before using this mathematical programming model. The decision about L, the depth level, is important as larger or smaller L influences the computational time and whether the optimal solution can be obtained or not.

Among all the variables defined, the decision variables are Z_{lj}, the status of the j^{th} web service in the l^{th} level of composition, $j = 1, 2, \ldots, m$; $l = 1, 2, \ldots, L$.

$$Z_{ij} = \begin{cases} 1 & \text{web service } Z_j \text{ is selected in the } l^{th} \text{ level} \\ 0 & \text{otherwise} \end{cases}$$

$$j = 1, 2, \ldots, m; \quad l = 1, 2, \ldots, L$$

6.3.3 Objective Function

The objective function is defined as follows:

Cost (criterion No.1): the cost of the service composition equals to the sum of the prices of the services in the composition.

$$\min \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot p_j \qquad 6\text{-}5$$

Service execution time (criterion No.2): Service execution time is the total processing time for executing the entire series of services. We assume that the services at one level are executed in parallel.

The maximum execution time of the services in the 1^{st} level is $\max_j \{t_j \cdot Z_{1j}\}$;

The maximum execution time of the services in the 2^{nd} level is $\max_j \{t_j \cdot Z_{2j}\}$;

The maximum execution time of the services in l^{th} level is $\max_j \{t_j \cdot Z_{lj}\}$;

So, the total service execution time of this composition is: $\sum_{l=1}^{L} \max_j \{t_j \cdot Z_{lj}\}$;

Let η_l be the maximum service execution time of the l^{th} level. The above total service execution time expression can be reformulated in terms of the following linear program:

$$\min \sum_{l=1}^{L} \eta_l \qquad 6\text{-}6$$

subject to

$$\eta_l - t_j * Z_{lj} \geq 0 \qquad 6\text{-}7$$

$$j = 1, 2, \ldots, m;\ l = 1, 2, \ldots, L.$$

Reliability (criterion No.3): Reliability of the service composition is described by the summation of the reliability scores of all the services included in the composition.

$$\max \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot q_j \qquad 6\text{-}8$$

The validity of the first and the second criteria are obvious; however, we need to validate the third criteria according to definition 1. It can be proved that criterion No.3 is valid.

Let S be the set of services appearing in a composition, $S \subseteq Z$. Then, expression (4) equals to

$$\max \sum_{Z_j \in S} q_j \qquad 6\text{-}9$$

$$\sum_{Z_j \in S} q_j = -\sum_{Z_j \in S} (\log f_j) = -\log(\prod_{Z_j \in S} f_j) \qquad 6\text{-}10$$

Since function $y = -\log(x)$ monotonically decreases in x when $x \in (0, +\infty)$

Now we have: (6-9) equals to

$$\min \prod_{Z_j \in S} f_j \qquad 6\text{-}11$$

where $\prod_{Z_j \in S} f_j$ is the overall failure rate of the composition.

Thus, (6-9) equals to (6-11).

So, maximizing the summation of the reliability scores of the services in the composition equals to minimizing the product of the failure rates of the services in the composition. So, (6-9) is validated.

6.3.4 Constraints

1. Input constraints: An input attribute of the query service should be included in the input attributes of the selected services in the composition. Thus,

$$\sum_{l=1}^{L} \sum_{j=1}^{m} I_{ij} \cdot Z_{lj} \geq I_{i0} \qquad i = 1, 2, \ldots, n. \qquad 6\text{-}12$$

This constraint can be neglected, if we allow some redundancy in the inputs provided by customers.

2. Output constraints: An output attribute of the query should be included in the output attributes of the selected services in the composition. Hence,

$$\sum_{l=1}^{L}\sum_{j=1}^{m} O_{ij} \cdot Z_{lj} \geq O_{i0} - I_{i0} \qquad i = 1, 2, \ldots, n. \tag{6-13}$$

3. The relationship of the outputs and inputs between the levels has to satisfy the following requirements.

All the inputs of the selected services in the first level must be a subset of the initial input set given in the query.

$$\sum_{j=1}^{m} I_{ij} \cdot Z_{1j} \leq I_{i0} \qquad i = 1, 2, \ldots, n. \tag{6-14}$$

Also, all the input sets of selected services at the k^{th} level must be a subset of the union of the initial input set given in the query and the output sets of services in previous levels. The formulation is:

$$\sum_{j=1}^{m} I_{ij} \cdot Z_{k+1,j} - \sum_{l=1}^{k}\sum_{j=1}^{m} O_{ij} \cdot Z_{lj} \leq I_{i0} \tag{6-15}$$

$$k = 1, 2, \ldots, L-1;\ i = 1, 2, \ldots, n.$$

The relation among the inputs of services in k^{th} level and the outputs from the previous levels and the attributes given in the query needs to satisfy equation (6-15).

4. Goal constraint on the total cost: The customer hopes that the total cost should not exceed C_0.

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j \le C_0 \qquad 6\text{-}16$$

5. Constraint on the service execution time: The customer hopes that the total service execution time should not exceed T_0. Since some services can be executed in parallel, we take the longest execution time as the execution time of the set of services executed in parallel. The execution time of the composition, *e.g*. total service execution time is the sum of the service execution times of L levels. Thus,

$$\eta_l \ge Z_{lj} t_j, \qquad l = 1, 2, \ldots, L; \qquad 6\text{-}17$$

$$\sum_{l=1}^{L} \eta_l \le T_0 \qquad 6\text{-}18$$

6. Constraint on reliability: The reliability of each service has to be equal to or better than a certain specified level, i.e.,

$$Z_{lj} \cdot (q_j - Q_0) \ge 0, \quad l = 1,2,\ldots,L;\ j = 1,2,\ldots,m; \qquad 6\text{-}19$$

7. Constraint on total reliability of service composition: The total reliability of the service composition should be equal to or greater than a certain level Q_1.

$$\sum_{l=1}^{L} Z_{lj} \cdot q_j \ge Q_1 \qquad 6\text{-}20$$

8. Non negative and binary constraints:

$$Z_{lj} \ge 0 \qquad Z_{lj} \in \{0,1\} \qquad 6\text{-}21$$

$$\eta_l \ge 0 \qquad 6\text{-}22$$

$$l = 1, 2, \ldots, L;\ j = 1, 2, \ldots, m;$$

The input-output constraints (6-12), (6-13), (6-14) and (6-15) can handle both sequential and parallel flow patterns in service network.

Three multi-criteria scenarios are formulated, wherein the customers require: 1. Optimal solution, 2. Optimal solution under a possible compromise of the QoS, and 3. an acceptable solution with both functional and nonfunctional attributes considered.

6.3.5 Multi-Criteria Programming with Pure Real Constraints (MCP)

Based on the formulations in previous sections, the following multi-criteria programming model with pure real constraints can be defined:

$$\min Z_1 = \sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j$$

$$\min Z_2 = \sum_{l=1}^{L} \eta_l$$

$$\max Z_3 = \sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j$$

subject to the constraints (6-12) through (6-22), *e.g.* subject to

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j \le C_0$$

$$\sum_{l=1}^{L} \eta_l \le T_0$$

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j \ge Q_1$$

$$\sum_{l=1}^{L}\sum_{j=1}^{m} I_{ij} \cdot Z_{lj} \ge I_{i0} \quad i = 1, 2, \ldots, n$$

$$\sum_{l=1}^{L}\sum_{j=1}^{m} O_{ij} \cdot Z_{lj} \ge O_{i0} - I_{i0} \quad i = 1, 2, \ldots, n$$

$$\sum_{j=1}^{m} I_{ij} \cdot Z_{1j} \le I_{i0} \quad i = 1, 2, \ldots, n$$

$$\sum_{j=1}^{m} I_{ij} \cdot Z_{k+1,j} - \sum_{l=1}^{k}\sum_{j=1}^{m} O_{ij} \cdot Z_{lj} \le I_{i0}$$

$$k = 1, 2, \ldots, L-1; \quad i = 1, 2, \ldots, n$$

$$Z_{lj} \cdot (q_j - Q_0) \ge 0$$

$$l = 1, 2, \ldots, L; \quad j = 1, 2, \ldots, m$$

$$\eta_l - t_j \cdot Z_{lj} \ge 0$$

$$j = 1, 2, \ldots, m; \quad l = 1, 2, \ldots, L$$

$$Z_{lj} \ge 0 \quad Z_{lj} \in \{0,1\}$$

$$l = 1, 2, \ldots, L; \quad j = 1, 2, \ldots, m$$

$$\eta_l \ge 0 \quad l = 1, 2, \ldots, L$$

This multi-criteria programming model can be solved either by the preemptive method or by the non-preemptive method (weighted average method). If the customer of the query gives the priority of the objectives in order. For instance, if $min\ Z_1$ has the highest priority (denote it P_1), $min\ Z_2$ has the second highest priority (denote it P_2), and $min\ Z_3$ has the least priority (denote it P_3), the model can be solved by solving three mathematical programming model sequentially (see (Arthur et al. 1980)). This is called preemptive method. We call the preemptive model with pure real constraints as **Model 1**.

If the customer of the query gives the weights to the objectives. For instance, if the weights of Z_1, Z_2 and Z_3 are W_1, W_2 and W_3 respectively, the above model can be solved by solving the mathematical programming model with objective *min* $W_1 \cdot Z_1 + W_2 \cdot Z_2 - W_3 \cdot Z_3$. Let us call the non-preemptive model with pure real constraints as **Model 2**.

The advantage of MCP models (Model 1 and Model 2) is that they find the service composition of the query which satisfies both the functional requirements (starting from the inputs given in the query, it finds the composition to give the outputs requested in the query), and the nonfunctional requirements (also called QoS requirements: for example cost, reliability and execution time). The disadvantage of this model is that it won't give a compromise solution if there is not a composition satisfying both functional and nonfunctional requirements. In general, the compromise solution is informative and useful to the customer, so let us revise some of the constraints into goal constraints. This distinguish our work from the existing literature.

6.3.6 Multi-criteria Goal Programming for Optimal Solutions (MCGPO)

In order to provide the customer a compromise solution when there is no composition that can satisfy both functional and nonfunctional requirements of the customer, we can revise some of the real constraints into goal constraints. Given these goals, our objective is to achieve them as best as we can. Let us revise the cost constraint $\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j \le C_0$ into $min \quad d_1^+, s.t. \sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j + d_1^- - d_1^+ = C_0$, and $d_1^-, d_1^+ \ge 0$; revise the execution time constraint $\sum_{l=1}^{L} \eta_l \le T_0$ into $min \quad d_2^+$, s.t. $\sum_{l=1}^{L} \eta_l + d_2^- - d_2^+ = T_0$,

and $d_2^-, d_2^+ \geq 0$; and revise the reliability constraint $\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j \geq Q_1$ into $min \ \ d_3^-$, s.t. $\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j + d_3^- - d_3^+ = Q_1$, and $d_3^-, d_3^+ \geq 0$.

The goal programming model is therefore as follows:

$$\min Z_1 = \sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j$$
$$\min Z_2 = \sum_{l=1}^{L} \eta_l$$
$$\max Z_3 = \sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j$$
$$\min Z_4 = d_1^+$$
$$\min Z_5 = d_2^+$$
$$\min Z_6 = d_3^-$$

subject to the goal constraints:

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j + d_1^- - d_1^+ = C_0$$
$$\sum_{l=1}^{L} \eta_l + d_2^- - d_2^+ = T_0$$
$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j + d_3^- - d_3^+ = Q_1$$
$$d_i^-, d_i^+ \geq 0 \quad i = 1, 2, 3$$

and the real constraints (6-12), (6-13), (6-14), (6-15), (6-17), (6-19), (6-21) and (6-22).

This multi-criteria programming model can be solved either by preemptive method or non-preemptive method (weighted average method), as in the previous case.

If the customer prioritizes the objectives in a descending order $Z_1, \ldots, Z_6$, the model can be solved by solving six goal programming models sequentially (see (Arthur

et al. 1980)). For example, solving the goal programming model with objective Z_1, add the real constraints corresponding to this goal and continue to solve the goal programming with the second goal Z_2; Continue the procedure until all the goals are calculated in order. Let us use P_j, $j = 1, \ldots, 6$ to denote the priority of $Z_1, \ldots, Z_6$. This is called preemptive goal programming method. Let us call the preemptive model with goal constraints as **Model 3**.

If the customer of the query gives the weights to the objectives. For instance, if the weights of Z_1, Z_2, Z_3, Z_4, Z_5 and Z_6 are W_1, W_2, W_3, W_4, W_5 and W_6 respectively, the above model can be solved by solving the mathematical programming model with the single objective $min \;\; W_1 \cdot Z_1 + W_2 \cdot Z_2 - W_3 \cdot Z_3 + W_4 \cdot Z_4 + W_5 \cdot Z_5 + W_6 \cdot Z_6$. Let us call the non-preemptive model with goal constraints as **Model 4**.

Next, let us study the difference between objectives and goals, and then the steps for building web service composition through goal programming.

An objective is to either minimize or maximize a measurable metric. A goal is the preference to drive a measurable metric either below or above a specific targeted value. It may be either achievable or not achievable.

The steps for building the web service composition problem through goal programming can be stated as follows:

Step-1: Get the query from customer. From the query, the real constraints can be formulated. Step-2: Obtain the objectives from the customer. This is to formulate the objectives in the multi-criteria model. Step-3: Get the goals from the customer. These goals help to build the goal constraints and the goal objectives. The customer can set up

many number of goals from the real constraints of QoS. Then we will try to achieve as many goals as we can. Goals are not constraints and, in the final analysis, some of them may not be achieved. Suppose, for instance, if the user has three goals of C_0 for the total cost; T_0 for the service execution time, and Q_1 for the total reliability. Then, the goal constraints can be written as:

$$\begin{aligned} &\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j + d_1^- - d_1^+ = C_0 \\ &\sum_{l=1}^{L} \eta_l + d_2^- - d_2^+ = T_0 \\ &\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j + d_3^- - d_3^+ = Q_1 \end{aligned} \qquad \text{6-23}$$

Step4: Obtain the customer's priority of the objectives and goals for the preemptive method, or the weights of the objectives and goals for non-preemptive method .

Step-5: Solve the goal programming model formulated in Step-1 through Step-4. If it is preemptive method (Model 3), we need to solve the single objective mathematical programming sequentially (see (Arthur et al. 1980)), *e.g.* goals at high priority have to be satisfied before those with low priorities predefined by the customer. If it is non-preemptive method (Model 4), we only need to solve a mathematical programming model with 1 single objective.

The advantage of the goal programming models (Model 3 and Model 4) is that they can give a compromise solution if there is not a composition satisfying both

functional and nonfunctional requirements. In order to reduce the computational time, let us further relax the objectives in the goal programming formulation.

6.3.7 Multi-criteria Goal Programming for Non-optimal Composition (MCGPN)

MCP model finds the optimal composition that satisfies both functional and nonfunctional (QoS) constraints. MCGPO can find the optimal composition that satisfy functional and nonfunctional constraints if it exists and finds the compromise composition that satisfies the functional constraints and part of the nonfunctional (QoS) constraints if the composition satisfying MCP model does not exist. We can further relax the objectives so as to find an acceptable composition that satisfies functional constraints and all or part of nonfunctional constraints(QoS). This model is useful when the customer does not require the high Quality of Service. In a large scale problem, this can be used in near real time composition environment. We can find an acceptable solution by just keeping the goal constraints, and removing the three real objectives from MCGPO model. The MCGPN model can be formulated as follows:

$$\begin{aligned} &\min Z_4 = d_1^+ \\ &\min Z_5 = d_2^+ \\ &\min Z_6 = d_3^- \end{aligned} \qquad \text{6-24}$$

subject to

The same set of constraints as in Model 3 and Model 4.

This multi-criteria programming model can be solved either by preemptive method or non-preemptive method (weighted average method). If the customer of the query gives the priority to the goals in order. For instance, if Z_4, Z_5 and Z_6 are in the decreasing order of priority for the user, as in the previous case, let us denote the priority

of Z_4, Z_5 and Z_6 as P_1, P_2 and P_3 respectively. in decreasing order of priority. This model can be solved in a similar manner as Model 3 (see (Arthur et al. 1980)). Let us call this preemptive model with only goal objectives **Model 5**. For example, in the following formulation, the highest priority is placed on the total cost; the second priority is placed on total execution time; and the least priority is put on reliability. Then the web service composition problem can be formulated as a preemptive goal programming:

$$\min Z = P_1 \cdot d_1^+ + P_2 \cdot d_2^+ + P_3 \cdot d_3^- \qquad 6\text{-}25$$

where notations P_1, P_2 and P_3 represent the priority order of $Z_4, Z_5 and Z_6$ respectively.

If the customer gives the weights to the goals, the problem can be formulated as non-preemptive goal model:

$$\min Z = w_1 \cdot d_1^+ + w_2 \cdot d_2^+ + w_3 \cdot d_3^- \qquad 6\text{-}26$$

Let us call this model as **Model 6**.

It should be noted that, before using non-preemptive goal programming, the costs, the execution times and the reliability have to be scaled properly; otherwise, the criterion with large raw value will dominate the remaining.

6.4 Scenario Analysis of the Models

Next, let us discuss the worst case and the best case scenarios. As shown in Figure 6-3(a), when $\Omega = F$, the mathematical programming model is at the worst case, and behaves similar to exhaustive search. When $\Omega \supset F \supset N$ and N contains only one solution, the mathematical goal programming is at the best case, it finds the optimal solution really fast.

6.4.1 Complexity Analysis

In this section, we analyze the time complexity of multi-criteria mathematical programming model. Here we use the total comparison times before reaching the optimal solution to compute the time complexity.

a) Time Complexity (TC):

In order to reach the optimum solution quickly, we may run the ceiling test, infeasibility test, cancelation zero test and cancelation one test for the Mixed IP Model (See chapter 9 in (Salkin 1989)). Under the best conditions, the infeasible solutions can be discarded before running the actual search in IP solver, so the optimum solution can be found right away under the best condition. Under the worst condition, no variable can be eliminated from the free variable set via these tests, and the time complexity of the IP solver would be the same as that of the exhaustive search. Let us suppose that different solution cases occur with the same probability. If the size of web service set Z is m, and the maximum number of levels of composition is L, there will be $L \cdot m$ zero-one variables, and L nonnegative variables in the Mixed Integer Programming Model. The m nonnegative variables are introduced for the sake of comparing the maximal service execution time of each level. $C_v^i * 2^{v-i}$ is the total checking times of the binary variables when i variables out of v are eliminated, which means $(v-i)$ variables need to be checked whether each of them is 0 or 1. There may be 0 variables eliminated, 1 variable eliminated, 2 variables eliminated, ..., or all variables eliminated. $(C_v^0 + C_v^1 + C_v^2 + \cdots + C_v^v)$ is the total number of cases that includes all the above circumstances.

Lemma 6-1 *The time complexity of the weighted average Mixed IP Model is:*

$$\frac{C_v^0 2^v + C_v^1 2^{v-1} + \cdots + C_v^v 2^0}{C_v^0 + C_v^1 + \cdots + C_v^v} = 1.5^v \qquad 6\text{-}27$$

where v is the total number of zero-one variables, and $V = L \cdot m$.

Table 6-2 shows the times of comparisons for the goal programming and exhaustive search. Table 6-3 gives an example when the number of services $m = 10$ and the number of levels $L = 5$, and thus $v = 50$. In the formulation of web service composition, the non-preemptive goal programming is a mixed integer linear programming (MILP) with weighted goals. And, at each step, the preemptive goal programming is a mixed integer linear programming (MILP).

Table 6-2 Computational complexity of MILP

Item	**Best TC**	**Average TC**	**Worst TC**
Mixed IP (1)	1	1.5^v	2^v
Exhaustive Search (2)	2^v	2^v	2^v
TC Ratio $\frac{(1)}{(2)}$	$\frac{1}{2^v}$	$(\frac{3}{4})^v$	1

Table 6-3 Computational complexity of MILP with $M = 10$ and $L = 5$

Item	**Best TC**	**Average TC**	**Worst TC**
Mixed IP (1)	1	1.5^{50}	2^{50}
Exhaustive Search (2)	2^{50}	2^{50}	2^{50}
TC Ratio $\frac{(1)}{(2)}$	4.44×10^{-14}	5.67×10^{-7}	1

b) Example:

A comparison of the time complexity of the mixed IP model and exhaustive search is shown for the case where the total number of web services online is ten, that is $m = 10$, and the customer allows the maximum composition level $L = 5$.

Under most conditions, only a few services are related to the request of the customer in a large web service pool. Therefore, a large number of unrelated services can be eliminated during the initial step. The theoretical comparison times in the above table are actually the upper bounds for the real problems considered. Since the computational complexity is highly dependent on the correlation between the services, and the correlation matrix of the services is sparse for large scale problems, the real time complexity of the service composition programming is not likely to grow exponentially. Thus, the worst condition shown in the table will seldom ever appear in a real web service composition scenario.

From Table 6-2, we can see that the Mixed IP model can improve the average computational efficiency significantly as compared to that of the exhaustive search for the optimal solution.

6.4.2 Solution Analysis of the Mathematical Programming Models

In Section 4.5, six different composition models (three categories) are introduced: (1) preemptive MCP model (Model 1), (2) non-preemptive MCP model (Model 2), (3) preemptive MCGPO model (Model 3), (4) non-preemptive MCGPO model (Model 4), (5) preemptive MCGPN model (model 5) and (6) non-preemptive MCGPN model (model 6). We notice that there are two categories of multi-criteria mathematical programming

models: One is preemptive method, and the other is non-preemptive method. The following relation between the two methods can be proven:

Theorem 6-1 Relation of the mathematical models

Model 2 will be approximately equal to Model 1, Model 4 will be approximately equal to Model 3 and Model 5 will be approximately equal to Model 6, if and only if $w_1 >> w_2 >> w_3$, where $a >> b$: a is infinitely larger than b.

Next, let us discuss the solution of the multi-criteria mathematical programming models of web service composition.

Figure 6-3(a) shows the feasible region of MCP model, where Ω represents the entire space, F is the functional feasible space and N is the nonfunctional feasible space. Similarly, Figure 6-3(b) shows the feasible region of MCGPO and MCGPN models. We notice that MCGPO and MCGPN models have the same feasible region which is F, and MCP model has a smaller feasible region which is the intersection of F and N. Figure 6-3 shows the relation amongst the solution spaces of the MCP, MCGPO and MCGPN models. the solution space of MCGPN is the largest one, and it can equal to F. The solution space of MCGPO is a subset of the solution space of MCGPN. The solution space of MCP is the smallest, and it is the subset of the solution space of MCGPO. In Figure 6-4, the quality of the solutions from the MCP, MCGPO and MCGPN models are explored. The details follow:

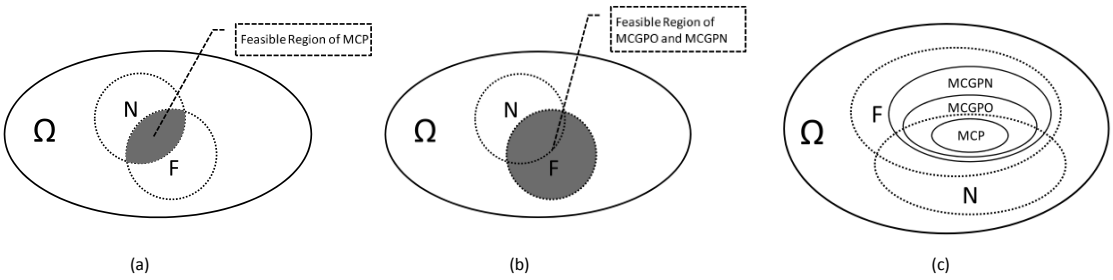


Figure 6-3 Feasible region and solution space

(a) Feasible region of MCP model, (b) Feasible region of MCGPO and MCGPN models, (c) Solution spaces of MCP, MCGPO and MCGPN models

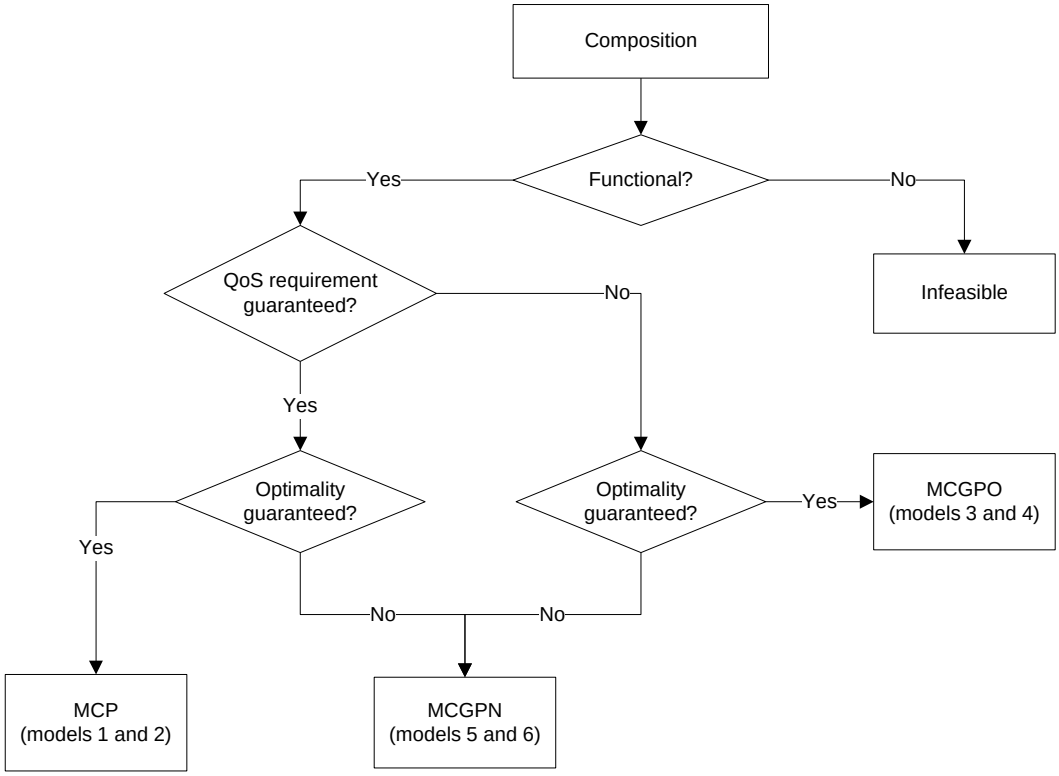


Figure 6-4 Solution quality of MCP, MCGPO and MCGPN models

1. The MCGPN, MCGPO and MCP models will not yield a solution if there does not exist any work-flow that can satisfy the customer's functional requirement since the functional constraints are real constraints which determines the feasible region;

2. The MCP models will generate the optimal solution if there is a solution that can satisfy the customer's functional requirements and nonfunctional requirements. In this case, goals are not considered. MCP model's solution space is in the intersection of F and N ;

3. The MCGPO model will generate an optimal solution that can meet as many goals as possible, if there are multiple solutions that can meet the customer's functional requirement, but no solution can meet all the nonfunctional requirements at the same time. This allows the model to choose a trade-off among the objectives according to the customers' preferences, if not all of them can be achieved simultaneously, in this case, the MCP model does not have a solution. MCGPO model's solution is optimal, but the QoS constraint may be violated.

4. The MCGPN model will generate an acceptable solution that satisfies the functional requirements of the query. The model tries its best to achieve the goals of nonfunctional constraints,but the optimality is not guaranteed; Table 6-4 Characterization of the modelssummarizes the characteristics of the proposed three types of models in the chapter.

Table 6-4 Characterization of the models

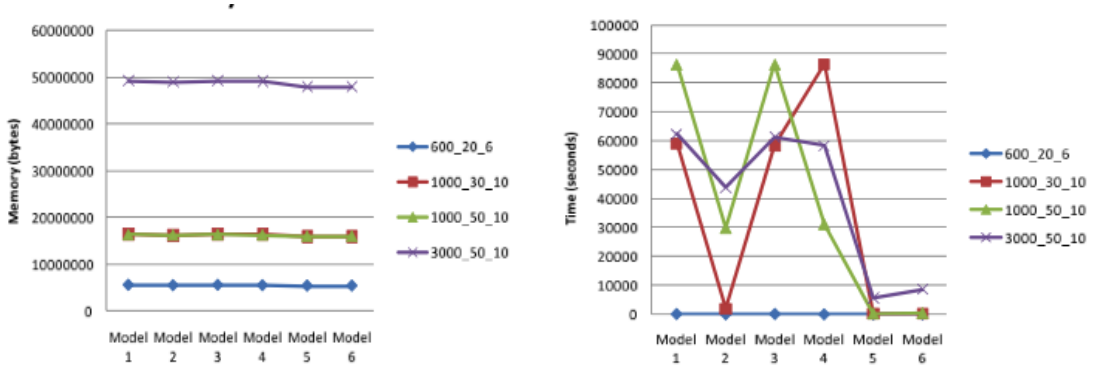
Model	**Category**	**Solution method**	**Optimality**	**QoS preference guaranteed**	**Compromise**
Model (1)	MCP	preemptive	Yes	Yes	No
Model (2)	MCP	nonpreemptive	Yes	Yes	No
Model (3)	MCGPO	preemptive	Yes	No	Yes
Model (4)	MCGPO	nonpreemptive	Yes	No	Yes

Model (5)	MCGPN	preemptive	No	No	Yes
Model (6)	MCGPN	nonpreemptive	No	No	Yes

6.5 Experiments

The models were solved using C-PLEX solver, and the experiments were carried out on a server in the High Performance Computing Center at Pennsylvania State University. The configuration of the computer is: Dell PowerEdge 1950 1U Rackmount Server , Dual 3.0 GHz Intel Xeon 3160 Dual-Core Processors, 16 GB of ECC RAM. The experiment E1 has 600 web services with possible 20 different attributes. E1 allows a maximum search levels of 6. And there are 15 runs in experiment E1. Similarly, we can define experiments E2, E3 and E4. The first set of experiment E1 has 600 services, 30 attributes and the maximum searching levels are 6; the second set of experiment E2 has 1000 services, 30 attributes and the maximum searching levels are 10; the third set of experiment E3 has 1000 services, 50 attributes and the maximum searching levels are 10; the fourth set of experiment E4 has 3000 services, 50 attributes and the maximum searching levels are 10. In all the experiments, the run was stopped if it could not finish in 24 hours according to the policy of the HPC at Penn State. If a run was terminated, the maximum memory use of all the models in all the experiments is the memory use for this run; and use the maximum computational time of all the models in all the experiments as the computational time for this run. The problem sizes described above are the selected searching component sizes in a much larger scale network.

Figure 6-5(a) shows the memory use of the six models in four different designs of experiment. Some of the curves almost overlapped when the values are close to each other. From the four experiments, we can see that Models 4 and 5 always use less memory than the other models; and Models 3 and 1 always use more memory than the other models.



(a) Memory vs. Models

(b) Time vs. Models

Figure 6-5 Experiments

We assume that the customers give both priorities and the weights to the objectives. So, both preemptive and non-preemptive models can be formulated and tested. For the 6 models, the average memory usage on each experiment is plotted in Figure 6-5(a). We can see that the memory usage of preemptive models is around 2% percent lower than the memory usage of the non-preemptive models. However, the difference is relatively small when the problem size is small. In fact, the memory use of a service composition is mainly determined by the problem size (e.g. the number of variables) rather than which model we chose among Models 1, 2, 3, 4, 5 and 6. From Figure 6-5(a) it can be seen that the memory size increases in problem size (number of services and

number of attributes). For a given problem size, memory usage is virtually no big difference among the six models.

Figure 6-5(b) shows the computational time of the six models in four different designs of experiment.Some curves may overlap when their values are close to each other. From the four experiments, we can see that Models 4 and 5 always take less computational time than the other models do; and Models 3 and 1 always take more computational time than the other models do.

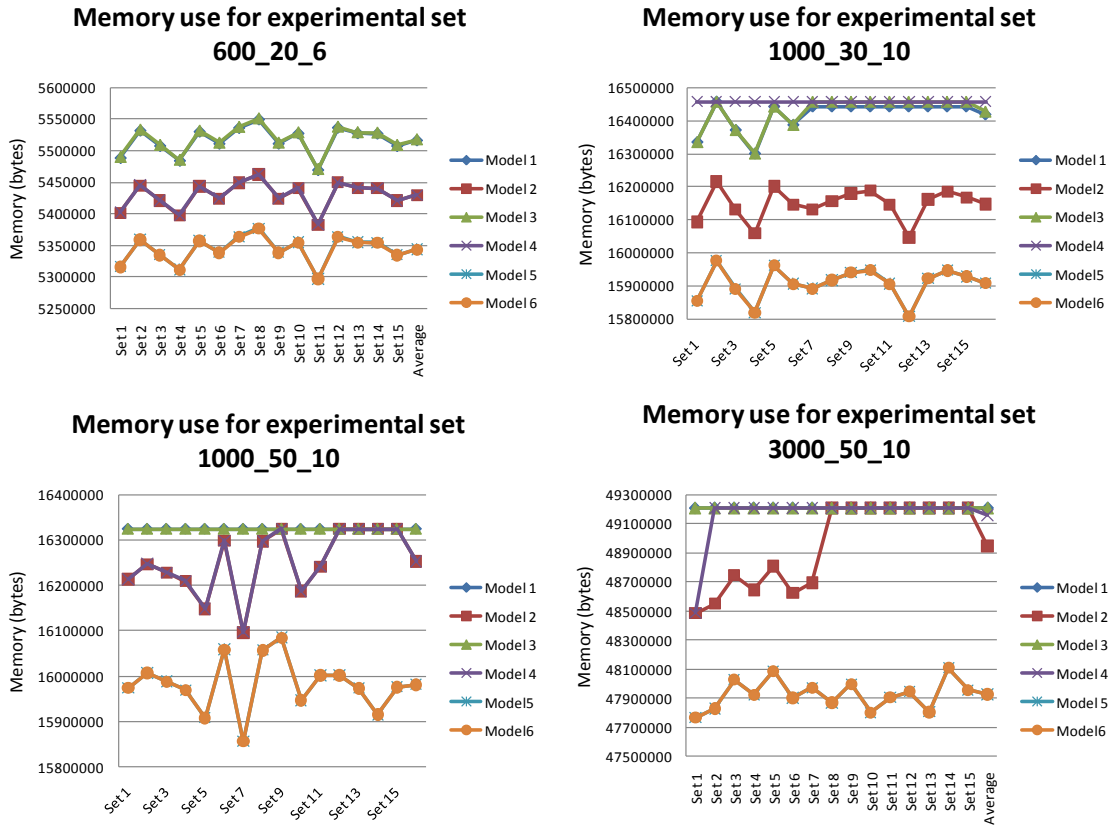


Figure 6-6 Memory usage performance of the six models in four different experimental sets

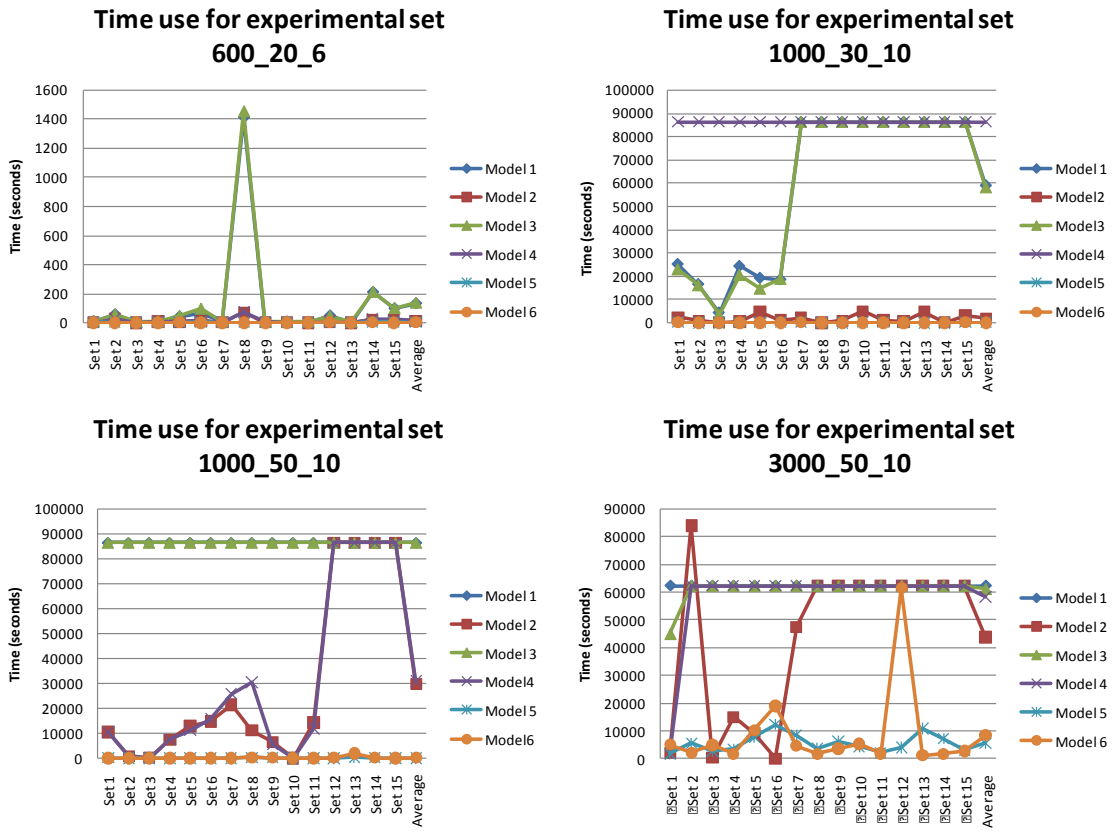


Figure 6-7 Computational time performance of the six models in four different experimental sets

For the six models, the average computational time on each experiment is plotted in Figure 6-7. It can be seen that the computational time of each model on each experiment fluctuates. Model 5 and Model 6 always have less computational time for all the experiments compared with Models 1, 2, 3, and 4. This is because Model 5, and 6 find the acceptable solution while Models 1, 2,3 and 4 find the optimal solution. Amongst Models 1, 2, 3 and 4, Models model 1 and 2 return without a solution is there is no such a solution satisfying all the functional and QoS requirements, while Models 3 and 4 still return with acceptable solution if there is a solution which is feasible but does not satisfy

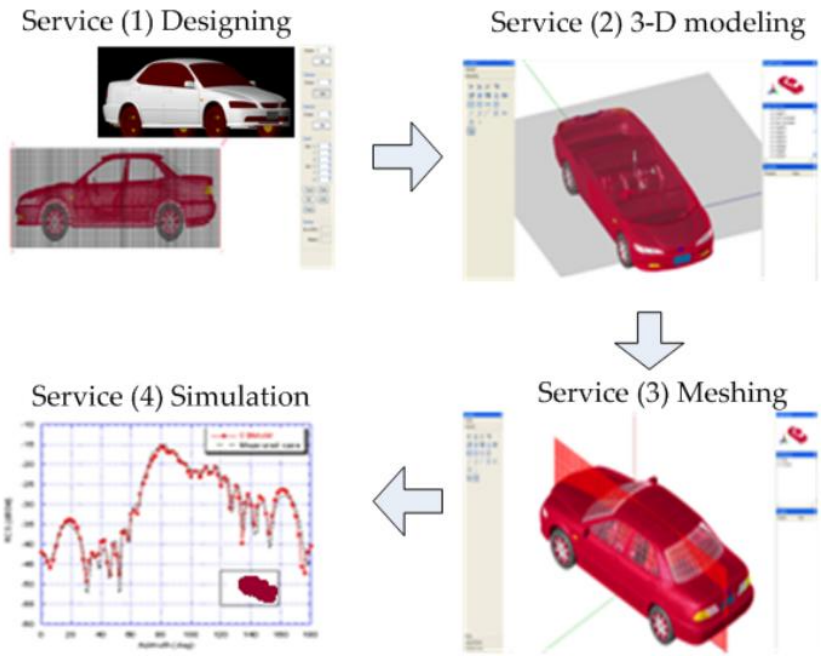
all the QoS requirements. In summary, if the computational time is critical for the composition, we suggest that Models 5 and 6 will be good choices. If the optimality is important, Models 1 and 2 will be the choice. And we can see that Model 2's computational time is less than Model 1's for all the four experiments. If we want to take care of both the optimality and the finding of a solution are important, and we do not care about computational time, we can use Models 3 and 4.

6.6 An Application to Manufacturing Process Integration

The standardization of XML (Extensible Markup Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description, Discovery, and Integration) enables information exchange across platforms in varieties of areas. Web service integration engine provides the manufacturing industry the ability to horizontally and vertically integrate data across a wide range-machines plants, vendors and enterprise domains. Manufacturing Execution System (MES) and Enterprize Resource Planning (ERP) are able to exchange data of distributed processes through the Internet. This whole system comprises of a wide-area distributed systems which are typically connected to the Internet or the Intranet.

In the case study we considered, the manufacturer user has an information system, which has a client machine for a business domain. There are separate service providers to collect and manipulate information via the Internet. An application from the business domain is sent to a service broker, which is running on the Internet, and reads information pool of service providers from the service registry center. According to the information that the manufacturer knows, and the information that the manufacturer

needs, a proper service that can be found, if it exists. Thus, the manufacturer and the service provider can exchange data between them using SOAP communication protocol. If no such a service can meet the manufacturer's needs alone, a series of web services may be able to accomplish it. To accomplish this, an algorithm of service composition is needed. Figure 8-1 shows the web service topology mentioned here.



(car models simulated by GEMS, an EM simulation software)

Figure 6-8 The composition chain of the application

The manufacturers have the following advantages by using the online services instead of operating all the information by themselves: (1) Reducing the cost; (2) Increasing reliability and robustness; (3) Increasing inter-operability; (4) Rich support of wide-area network via SOAP communications.

Scenario: In the near future we can visualize that the services online are all standard modules in WSDL (Web Services Description Language), and they can communicate with each other via SOAP (Service Oriented Architecture Protocol). The automobile manufacturer is trying to design a new car for the future and decides to use online web services as one of the options.

After running large-scale experiments with thousands of services in Section 6, we illustrate our approach through a simple example with a service set of 5 services:

Z , the set of web services, including 5 online web services, i.e., $m = 5$.

$$
\begin{aligned}
&Service_1 : \text{"sketchdesigning"} - - - \\
&\begin{cases} inputs: & \text{style,functions,price} \\ outputs: & \text{2Dmodel,tolerance,materialinfo} \end{cases} \\
&Service_2 : \text{"3Dimensionmodeling"} - - - \\
&\begin{cases} inputs: & \text{2Dmodel,tolerance,materialinfo} \\ outputs: & \text{shapes,vertices,structure} \end{cases} \\
&Service_3 : \text{"surfaceandvolumemeshing"} - - - \\
&\begin{cases} inputs: & \text{shapes,vertices,structure} \\ outputs: & \text{nodes,elements,assignedmaterial} \end{cases} \\
&Service_4 : \text{"simulationandanalysis"} - - - \\
&\begin{cases} inputs: & \text{nodes,edges,assignedmaterial} \\ outputs: & \text{safety,mileage,speed} \end{cases} \\
&Service_5 : \text{"onlinesurvey"} - - - \\
&\begin{cases} inputs: & \text{safety,mileage,speed} \\ outputs: & \text{price,structure,speed} \end{cases}
\end{aligned}
$$

The total set of the attributes are: [style, functions, price, 2D model, tolerance, material info, shapes, structure, vertex, nodes, elements, assigned material, safety, mileage, speed]

Z_0 is the service that the automobile manufacturer requests. I, input attributes of the web services: $I = O$ and $n = 15$.

$$I_1 = [1,1,1,0,0,0,0,0,0,0,0,0,0,0,0];$$

$$O_1 = [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0];$$

$$I_2 = [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0];$$

$$O_2 = [0,0,0,0,0,0,1,1,1,0,0,0,0,0,0];$$

$$I_3 = [0,0,0,0,0,0,1,1,1,0,0,0,0,0,0];$$

$$O_3 = [0,0,0,0,0,0,0,0,0,1,1,1,0,0,0];$$

$$I_4 = [0,0,0,0,0,0,0,0,0,1,1,1,0,0,0];$$

$$O_4 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1];$$

$$I_5 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1];$$

$$O_5 = [0,0,1,0,0,0,0,0,1,0,0,0,0,0,1];$$

So, the attributes set of input I_0, and the attribute set of output O_0 are:

$$I_0 = [1,1,1,0,0,0,0,0,0,0,0,0,0,0,0];$$

$$O_0 = [0,0,0,0,0,0,0,0,1,0,0,0,0,0,1];$$

which will formulate the functional constraints. p_j, fixed price of service Z_j, $j = 1,2,\ldots,5$:

$$[p_1, p_2, p_3, p_4, p_5] = [\$115K, \$114K, \$92K, \$101K, \$49K]$$

t_j, service execution time of the service Z_j:

$$[t_1, t_2, t_3, t_4, t_5] = [5days, 8days, 7days, 9days, 3days]$$

q_j, reliability of the service Z_j, the reliability is defined by the failure rate:

$$[f_1, f_2, \ldots, f_5)] = [0.02, 0.02, 0.01, 0.1, 0.1]$$

$$= [-\log 0.02, -\log 0.02,$$

$$-\log 0.01, -\log 0.1, -\log 0.1]$$

$$= [1.699, 1.699, 2, 1, 1]$$

C_0, the maximum total cost that the automobile manufacturer would be willing to pay is: $C_0 = \$550,000$. This is a nonfunctional constraint.

T_0, total service execution time limitation that the automobile manufacturer requires: $T_0 = 5$ days. This is a nonfunctional constraint.

Q_0, minimal reliability of a single composing service that the automobile manufacturer allows. If the upper bound of failure rate of each service involved is $f_0 = 0.2$; then $Q_0 = -\log 0.2$. This is a nonfunctional constraint.

Q_1, minimal total reliability of all composing services that the automobile manufacturer allows. For example, $Q_1 = 0$. This is a nonfunctional constraint.

L, the maximum times of composing, i.e. the maximal number of levels. Here let us define $L = 5$.

Decision variables are:

Z_{lj}, the i^{th} web service in the l^{th} level of composition, $l = 1, 2, 3, 4, 5$; $j = 1, 2, 3, 4, 5$.

Let us take the preemptive goal programming model as an example. Then the procedure is as follows:

Step 1: Normalize

1. Normalize the fixed cost of acquiring the service from Z_j, $j = 1,2,\ldots,5$, and the total cost limit, using the maximum value of the elements (which is the C_0 value 550, in this case).

$$[p_1, p_2, p_3, p_4, p_5, C_0] = [115,114,92,101,49,550]$$

$$\Rightarrow [\frac{115}{550}, \frac{114}{550}, \frac{92}{550}, \frac{101}{550}, \frac{49}{550}, \frac{550}{550}]$$

2. Normalize the service execution time of the service Z_j, $j = 1,2,\ldots,5$ and the total service execution time limit, using the maximum value of the elements (which is T_0 value 50 in this case):

$$[t_1, t_2, t_3, t_4, t_5, T_0] = [5,8,7,9,3,35]$$

$$\Rightarrow [\frac{5}{35}, \frac{8}{35}, \frac{7}{35}, \frac{9}{35}, \frac{3}{35}, \frac{35}{35}]$$

3. Normalize the reliability of the service Z_j, the reliability is defined by the failure rate and the total failure rate limit, by using the maximum value of the elements (which is the Q_1 value 1):

$$[q_1, q_2, \ldots, q_5] = [1.699,1.699,2,1,1]$$

$$\Rightarrow [\frac{1.699}{7.398}, \frac{1.699}{7.398}, \frac{2}{7.398}, \frac{1}{7.398}, \frac{1}{7.398}]$$

Step 2: Using the MCGPO method discussed in Section 4.5.2, we can build a preemptive goal programming model. Given that the maximum depth of levels is 5, the preemptive goal programming model finds a solution that needs to carry out service

composition in 4 levels, and the solution is: $Z_{11} = Z_{22} = Z_{33} = Z_{44} = 1$, and the others are zero.

At the first level, service Z_1 is selected to execute using the input of initial request I_0; at the second level, service Z_2 is selected execute using the output from Z_1; at the third level, service Z_3 is selected to execute using the output from Z_1, Z_2; at the fourth level, service Z_4 is selected to execute using the outputs from Z_1, Z_2, Z_3. This is shown schematically in Figure 6-8. In the solution, all the artificial variables related with the goal constraints are zero, and the optimal solution is found in the MCGPO model.

6.7 Conclusions and Future Work

In this chapter, we have built and analyzed multi-criteria mathematical programming models (MCP, MCGPO and MCGPN) in web service composition. Six models were explored in different situations. The six models can be placed into two categories: preemptive and non-preemptive methods. In the non-preemptive goal programming model, we need the weights for the objectives, which may be difficult for the customers to know. Preemptive goal programming is more suitable for web service composition, since customers can easily specify an order of priorities of the criteria. Generally, the customers have upper bounds on the total cost and the total service execution time, and a lower bound on the reliability score of the services and the composition (Here we assume that higher the reliability score, the more reliable the service is). Moreover, the customers can determine which goal or goals that they can sacrifice in the case these three goals cannot be achieved at the same time. In the

preemptive goal programming, the customer only needs to specify the most important goal, and then follow by those that are sequentially with lower priority. This enables the e-business customer to make their decisions in a practical and yet in a simple manner. However, in large-scale service networks, the computational speed of non-preemptive method is higher than that of preemptive method. MCP and MCGPO modes can be used to find optimal compositions. MCGPN models can be used to find acceptable non-optimal compositions. Both MCGPO and MCGPN models can give a compromise composition when MCP does not have a solution that satisfies both functional and nonfunctional (QoS) requirements of the query. A summary of all the six models developed in this work can be found in Table 6-4.

In the future, it is important to introduce uncertainty in the web services composition modeling. Moreover, it would be necessary to consider negotiation between the customers and the service providers, allowing for the service providers to sometimes reject a request due to network constraints.

Chapter 7. Real Time Stochastic Solution using CS Network Matrix

Two dynamic programming models based on concept service (CS) network matrix to solve service composition problem are introduced in this chapter. This problem is modeled as a Markov decision process (MDP) which considers QoS (Quality of Services), and uncertainty in service providers' availability and rate of acceptance of a query. In addition, customers may also change their queries during the composition. Before we explore this piece of work, we start with Markov Decision Process in service composition literature.

7.1 Related Literature: Markov Decision Process

In Markov Decision Process, the decision problem is inherently stochastic, and sequential in time. The transition probability and reward function depends only on the current state and action taken and independent of history.

A MDP process can be represented as a *tuple* <*S, A ,p, R*>, where

• *S* is a finite set of states;

• *A* is a finite set of actions;

• p is the state transition function, mapping a state to another state through an action, is a probability distribution. The probability of reaching state s' by performing action a in state *S* is written as $p(S'|S, a)$, and

• *R:* $S \times A \rightarrow \mathbb{R}$ is the reward function, which is the reward that either the system or the customer can gain by transiting from one state to another state by executing an action.

The objective of MDP is to find an optimal policy which maps states to actions. The optimal solution to an MDP is a policy that maximizes its expected total utility. There are two different methods for calculating the optimal policy: (1) value iteration and (2) policy iteration. Value iteration and policy iteration are polynomial time methods with respect to the size of the state space. And the state space is usually exponentially large in the size of input set, the number of web services. Overall, MDP does not scale well to large-problem size.

Gao *et.al*. (Gao et. al., 2005) formulated a MDP model considering QoS. In the utility function of the model reliability, cost, response time and availability are considered. The decision problem is to compute a policy that maximizes the value of the utility function.

Definition 7-1 *Decision rule:* A decision rule δ_t is to determine an action over an action set A.

Definition 7-2 *Policy*: A policy is a sequence of decision rules. A policy is denoted by π and takes the form $\pi = (\delta_1, \delta_2, \ldots, \delta_t, \ldots)$. A policy decides how to choose actions for any state.

Decision Time Unit is defined as steps. The moment that needs to make decision on the next selection of services is treated as a step. Because the number of nodes defined

in a compound web service is finite, finite horizon MDP is used in the composition problem.

Definition 7-3 *State*: State is defined as a conjunction of status of each task node. Suppose there are M task nodes, then a system state of compound service is written as an M-tuple:

$< s_1 \dots s_i \dots s_M > (1 \leq i \leq M, 0 \leq s_i \leq 1)$, where $s_i = 1$ represents that this node is active and has been used as a task; while $s_i = 0$ means that this node is not active.

Definition 7-4 *Action:* Actions may be taken at some state is the set of candidate web services that can be bound to the current active task node. For node i if there are M_i candidate services, then the cardinality of the set $A(i)$ is M_i.

Definition 7-5 *Reward:* In a state, the reward is the pay off which can be defined as QoS vectors associated with the service $s_{i,j}$ selected.

Definition 7-6 *Transition Probability Matrix:* The state where node i is active is written as $< s_1 \dots s_i \dots >$ with $s_i = 1$, from which system will enter state $< s_1 \dots s_i \dots s_j \dots >$ (with $s_j = 1$) with probability of $reliability(s_{i,j})$ or stay at state $< s_1 \dots s_i \dots >$ with probability of $1 - reliability(s_{i,j})$.

An action of choosing service $s_{i,j}$ may have two possible results: (1) If the web service $S_{i,j}$ succeeds, the system enters the next sate with probability $reliability(s_{i,j})$; (2) If the web service $s_{i,j}$ fails, the system remains unchanged in the current state with probability $1 - reliability(s_{i,j})$.

The above definition of transition probability from one state to another is for sequential construct. As for OR and AND parallel, and iterative cases, we need to revise the constructs in the following format:

Under conditional case /OR parallel, if there are k choices in a state and the probability associated with each branch is $p_1, \ldots, p_k$, and the current state is $< s_1 \ldots s_i \ldots s_M >$ with s_i corresponding to the choice in the next stage, system will enter state $< s_1 \ldots s_i \ldots s_t \ldots s_M >$ with probability $p_t * reliability(s_{i,t}), t = 1,2, \ldots, k$.

Under parallel case (we also call it And parallel.), if the current state is $< s_1 \ldots s_i \ldots s_j \ldots s_M >$, the system will enter state $< s_1 \ldots s_i \ldots s_{j1}, \ldots, s_{jK} \ldots s_M >$ with transition probability $\prod_{1 \le k \le K} Reliability(s_{i,jk})$, where $s_{i,jk}$, $k = 1, \ldots, K$ are the branch services in the parallel stage.

Under iterative case in MDP, if t_{entry} and t_{exit} is the entry and exit of an iterative construct respectively. The services inside the loop are modeled using the sequential, conditional and parallel cases corresponding to their construct. The only difference locating at the position after task t_{exit} finishes. We can assume that the probability that the system will break the loop is p, and the probability that the system will remain in the loop is $(1 - p)$.

After the Dynamic model is built, we can use backward value iteration to solve it. We start from horizon N and iterates to horizon 0. When horizon 0 is reached, optimal criteria and optimal actions at each horizon are computed. The sequence of actions

computed from this algorithm is the optimal policy. The following is the value iteration algorithm. Please refer to (Liu, 2004; White, 1993; Hu, 2000) for details.

Algorithm 7-1 Backward Induction based on Value Iteration

Step1: $t = N$:

$u_N^*(i_N) = r_N(i_N),\ \forall i_N \in S.$

Step2: $t = t - 1$:

If $t = 0$, stop.

$\pi = (A_0^*, A_1^*, \ldots, A_{N-1}^*)$ is the optimal Markov policy; and $V_N^*(i) = u_0^*(i)$ is the optimal value.

Otherwise, continue with step 3.

Step3: Let

$$u_t^*(i_t) = \max_{a \in A(i_t)} \left\{ r_t(i_t, a) + \sum_{j \in S} p_t(j|i_t, a) u_{t+1}^*(j) \right\}, \quad \forall i_t \in S$$

$$A_t^*(i_t) = \arg \max_{a \in A(i_t)} \{ r_t(i_t, a) + \sum_{j \in S} p_t(j|i_t, a) u_{t+1}^*(j) \}$$

And go to step 2.

This backward recursive iteration algorithm reflects the concept of Bellman's **Principle of Optimality**, which is first presented in (Bellman, 1957). An optimal policy has the property that whatever the initial state and initial decision are, the remaining

decisions must constitute an optimal policy with regard to the state resulting from the first decision.

The drawback of the MDP algorithm is that the computational complexity grows exponentially in the number of web services in the domain. So, it does not scale well with large problem size.

7.2 Potential of Concept Service (CS) Network and Katz Index

In Chapter 5, we studied the potential of CS networks, denoted as $G^*_{(m+n)\times(m+n)} = \left[\frac{C^*_{m\times m} \mid I^*_{m\times n}}{O^*_{n\times m} \mid S^*_{n\times n}}\right]$. This matrix can give us information about the connectedness of the service network, the preliminary solution for a query $q(X_0, X_1)$, and QoS range of optimal solution. Further, in Chapter 4, service composition can be customized and optimized in different scenarios. However, we have not addressed the stochastic property of service composition and service networks. In this chapter, we will address this issue based on what we have established in Chapter 3.

7.2.1 Relation between Non-QoS(G^*) and Transition Matrix

For $G^*_{(m+n)\times(m+n)} = \left[\frac{C^*_{m\times m} \mid I^*_{m\times n}}{O^*_{n\times m} \mid S^*_{n\times n}}\right]$ obtained from the matrix operations in Euclidean space. We can further normalize each row in and each column in the way that we described in Chapter 5 for the initial concept service (CS) matrix. Here, G^* matrix can also be normalized column by column individually. $G^* = [G_1, G_2, \dots, G_{(n+m)(n+m)}]$ becomes $[\frac{G_1}{\|G_1\|_1}, \frac{G_2}{\|G_2\|_1}, \dots, \frac{G_{(n+m)}}{\|G_{(n+m)}\|_1}]$,

where $G_1, G_2, \ldots, G_{(n+m)(n+m)}$ are the column vectors in G^*. After normalizing the columns, furthermore, the rows can also be normalized. If the L_1norm of any row vector in G^* is greater than 1, I divide each elements of the row by the L_1 norm of this row; otherwise, the row is kept the same. Let us still denoted this normalized CS network matrix as G^*. Thus, G^* has the following properties:

Properties of normalized G^*:

(1) $\sum_{j=1}^{(m+n)} G^*{}_{ij} \leq 1,\ i = 1,2, \ldots, m+n.$

(2) $\sum_{i=1}^{(m+n)} G^*{}_{ij} \leq 1,\ j = 1,2, \ldots, m+n;$

For these rows with $\sum_{j=1}^{(m+n)} G^*{}_{ij} \leq 1,\ i = 1,2, \ldots, m+n$, the difference between 1 and the sum of the rows represents the ***rate of down time of the services or concepts***.

$$P_i = 1 - \sum_{j=1}^{(m+n)} G^*{}_{ij}, \quad i = 1,2, \ldots, m+n \qquad 7\text{-}1$$

For these columns with $\sum_{i=1}^{(m+n)} G^*{}_{ij} \leq 1,\ j = 1,2, \ldots, m+n$, the difference between 1 and the sum of columns represent the failure rate of connection with other services in the network. This is used to model ***connection failure of the services or concepts***.

$$Q_j = 1 - \sum_{i=1}^{(m+n)} G^*{}_{ij}, \quad j = 1,2, \ldots, m+n \qquad 7\text{-}2$$

Thus, $G^*{}_{(m+n)\times(m+n)} = \left[\frac{C^*_{m\times m}}{O^*_{n\times m}} \middle| \frac{I^*_{m\times n}}{S^*_{n\times n}}\right]$ is the transition matrix among concepts and services. G^*_{ij} represents the probability that an element (concept or service) is used or generated by another element.

From Chapter 5, we know that $G^* = \lim_{t\to\infty} \sum_{i=1}^{t} M^i = (I - M)^{-1} - I$. When M is the transition probability matrix, the expression becomes the Katz index (Katz, 1953) when the weights for path length are all 1, in Socio-metric area which is used to describe the activity of person.

Further, if we consider a discounted form of G^*, let $G^* = \lim_{t\to\infty} \sum_{i=1}^{t} \rho^i M^i = (1 - \rho M)^{-1} - I$, this is exactly of the form of Katz index. This coincidence shows that the mathematical foundation of service computing in the dissertation is valid, and can be used in other areas such as social science, global production planning, etc.

7.2.2 Relation between QoS(G^*) and Rewards

In Chapter 5, we discussed G^* in QoS, let us denote it as $(QoS(G^*))$ and we defined new operation rules for different QoS attributes (cost, reliability and delivery time). Each element in $QoS(G^*)$ record the best reward from the row element to the column element. For example, $P(G_{ij}^*)$ represents the lowest cost from element i to element j. Here the elements can be either services or concepts.

The dynamic programming technique is used to find the structure of optimal policy which can maximize the objective function. For this purpose, the actions in each state can be evaluated based on the rewards accumulated. Markov decision-making process is used in service composition, because service composition has memoryless property. In each state, the decision only considers the current given input concept set and the required output concept set, plus flow factors.

7.3 MDP Model

From the CS network potential matrix, a service provider can always find the peer services (neighbors) that can connect with it and generate the required output concepts. We further assume that the total network and edges do not have limitation on the capacity of flow; however, each node in the network has its flow capacity limitation.

7.3.1 Parameters and Variables

If there are a total of m homogeneous nodes (peers) in the network. The time horizon is finite, and $T \leq m$. we have the following parameters in the system:

$F_i,\ i = 0,1,2,\dots,m$; the flow bandwidth capacity of each node, where F_0 is the bandwidth capacity of the compound service, and F_i, i=1, 2,…, m is the bandwidth capacity of the peers.

The current value of flow of each node is f_i, i=0,1,2, …, m. We assume that the probability distribution of f_i is in uniform $[0, F_i]$, so the distribution of $\frac{f_i}{F_i}$ is in uniform [0,1]; and $\xi_i = (1 - \frac{f_i}{F_i})$ is in the uniform [0,1] distribution; i=0, 1, 2, …,m. Each of these functions can be constructed by examining the history of these providers performance.

$\xi_i = \left(1 - \frac{f_i}{F_i}\right)$ is the proportion of the free flow bandwidth of the Node N_i , i =0,1,…, T

$\xi_0 = \left(1 - \frac{f_0}{F_0}\right)$ is the proportion of the free flow bandwidth of the compound service currently;

T, the due date of searching for the service provider.

p , is a constant related with the price of the query;

K, is a very large positive number, and k, is a very small positive number (we will use them to construct reward later).

7.3.2 Decision Epochs

It is assumed that the time spent on each node for decision making is exactly 1 unit. So there are totally T decision epochs, if the query due date is T. (i=1,2,…,T). If neither the compound service nor any of the peers accept the request in T steps, the procedure stops.

7.3.3 States

Let us define the states $S_i\big(q(X_i, Y_i)\big), i = 1,2, \dots, T$, where,

$q(X_i, Y_i)$ is the equivalent query with available concepts of X_i and the required output concepts of Y_i.

In each state, we also consider the following parameters in decision making:

ξ_0 is the free flow bandwidth that the compound service has currently;

ξ_i is the free bandwidth that the node N_i has currently, and $\xi_i, i = 0,1, \dots, T,$ are the identical independent random variables.

Later, we will see that ξ_i can be estimated from the potential CS network matrix. $G^* = \lim_{(T\to\infty)} \sum_{t=1}^{T} M^t$.

δ_i indicates whether the current peer on board can provide the service or not. δ_i can be computed from the potential of CS network matrix G^*.

7.3.4 Transition Probability

The transition function maps (s, a) to a new state s'. P $(s'|s,a)$ is the probability of s' being the next state, given that a is performed in s.

The transition matrix is defined by the normalized potential CS network matrix$G^* = \lim_{(T\to\infty)} \sum_{t=1}^{T} M^t$.

7.3.5 Actions

Actions consist of accepting requests and passing the request to the neighbors.

The following three actions are considered:

a_1 ={Compound service accept, process stops}

a_2 ={Node N_i accept, process stops}

a_3 ={Compound service reject, Node N_i reject}

7.3.6 Rewards

$$R_{a\in A_s}(s_t,a_t)=\begin{cases} K*p*\xi_0^t, & a_t=a_1 \\ p*\xi_t, & a_t=a_2, \\ 0, & a_t=a_3 \end{cases} \quad \text{t=1,2,…, T-1} \qquad 7\text{-}3$$

$$R_{a\in A_{s_T}}(s_T,a_T)=\begin{cases} K*p*\xi_0^T, & a_t=a_1 \\ p*\xi_T, & a_t=a_2, \\ k, & a_t=a_3 \end{cases} \qquad 7\text{-}4$$

when the request is not accepted at the due step T, a small award m is given.

K is a large positive number, k is a small positive number;

p is the reward factor;

ξ_0^t is the factor of free flow bandwidth of the compound service, and ξ_t is the factor of the flow bandwidth of the peer in the current stage.

7.3.7 Optimality Equation

The utility function in an MDP state equals to the reward of that state plus the expected rewards of the future states. The optimal action at each state is then the one to yield the highest expected utility. There are two types of rewards given in the MDP: the reward from the compound service, or the reward from the $node\ \ N_i$ or the system reward in the final step. And the optimal utility function of the system is.

$$U_t^*(s_t) = \max_{a\in A_{s_t}}\left\{R(s_t,a) + \sum_{s_{t+1}\in S} P(s_{t+1}\lceil s_t,a) * U_{t+1}^*(s_{t+1})\right\}\ , \qquad 7\text{-}5$$

$$t = 1,2,\ldots, T;$$

so,

$$d_t(s_t) = \operatorname{argmax}_{a\in A_{s_t}}\left\{R(s_t,a) + \sum_{s_{t+1}\in S} P(s_{t+1}\lceil s_t,a) * U_{t+1}^*(s_{t+1})\right\}, \qquad 7\text{-}6$$

$$t=1,2,\ldots,T.$$

7.3.8 Backward Induction

In this section, let us explore the structure of optimal policy via backward induction.

(1) t=T+1:

$U_{T+1}^*(S_{T+1}) = 0;$

(2) t=T:

$$U_T^*(S_T) = \max_{a_1,a_2,a_3}\{R_T(S_T,a)\} = \max\{Kp\xi_0^T,\quad p\xi^T,\quad k\}$$

$$U_T^*(S_T) = \begin{cases} Kp\xi_0^T, & Kp\xi_0^T \geq \xi^T p \ and\ Kp\xi_0^T \geq k \\ p\xi^T, & p\xi^T > Kp\xi_0^T \ and\ p\xi^T \geq k \\ k, & Kp\xi_0^T < k \ and\ p\xi^T < k \end{cases}$$

$$d_T(S_T) = arg\max_{a_1,a_2,a_3}\{R_T(S_T,a)\}=\begin{cases} a_1\,, & Kp\xi_0^T \geq \xi^T p \; and \; Kp\xi_0^T \geq k \\ a_2, & p\xi^T > Kp\xi_0^T \; and \; p\xi^T \geq k \\ a_3, & Kp\xi_0^T < k \; and \; p\xi^T < k \end{cases}$$

(3) t=T-1:

$$U_{T-1}^*(S_{T-1}) = \max_{a\in A_{s_{T-1}}}\left\{R(s_{T-1},a) + \sum\nolimits_{s_T\in S} P(s_T\lceil s_{T-1},a) * U_T^*(s_T)\right\}$$

$$= \max\{Mp\xi_0^{T-1}, \quad p\xi^{T-1}, \quad 0 + E[U_T^*(S_T)]\}$$

so,

$$U_{T-1}^*(S_{T-1}) = \begin{cases} Kp\xi_0^{T-1}, & K\xi_0^{T-1} \geq \xi^{T-1} \; and \; K\xi_0^{T-1} \geq E[U_T^*(S_T)] \\ p\xi^{T-1}, & K\xi_0^{T-1} < \xi^{T-1} and \; \xi^{T-1} \geq E[U_T^*(S_T)] \\ E[U_T^*(S_T)], & Kp\xi_0^{T-1} < E[U_T^*(S_T)] \; and \; p\xi^{T-1} < E[U_T^*(S_T)] \end{cases}$$

$$d_{T-1}(S_{T-1}) = \begin{cases} a_1, & K\xi_0^{T-1} \geq \xi^{T-1} \; and \; k\xi_0^{T-1} \geq E[U_T^*(S_T)]; \\ a_2, & K\xi_0^{T-1} < \xi^{T-1} and \; \xi^{T-1} \geq E[U_T^*(S_T)]; \\ a_3, & Kp\xi_0^{T-1} < E[U_T^*(S_T)] \; and \; p\xi^{T-1} < E[U_T^*(S_T)] \end{cases}$$

(4) At step t:

$$U_t^*(S_t) = \max_{a\in A_{s_t}}\left\{R(s_t,a) + \sum_{s_{t+1}\in S} P(s_{t+1}\lceil s_t,a) * U_{t+1}^*(s_{t+1})\right\}, \quad t = 1,2,\ldots, T;$$

So,

$$U_t^*(S_t) = \begin{cases} Kp\xi_0^t, & K\xi_0^t \geq \xi^t \; and \; Kp\xi_0^t \geq E[U_{t+1}^*(S_{t+1})] \\ p\xi^t, & K\xi_0^t < \xi^t and \; p\xi^t \geq E[U_{t+1}^*(S_{t+1})] \\ E[U_{t+1}^*(S_{t+1})], & Kp\xi_0^t < E[U_{t+1}^*(S_{t+1})] \; and \; p\xi^t < E[U_{t+1}^*(S_{t+1})] \end{cases}$$

And

$$d_t(S_t) = \begin{cases} a_1, & K\xi_0^t \geq \xi^t \; and \; K\xi_0^t \geq E[U_{t+1}^*(S_{t+1})]; \\ a_2, & K\xi_0^t < \xi^t and \; \xi^t \geq E[U_{t+1}^*(S_{t+1})]; \\ a_3, & Kp\xi_0^t < E[U_{t+1}^*(S_{t+1})] \; and \; p\xi^t < E[U_{t+1}^*(S_{t+1})] \end{cases}$$

7.3.9 Optimal Policy

Next, let us estimate $E[U_{t+1}^*(S_{t+1})]$ for the optimal policy:

$$d_t(S_t) = \begin{cases} a_1, & K\xi_0^t \geq \xi^t \; and \; K\xi_0^t \geq E[U_{t+1}^*(S_{t+1})]; \\ a_2, & K\xi_0^t < \xi^t and \; \xi^t \geq E[U_{t+1}^*(S_{t+1})]; \\ a_3, & Kp\xi_0^t < E[U_{t+1}^*(S_{t+1})] and \; p\xi^t < E[U_{t+1}^*(S_{t+1})] \end{cases}$$

If the state S_{t+1} can be represented by a new query $q(X_{t+1}, Y_{t+1})$ where X_{t+1} are the concepts available and Y_{t+1} is the output concepts needed.

The normalized G^* is the transition matrix. From Chapter 5, we know that G^* reports the connectivity of concepts and services. It is reasonable to predict that these with more connections with other services will have more work load. So, G^* is also an indicator of the rate of work load of each element, especially service servers. Use A to represent the matrix of all ones. Then A-G^* can represent the matrix of free capacity.

Then

$$E[U_{t+1}^*(S_{t+1})]$$

$$= [X^T{}_{t+1}, 0_{1\times m}]\left(\left(G^*.QoS(G^*)\right).\left(A_{(m+n)\times(m+n)} - G^*\right)\right)\begin{bmatrix} Y_{t+1} \\ 0_{m1} \end{bmatrix}$$

$$= X^T{}_{t+1}\left(\left(C^*.QoS(C^*)\right).\left(A_{m\times m} - C^*\right)\right)Y_{t+1}. \qquad 7\text{-}7$$

Where $\left(C^*.QoS(C^*)\right)$ is the matrix with elements as the product of the two corresponding elements in C^* and $QoS(C^*)$; and $\left(\left(C^*.QoS(C^*)\right).\left(A_{m\times m} - C^*\right)\right)Y_{t+1}$ is the matrix obtained by the product of the two corresponding elements in $\left(C^*.QoS(C^*)\right)$ and $(A_{m\times m} - C^*)$.

When K is a very large number, we know that,

(1) $K\xi_0^t \geq \xi^t \; and \; K\xi_0^t \geq E[U_{t+1}^*(S_{t+1})]$ holds, whenever $\xi_0^t > 0$, that means, we let the compound service respond to the request if the compound service's current transaction flow is less than its capacity. We take action a_1.

(2) $K\xi_0^t < \xi^t and \; \xi^t \geq E[U_{t+1}^*(S_{t+1})]$ holds, when $\xi_0^t = 0$, that is the compound service reaches its full capacity, and the current flow bandwidth of the peer on board is less than a threshold $E[U_{t+1}^*(S_{t+1})]$, which is the expected return of the successive steps.

(3) $Kp\xi_0^t < E[U_{t+1}^*(S_{t+1})] \; and \; p\xi^t < E[U_{t+1}^*(S_{t+1})]$ holds, when $\xi_0^t = 0$, that is the compound service reaches its full capacity, and the reward based on the current free bandwidth of the peer on board is smaller than a threshold $E[U_{t+1}^*(S_{t+1})]$, which is the expected return of the successive steps. We take action a_3.

Now, let us study an extreme case when the reword K to the compound service is infinitely large.

(i) If the compound service's current transaction flow is less than its capacity F_0, the compound service should accept the query and provides service to the requester. We take action a_1.

(ii) If the compound service's current transaction flow reaches its capacity F_0, we start searching among the peers, and continue searching until: any peer accepts the request positively and provides service to the requester, and each peer has a threshold for accepting the query in each step; If the compound service's current bandwidth turns out to be lower than its capacity, then it accepts the request, and provides service to the requester.

(iii) If neither the compound service nor the peers accept the query at the end of the due time, we take action a_3.

7.4 Discounted MDP Model

In Gao *et. al*.'s model based on Dynamic programming, the capacity of web service providers is not considered, which is important in a dynamic environment. The capacity of web service providers is considered in the proposed model. Furthermore, it is necessary to encourage service providers' to accept the task if they can provide the service, since dynamic programming grows exponentially with the problem size. It consumes computational and transportation resources on the internet if a request lingers in the network for long. In the proposed dynamic programming formulation, discount factor is added in the reward to encourage the acceptance of tasks by service providers. Moreover, in order to save computational time, the algorithm also encourages the use of existing nested service complex to generate a compound service plan for customers.

7.4.1 Discounted MDP with Nested Work Flow and Capacity Constraints

Let T be the maximal steps of searching horizon. If we assume the time for decision making at each step is 1 unit, then T is the maximum steps of searching. Let's assume that$T \leq m$, where m is the total number of services in the network. T is determined by the query from the customer. Then we will need the following parameters to define the system:

$F_t^s,\ t = 1, 2, \dots, T$, be the capacity of the nested abstract web service at the t^{th} step. The nested abstract service is a known nested workflow that can fulfill the job from the

current state t towards the end. $s \in S_t^c$, which is the set of matched compound services, e.g. nested work flows, at step t.

f_t^s, t = 1, 2, …, T, be the work load of the nested abstract web service at the t^{th} step; If the work load on this nested abstract web service is full, this means that the abstract service is not available at the moment. If the capacity of this abstract web service is 0, this nested abstract web service does not exist at the t^{th} step, $s \in S_t^c$.

$\xi_t^s = \left(1 - \frac{f_t^s}{F_t^s}\right)$, the proportion of the free capacity of the nested workflow at the t^{th} step; t =0, 1, …, T; Let η_1 is the random variable that denotes the proportion of the free capacity of the compound service at each step. And we assume that η_1 follows the distribution $p\{\eta_1 = \xi_t^s\} = \xi_t^s, s \in S_t^c, \mathrm{t} = 1, 2, \ldots, \mathrm{T}$.

$F_t^p,\ t = 1, 2, \ldots, T$, the capacity of the service on board at the t^{th} step, $p \in S_t^p$, which is the set of matched individual services at step t.

f_t^p, the work load of the web service on board at the t^{th} step, $p \in S_t^p$, t=1,2, …, T

$\xi_t^p = \left(1 - \frac{f_t^p}{F_t^p}\right)$, the proportion of the free capacity of the web service on board at the t^{th} step, Let η_2 be the random variable that denotes the proportion of the free flow capacity of the provider on board at each step. Later, we will see that ξ_t^p can be estimated from the potential CS network matrix. $G^* = \lim_{(T\to\infty)} \sum_{t=1}^{T} p^t M^t$.

Q_t^i, a QoS (Quality of Service) of the web services on board at the t^{th} step, $i \in S_t^p \sqcup S_t^c$;

K, is a very large positive number, and k is a very small positive number. The two numbers will be used to construct the rewards later.

7.4.2 Decision Epochs

At each step, the system needs to select either a compound web service or an individual web service on board, i=1,2,…,T. If the requested service cannot be composed in T steps, the procedure terminates.

7.4.3 States

The state set S_t is defined as same as in Section 7.3.3.

7.4.4 Actions

A_t ={Selections of services and compound services}

7.4.5 Rewards

The system can receive the following rewards according to the action in each state. The system encourages the provider to accept the request if its free flow bandwidth is high. The purpose of this design is to avoid overload of some busy service providers in the system. The system also encourage an early acceptance of the query by adding a discount factor ρ.

Varied rewards in terms of the rate of free flow are formulated as follows:

$$R_{a\in A_S}(s_t, a_t) = \begin{cases} K * \rho^t * Q_t^{a_t} * \xi_t^s, & a_t = a_1 \\ \rho^t * Q_t^{a_t} * \xi_t^p, & a_t = a_2, \\ 0, & a_t = a_3 \end{cases} \quad \text{t=1,2,…, T-1.} \qquad 7\text{-}8$$

$$R_{a\in A_S}(s_T, a_T) = \begin{cases} K * \rho^t * Q_t^{a_t} * \xi_T^s, & a_T = a_1 \\ \rho^t * Q_t^{a_t} * \xi_T^p, & a_T = a_2, \text{ at the final step.} \\ k, & a_T = a_3 \end{cases} \quad 7\text{-}9$$

Where a_1 represents a compound service, and a_2 represents an individual service, and a_3 represents no choice.

ρ is the discount factor, $0 < \rho < 1$.

7.4.6 Transition Probability

The same as in Section 5.4, the transition matrix is defined by normalized G^*, specifically, the sub-matrix S^* in $G^* = \lim_{(T\to\infty)} \sum_{t=1}^{T} \rho^t M^t$.

7.4.7 Optimality Equation

The utility function in a state equals to the reward in that state plus the expected rewards in the future states. The optimal action at each state is then the one to yield the highest expected utility in the state. The optimality equation of the system is:

$$U_t^*(s_t) = \max_{a\in A_{s_t}}\{R(s_t, a) + \sum_{s_{t+1}\in S} P(s_{t+1}\lceil s_t, a) * U_{t+1}^*(s_{t+1})\}, \quad 7\text{-}10$$

t =1,2,…, T;

so,

$$d_t(s_t) = \text{argmax}_{a\in A_{s_t}}\{R(s_t, a) + \sum_{s_{t+1}\in S} P(s_{t+1}\lceil s_t, a) * U_{t+1}^*(s_{t+1})\}, \quad 7\text{-}11$$

t=1,2,…,T;

In this model, the bandwidths of service providers are considered as state variables.

7.4.8 Optimal Policy

The optimal policy can be further studied based on:

$$d_t(S_t) =$$

$$\begin{cases} a_1,\ K * \rho^t * Q_t^{a_t} * \xi_t^s \geq \rho^t * Q_t^{a_t} * \xi_t^p \ and\ K * \rho^t * Q_t^{a_t} * \xi_t^s \geq E[U_{t+1}^*(S_{t+1})]; \\ a_2,\ K * \rho^t * Q_t^{a_t} * \xi_t^s < \rho^t * Q_t^{a_t} * \xi_t^p and\ \rho^t * Q_t^{a_t} * \xi_t^p \geq E[U_{t+1}^*(S_{t+1})]; \\ a_3,\ \ K * \rho^t * Q_t^{a_t} * \xi_t^s < E[U_{t+1}^*(S_{t+1})] and\ \rho^t * Q_t^{a_t} * \xi_t^p < E[U_{t+1}^*(S_{t+1})] \end{cases} \quad 7\text{-}12$$

Now, let us estimate $E[U_{t+1}^*(S_{t+1})]$ in the discounted case.

If the state S_{t+1} can be represented by a new query $q(X_{t+1}, Y_{t+1})$ where X_{t+1} are the concepts available and Y_{t+1} is the output concepts needed.

The normalized $G^* = \lim_{(T\to\infty)} \sum_{t=1}^{T} p^t M^t$ is the transition matrix. From Chapter 5, we know that G^* reports the connectivity of concepts and services. It is reasonable to predict that these with more connections with other services will have more work load. So, G^* is also an indicator of the rate of work load of each element, especially service servers. Use A to represent the matrix of all ones. Then A-G^* can represent the matrix of free capacity.

Then

$$E[U_{t+1}^*(S_{t+1})]$$

$$= [X^T{}_{t+1}, 0_{1\times m}]\left(\left(G^*.QoS(G^*)\right).\left(A_{(m+n)\times(m+n)} - G^*\right)\right)\begin{bmatrix} Y_{t+1} \\ 0_{m1} \end{bmatrix} \quad 7\text{-}13$$

$$= X^T{}_{t+1}\left(\left(C^*.QoS(C^*)\right).\left(A_{m\times m} - C^*\right)\right)Y_{t+1}\,.$$

Where $\left(C^*.QoS(C^*)\right)$ is the matrix with elements as the product of the two corresponding elements in C^* and $QoS(C^*)$; and $\left(\left(C^*.QoS(C^*)\right).\left(A_{m\times m} - C^*\right)\right)Y_{t+1}$ is the matrix obtained by the product of the two corresponding elements in $\left(C^*.QoS(C^*)\right)$ and $(A_{m\times m} - C^*)$.

Since K is a very large number, we know that:

(1)$K * \rho^t * Q_t^{a_t} * \xi_t^s \geq \rho^t * Q_t^{a_t} * \xi_t^p \; and \; K * \rho^t * Q_t^{a_t} * \xi_t^s \geq E[U_{t+1}^*(S_{t+1})]$ holds, whenever $\xi_t^S > 0$, that means, we let the compound service response the request if the compound service's current transaction flow is less than its capacity. We take action a_1.

(2) $K * \rho^t * Q_t^{a_t} * \xi_t^s < \rho^t * Q_t^{a_t} * \xi_t^p and \; \rho^t * Q_t^{a_t} * \xi_t^p \geq E[U_{t+1}^*(S_{t+1})]$ holds, when $\xi_t^S = 0$, that is the compound service reaches its full capacity, and the current flow bandwidth of the peer on board is less than a threshold $E[U_{t+1}^*(S_{t+1})]$, which is the expected return of the successive steps.

(3) $K * \rho^t Q_t^{a_t} * \xi_t^s < E[U_{t+1}^*(S_{t+1})] and \; \rho^t Q_t^{a_t} * \xi_t^p < E[U_{t+1}^*(S_{t+1})]$ holds, when $\xi_0^t = 0$, and $\rho^t * Q_t^{a_t} * \xi_t^p < E[U_{t+1}^*(S_{t+1})]$ that is the compound service reaches its full capacity, and the reward based on current flow bandwidth of the peer on board is smaller than a threshold $E[U_{t+1}^*(S_{t+1})]$, which is the expected return of the successive steps. We take action a_3.

Now, let us look at an extreme case when the reward *K* to the compound service is infinitely large.

(i) If the compound service's current transaction flow is less than its capacity F_0, the compound service should accept the query and provides service to the requester. We take action a_1.

(ii) If the compound service's current transaction flow reaches its capacity F_0, we start searching among the connected peers, and keep the searching process until: any peer accepts the request positively and provides service to the requester, and each peer has a threshold for accepting the query in each step; If the compound

service's current flow bandwidth turns lower than its capacity, the compound service accepts the request, and provides service to the requester.

(iii) If neither the compound service nor the peers accept the query at the end of the due time. We take action a_3.

This conclusion is the same as the regular MDP policy without discount factor.

7.5 Experiments

In this section, we provide two simple case studies to test the analytical optimal policy derived above. The MDP service composition problem is coded in MATLAB.

7.5.1 Simulation

Figure 7-1 shows that the utility is an increasing function in terms of the compound service's free flow bandwidth under the optimal policy.

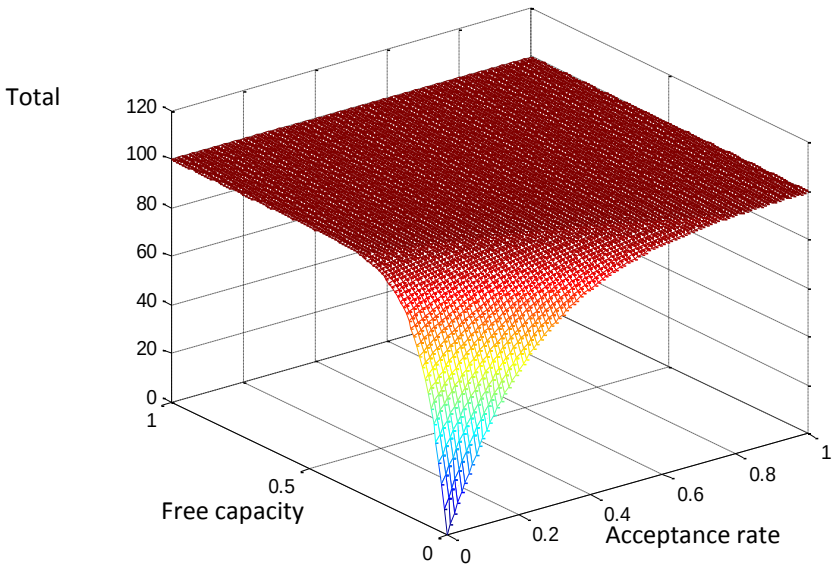


Figure 7-1 Total utility vs. the free capacity and acceptance rate of the compound service

7.5.2 Example

We consider the following parameters: reward factor $p = 10$, $K = 10000$, $k = 1$ and time horizon $T = 5$.

Case 1: In this experiment, we randomly assign the flow values of the compound services and the peer on board in each step. We pick the values for the preferred compound service as follows: the free bandwidth rates of the compound service $\{\xi_0^1, \xi_0^2, \dots, \xi_0^5\}$ = {0, 0.01, 0.7, 0.05, 0.3}; and the free bandwidth rates of the individual service provider on board in each step $\{\xi_1, \xi_2, \dots, \xi_5\}$= {0.7, 0.9, 0.9, 0.1, 0.2}. The algorithm will terminate at step 1 with action a_2 (the peer accepts the query and provides service to the requester). This follows optimality policy (2). The actions are shown in Figure 7-2.

Case 2: Free bandwidth rates of the compound service are $\{\xi_0^1, \xi_0^2, \dots, \xi_0^5\}$ = {0, 0, 0, 0, 0.3}; and free bandwidth rates of the individual service provider on board in each step are $\{\xi_1, \xi_2, \dots, \xi_5\}$= {0.0001, 0.0002, 0.00005, 0.00003, 0.00008}. The algorithm terminates at step 5 with action a_1. The actions are shown in Figure 7-3.

Here we only consider one compound service and one individual service at each step. The MDP model and the analytical solutions can be easily extended to multiple compound services and multiple individual services on board at each step.

Figure 7-2 Case 1

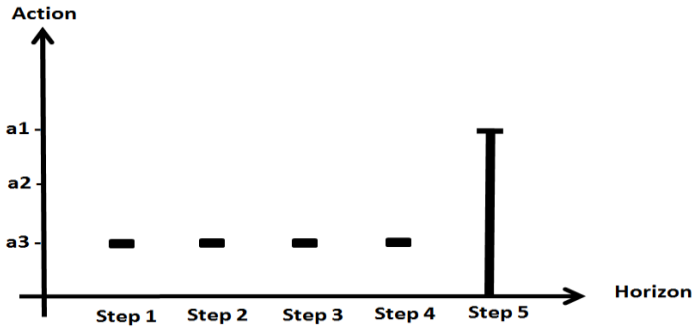


Figure 7-3 Case 2

7.6 Conclusions and Future Work

In this chapter, The optimal policies (discounted and non-discounted) for real time service composition problem considering the nested compound work flow and the work load of the service providers are developed. The real time service composition problem is modeled as a Markov decision making process, and solved using backward induction in two Dynamic Programming models (regular and discounted). This model uses the potential Concept Service (CS) network matrix that we developed in Chapter 5 to

calculate the parameters and threshold in Markov decision making process and in the optimal policies.

In the future, more case studies and large scale simulation is worth to build to test the analytical results and policies developed in this dissertation. More sophisticated factors may be studied further in real environments.

Chapter 8. Applications

The services currently available online are all standard modules in WSDL (Web Services Description Language), and they can exchange information interactively via SOAP (Service Oriented Architecture Protocol). Utilizing the online services is usually cheaper than hiring a bunch of physical consultants. Web service composition, among other domains, is applicable in (1) health care, (2) production design, and (3) enterprise application integration and supply chain. Figure 8-1 shows the superstructure of the service network including users, registry, service providers and service brokers.

8.1 Resource Allocation in Health Care

Recently, we have been experiencing extreme deficiencies of medical resources, such as medical personnel, medical devices, blood or organs for organ transplantation. Such deficiency can be reduced by the collaboration among medical institutions and national health organizations, which might be involved in the issues of home land security, disaster recovery, and emergency management and thus the efficiency of whose performance are important. Efficient and effective collaboration in such a geographically distributed environment needs a support of Information Technology (IT) through the Internet. It is believed that web services can provide such collaborations for health care services.

8.2 Product Design in Manufacturing

One of the recent trends in product design is modularization of parts. Modularization also contributes to defining parts in a standardized, machine-readable

way. Especially, modularization enables us to define the features of each part as input, output, function and geometric information. Defining a set of inputs, outputs and other features makes it possible for a manufacturing company to identify the required parts. In order to automate such production design processes, manufactures can utilize the Web service composition algorithm to match the parts from suppliers.

8.3 Business Process Integration

Globalization is a significant concept in manufacturing industry. Services are performed by varieties of companies depending on their expertise, such as production design companies, manufacturing companies, marketing companies *etc*. These enterprises can collaborate with each other through web services integration and automation. These online business processes can be executed automatically in a scheduled order. One service could use another's output as an input, and all of these services can work together to form the entire business process. For instance, automobile manufacturers are able to utilize the online services to help designing their new car models for the coming years. This designing service could be decomposed into several sub services by functionality, for example, sketch designing, 3-D modeling, meshing, simulation, analysis, survey, etc. Generally, it is necessary to decompose the requested service into sub-problems when the service cannot be delivered in its entirety. The service brokers need to match each sub-problem with the right service available on line. As the number of web services increases and varieties of emerging service requests appear in current, competitive, and complex business environments, automatic web service composition becomes an essential feature of commercial web services.

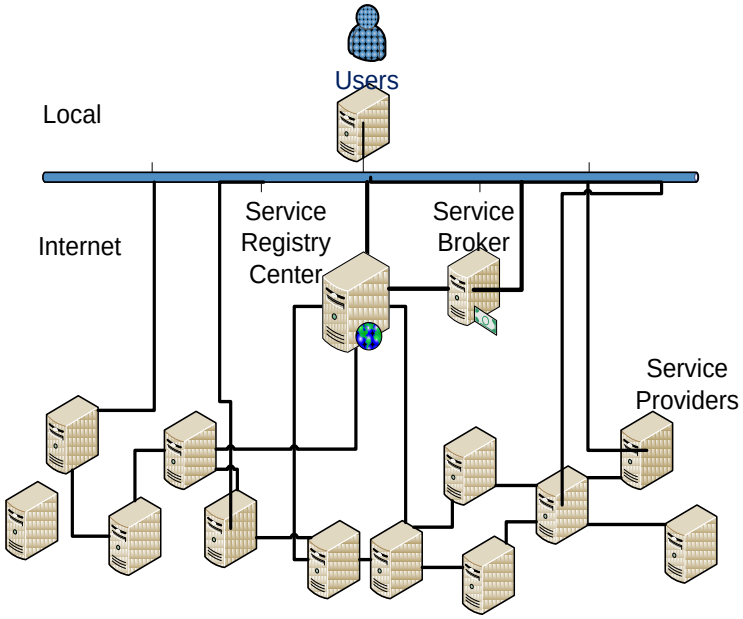


Figure 8-1 Topology of users, service providers, service brokers, and service registry center

8.4 Supply Chain Networks

Each multi-product, multi-stage, multi-supplier, and multi-customer supply chain is a system of systems. (Chen et.al, 2004) studied the supply chain problem with multiple incommensurable goals for a multi-echelon supply chain network with uncertain market demands and product prices.

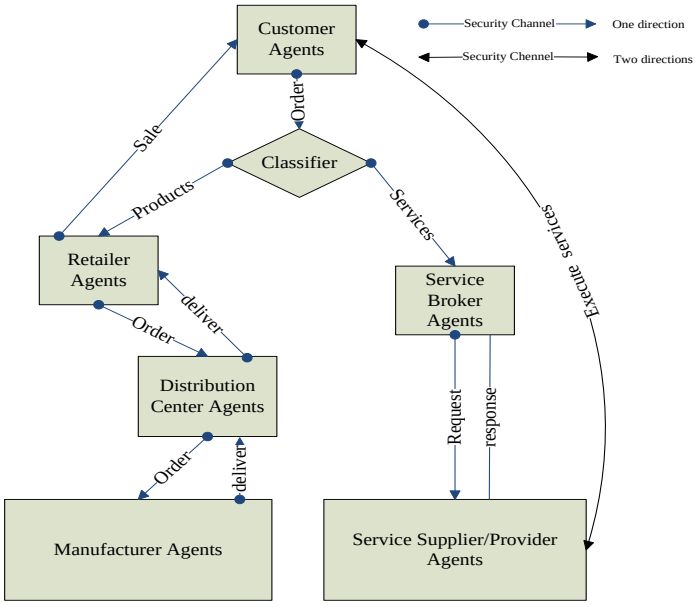


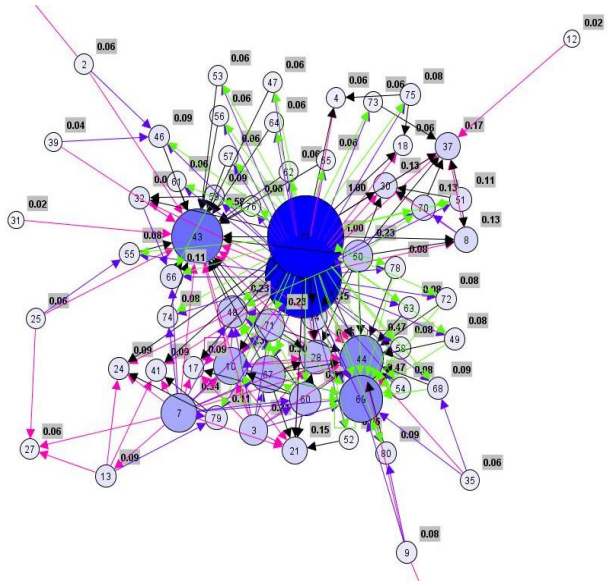
Figure 8-2 Agent based supply chain networks.

From the customers' (manufacturers') point of view, many supply chain problems can be modeled as minimum cost flow problems. Either mathematical programming or heuristic methods can be used to solve them. From the suppliers' point of view, it can be a maximum cover problem. Figure 8-2 shows an agent based architecture of supply chain networks. In these networks, customers, retailers, distribution centers, manufactures and even service providers and brokers are involved. The network is complex, and the scale of transportation network usually is national or world-wide. The work reported in (Thadakamalla et al, 2004) studies supply chains from a survivability point of view and lays the foundation for using network science in this field.

8.5 Service Networks

The number of organizations contemplating the integration of services into their strategic plans and daily operations is continuously increasing. Companies and experts

tend to offer their expertise to other companies, organizations and individuals so that the knowledge and resources can be fully utilized and well allocated among the whole society Numerous initiatives on service outsourcing have provided incentives for organizations to become more benign. One popular area that continues to gain importance is one that focuses on the external relationships among organizations. Service composition plays an important role in enterprise integration. From the customers' point of view, it is an optimization problem with reliability, speed, quality and cost of composition as the objectives, which can be transformed into a maximum flow problem. From the service providers' point of view, it will be a maximum cover problem. (Cui, et al., 2009) proposed a schedule of service composition based on network structure. Two services are connected if they share common inputs or outputs. Figure 8-3 shows a service network with betweenness centrality analysis. The nodes with larger diameter have large betweenness, indicating the popularity of these services. (Engelen, 2005) studied the web service repository of XMethod, and claimed that web service network is scale free.



(Nodes are the web services, and there is a link between two services if the input of one service can be used as the output of another service)

Figure 8-3 Web service centrality analysis

IT connectivity and reach have enabled the world to use services extensively through web connectivity. Though web services may not exist exactly the way they are in today, the services business is exponentially expanding and the need for a theoretical rigor and developing services in many application areas is becoming increasingly important. Though some applications are reported in this chapter, there is tremendous opportunity in the near future for developing varieties of web service applications in diverse areas.

Chapter 9. Conclusions and Future Work

9.1 Conclusions

This dissertation deals with the mathematical aspects of service networks and service composition including Mathematical Foundation of Service Composition(Chapter 4), Concept Service (CS) Network Matrix and Service Composition(Chapter 5), Service Composition based on Multi-criteria Goal Programming(Chapter 6), and Real Time Stochastic Solution using CS Network Matrix(Chapter 7). The main contributions of this dissertation are as follows:

(1) For the first time in the literature a mathematical theory for service composition to describe the services and concepts and their topological spaces is formulated. Theorems of convergence, existence, and mutuality/duality are studied.

(2) Using the mathematical theory of service composition developed in this dissertation, a Concept Service (CS) network matrix is defined and explored in detail. This CS network matrix is a useful tool for performing service composition. Concept Service (CS) network based service composition algorithms are generated as a fast service composition tool. Network structure analysis including Components and distance information helps service computing. Centrality identifies important services. Network dynamics helps us dig the hidden information by analyzing the

characteristics of components, distance, centrality, bridge etc in the network. These methods can be used as a preprocess in service computing.

(3) A set of multi-criteria goal programming models that considers both functional and QoS(Quality of Service) criteria is developed. These models are built to find the optimal and compromise solutions according to the QoS preferences. Three different models (MCP ─ Multi Criteria Programming, MCGPO ─ Multi Criteria Goal Programming for Optimal composition and MCGPN ─ Multi Criteria Goal Programming for Non-optimal composition) are explored, and the performances of these models are discussed. MCP and MCGPO models can be used to find optimal compositions. MCGPN models can be used to find acceptable non-optimal compositions. Both MCGPO and MCGPN models can give a compromise composition when MCP does not have a solution that satisfies both functional and nonfunctional (QoS) requirements of the query. A summary of all the six models developed in this work is reported in Table 6-4 Characterization of the models.

(4) Stochastic modeling technique in service composition to consider uncertainty is developed based on the potential of Concept Service (CS) network matrix. The stochastic models are explored to perform service composition under uncertainty. In these models, the nested compound work flow is taken into consideration, and the workload of each web service is considered.

9.2 Future Work

In the future, the growth of Concept Service (CS) networks, e.g., how new services and concepts can be added into the existing network will be useful to explore further. Next, the flow conservation will be worthwhile to be considered in the service network dynamics. It will also be useful to analyze how to design new services so that they can have a positive influence on the entire network and improve market share. The experiments under varieties of conditions can be run to further test the performance of each model. In addition, new models can be developed for the condition that the cardinality of concepts and services is infinite.

Bibliography

Ahmed, T. and M. Mushtaq. 2007. P2P Object-based adaptive Multimedia Streaming (POEMS). Journal of Network and Systems Management. 15, 3, 289-310.

Akinci, B., H. Karimi, A. Pradhan, C. Wu and G. Fichtl. 2008. CAD and GIS Interoperability through Semantic WEB Services. ITcon, 13, 39-56.

Albert, R., A. L. Barabási, H. Jeong and G. Bianconi, 2000. Power-law distribution of the World Wide Web. Science 287, 2115a.

Albert, R. and A. L. Barabási. 2000. Dynamics of complex systems: Scaling laws for the period of Boolean Networks. Physical Review Letters 84, 5660-5663.

Albert, R. and A. L. Barabási. 2002. Statistical mechanics of complex networks. Reviews of Modern Physics, 74, 1, 48-94.

Albert, R., H. Jeong and A. L. Barabási. 2000. Error and attack tolerance in complex networks. Nature, 406-378.

Albert, R., H. Jeong and A. L. Barabási. 1999. Diameter of the World Wide Web. Nature 401, 130-131.

Albert, R., I. Albert and G. L. Nakarado. 2004. Structural Vulnerability of the North American Power Grid. Phys. Rev. E 69, 025103(R).

Alon, U. 2007. Network motifs: theory and experimental approaches. Nature Reviews Genetics, 8, 450-461.

Annapureddy, S., C. Gkantsidis, P. Rodriguez and L. Massoulie. 2005. Providing Video-on-Demand using Peer-to-Peer Networks. Microsoft Technical Report, MSR-TR-2005-147.

Aral, S. and D. Walker. 2010. Viral Product design: Testing social contagion and peer influence effects in networks using randomized trials. working paper.

Arthur, J. and A. Ravindran. 1980. A Partitioning Algorithm for (Linear) Goal Programming Problems. ACM Transactions on Mathematical Software, 6, 3, 378-386.

Aydın, O., N. K. Cicekli and I. Cicekli. 2008. Automated Web Services Composition with the Event Calculus. Engineering Societies in the Agents World VIII, Springer Berlin, 4995/2008, 142-157.

Baggio, R., N. Scott and C. Cooper. 2010. NETWORK SCIENCE: A review focused on tourism. http://cdsweb.cern.ch/record/1245639.

Baker, A. 2002. Matrix Groups: An Introduction to Lie Group Theory. Springer undergraduate mathematics series, Springer-Verlag, London.

Barabási, A. L. 2009. Scale-free networks: A decade and beyond. Science 2009, 412-413.

Barabási, A. L., R. Albert and H. Jeong. 2000. Scale-free characteristics of random networks: The topology of the World Wide Web. Physica A 281, 69-77.

Barabási, A. L. and R. Albert. 1999. Emergence of scaling in random networks. Science 286, 509-512.

Bellman, R. E. 1957. Dynamic Programming. Princeton University Press,Princeton, New Jersey.

Braunstein, L. 2010. Effects of degree correlations to the pressure congestion on complex networks. NetSci conference, Boston, May, 2010.

Bravetti, M. and G. Zavattaro. 2007. Towards a Unifying Theory for Choreography Conformance and Contract Compliance. Software Composition, Springer Berlin, 4829/2007, 34-50.

Bravetti, M. and G. Zavattaro. 2007. Towards a Unifying Theory for Choreography Conformance and Contract Compliance. Software Composition, Springer Berlin, 4829/2007, 34-50.

Calamoneri, T., A. Clementi, M. Lauria, A. Monti and R. Silvestri. 2008. Minimum-energy broadcast and disk cover in grid wireless networks. Theoretical Computer Science, 399, 1-2, 38-53.

Canfora, G.,M. Di Penta, R. Esposito, and M. L. Villani. 2004. A Lightweight Approach for QoS-Aware Service Composition, 2nd International Conference on Service Oriented Computing(ICSOC), ACM Press, New York, 63-63.

Chan, K.S.M., J. Bishop and L. Baresi. 2005. Survey and comparison of Planning Techniques for Web Services Composition. Semantic Web Services and Web Process Composition, Springer Berlin, 43-54.

Chan, K. and J. Bishop. A Self-healing Cycle for Web Service Composition, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.3918.

Chen, C. L. 2004. Multi-objective optimization of multi-echelon supply chain. Computers & Chemical Engineering, 28, 6-7, 1131-1144.

Cohen, B. Bit Torrent. http://www.bittorrent.com/

Crofts, J. 2010. Discovering hierarchical and asymmetric network structure, with applications in neuroscience. NetSci conference, Boston, USA, May 2010.

Cui, L.Y., S.R.T. Kumara and T. Yao. 2009. Service composition based on networks and mathematical programming. INFORMS annual conference, San Diego, USA, Oct., 2009.

Cui, L.Y. and S. R. T. Kumara. 2010. Binary code heredity and provenance detection based on networks. NetSci conference, Boston, USA, May, 2010.

Cui, L., S. Kumara, D. W. Lee. 2011. Scenario Analysis of Web Service Composition based on Multi-Criteria Mathematical Goal Programming, Service Science Vol.3, No.3, 2011 (in printing)

Cui, L., S. Kumara, J. Yoo, F. and Cavdur. 2009. Large-Scale Network Decomposition and Mathematical Programming Based Web Service Composition. E-Commerce Technology, IEEE International Conference on, 511-514.

Cui, L., S. Kumara and R. Albert. 2010. Complex Networks: An Engineering View. IEEE Circuits and Systems Magzine, 10, 3, 10-25.

Den Briel, M.V. and S. Kambhampati. 2005. Optiplan: Unifying IP-based and Graph-based Planning. J. of Artificial Intelligence Research, 24, 919-931.

Digiampietri, L. A., J. J. Pérez-Alcázar and C. B. Medeiros. 2007. AI Planning in Web Services Composition: a review of current approaches and a new solution. SBC 2007, 983-992.

Digiampietri, L., J. Alcázar, C. Medeiros and J. Setubal. 2006. A framework based on semantic Web services and AI planning for the management of bioinformatics scientific workflows. Technical report, 2006.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269-271.

Documentation for JSHOP 2.0. http://www.cs.umd.edu/projects/shop, retrieved in Dec. 2005.

Doshi, P., R. Goodwin, R. Akkiraju and K. Verma. 2005. Dynamic Workflow Composition: Using Markov Decision Processes. JWSR, 2, 1, 1-17.

Engelen, R.V. 2005. "Are web services scale free?" http://www.cs.fsu.edu/~engelen/powerlaw.html.

Faloutaos, M., P. Faloutsos and C. Faloutsos. 1999. On power-law relationship of the internet topology. Computer Communication Review, 29, 4, 251-262.

Fensel, D., M. Kifer , J. de Bruijn and J. Domingue. 2005. Web service modeling ontology (wsmo) submission. w3c member submission, 2005.

Finzi, A. and T. Lukasiewicz. 2004. Relational Markov Games. Logics in Artificial Intelligence, Springer Berlin, 3229/2004, 320-333.

Fritz, C., J. A. Baier, S. A. McIlraith. 2008. ConGolog, Sin Trans: Compiling ConGolog into Basic Action Theories for Planning and Beyond. Eleventh International Conference on Principles of Knowledge Representation and Reasoning, 600-610.

Gabaldon, A. 2002. Non-markovian control in the situation calculus. Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02), 519-524.

Gao, A., D. Yang, S. Tang and M. Zhang. 2005. Web Service Composition Using Integer Programming-based Models. WS Proceedings of the 2005 IEEE International Conference on e-Business Engineering (ICEBE'05), 2005.

Gao, A., D. Yang, S. Tang and M. Zhang. 2005. Web Service Composition Using Markov Decision Processes. Advances in Web-Age Information Management, Springer Berlin, 3739/2005, 308-319.

Ghallab, M., D. Nau and P. Traverso. 2004. Automated Planning: Theory and Practice. Morgan Kaufmann.

Glossary of US Government Accountability Office, 2005. http://www.gao.gov/policy/itguide/glossary.htm, retrieved in Dec. 2005.

Gnanasambandam, N., S. Lee, N. Gautam, S.R.T. Kumara, W. Peng, V. Manikonda, M. Brinn and M. Greaves. 2004. Reliable MAS performance evaluation using queueing models. IEEE Multi-agent security and survivability symposium, August 2004.

Grady, D., C. Thiemann and D. Brockmann. 2010. The tomography of human mobility-What do shortest-path trees reveal? APS meeting, March, 2010.

Gross, J.L. and J. Yellen. 2004. Handbook of Graph Theory, CRC Press.

H. C.-L and K. Yoon. 1981. Multiple Criteria Decision Making. Lecture Notes in Economics and Mathematical Systems, Springer-Verlag.

Hidalgo, C. 2009. The dynamics of economic complexity and the product space over a 42 year period. working paper.

Hong, Y., N. Gautam, S.R.T. Kumara, A. Surana, H. Gupta, S. Lee, V. Narayanan and H. Thadakamalla. 2002. Survivability of Complex System – Support Vector Machine Based Approach. International conference on Artificial Neural Networks in Engineering (ANNIE), 153-158.

Howard, R. 1960. Dynamic Programming and Markov Decision Processes. MIT Press.

http://www.uddi.org/pubs/DataStructure-V1.00-Published-20020628.pdf

http://www.w3.org/TR/wsdl20.

http://www-128.ibm.com/developerworks/webservices

Hu, Q. and J. Liu. 2000. An introduction to Markov Decision Processes(in Chinese). Xidian University Press.

Huang, S., H. Chen and L. Zhang. 2005. Progressive Auction Based Resource Allocation in Service-Oriented Architecture. Proceedings of the 2005 IEEE International Conference on Services Computing, July, 2005, 2, 85-92.

Hwang, S., E. Lim, C. Lee and C. Chen. 2008. Dynamic Web Service Selection for Reliable Web Service Composition. IEEE Transactions on Services Computing, 1, 1, 104-116.

Jeong, H., Z. Neda and A. L. Barabási. 2003. Measuring preferential attachment for evolving networks. Europhysics Letters 61, 567-572.

K¨usters, R. 2001. Non-Standard Inferences in Description Logics. Volume 2100 of Lecture Notes in Computer Science, Springer.

Kaelbling, L. P., M. L. Littman and A. R. Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence, 10, 1- 2, 99-134.

Karloff, H. 1991. Linear Programming. Birkhauser.

Katz, L. 1953. A new Status Index Derived from Sociometric Analysis. Psychometrika 18, 1, 39-43.

Kautz, H and J. P. Walser. 1999. State-Space Planning by Integer Optimization. Proc. of American Association of Artificial Intelligence, 526-533.

KaZaA. http://www.kazaa.com/

Khot, S. and O. Regev. 2003. Vertex cover might be hard to approximate to within 2-ϵ. Proceedings of the 18th Annual conference on Computational Complexity (CCC'03), IEEE Computer Society, 2003, 379-386.

Kleinberg, J. 2000. The small-world phenomenon: An algorithmic perspective. Proc. 32nd ACM Symposium on Theory of Computing.

Kochen, M. and S. Pool. 1978. Contacts and Influences. Social Networks, 1, 1, 5-51.

Kozinets, R. V., A. Hemetsberger and H. J. Schau. 2008. The Wisdom of Consumer Crowds: Collective Innovation in the Age of Networked Marketing. Journal of Macromarketing, 28, 4, 339-354.

Krioukov, D., F. Papadopoulos, M. Boguna and A. Vahdat. 2008. Arxiv preprint, arxiv: 0805. 1266.

Kruskal, J. B. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proceedings of the American Mathematical Society, 7, 1, 48-50.

Kuter, U., D. Nau, M. Pistore and P. Traverso. 2005. A Hierarchical Task-Network Planner based on Symbolic Model Checking. Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling, 2005, 300-309.

Kuter, U. and D. S. Nau. 2004. Forward-Chaining Planning in Nondeterministic Domains. Proceedings of the AAAI, 2004, 513-518.

L´ecu´e, F. and A. L´eger.2006 Semantic web service composition through a matchmaking of domain. 4th IEEE European Conference on Web Services (ECOWS), 2006.

Lapouchnian, A. and Y. Lespérance. 2002. Interfacing Indigolog and OAA: A Toolkit for Advanced Multiagent Applications. Applied Artificial Intelligence, 16:9, 813-829.

Lécué, F. and A. Léger. 2006. A Formal Model for Semantic Web Service Composition. Advances in Web-Age Information Management, Springer Berlin, 4273/2006, 385-398.

Lee, S.H., P J. Kim, YY Ahn and H Jeong. 2007. Googling social interactions: Web search engine based social network construction. Arixv preprint, Arxiv: 0710.3268.

Levesque, H. J., R. Reiter, Y. Lesperance, F. Lin and R. B. Scherl. 1997. GOLOG: A Logic Programming Language for Dynamic Domains. Journal of Logic Programming, 31, 1-3, 59-83.

Li, Y., J. Huai, H. Sun, T. Deng and H. Guo. 2008. PASS: An Approach to Personalized Automated Service Composition. 2008 IEEE International Conference on Services Computing, 283-290.

Li, L. and L. Horrocks. A software framework for matchmaking based on Semantic Web Technology. International Journal of Electronic Commerce, 8, 4, 2004

Liu, K. 2004. Applied Markov Decision Processes(in Chinese). Tsinghua Press, Beijing, 21-21.

Lu, L. and T. Zhou. 2010. Link Prediction in Complex Networks: A survey. Preprint by Elsevier Science.

Lufei, H., W. Weisong and V. Chaudhary. 2008. Adaptive Secure Access to Remote Services in Mobile Environments. IEEE Transactions on Services Computing, 1, 1, 49-61.

Luo Z. and J. Li. 2005. A Web Services Provisioning Optimization Model in a Web Services Community. ICEBE 2005, 689-696.

Magee J. and J. Kramer. 1999. Concurrency - State Models and Java Programs, John Wiley.

Martínez, E. and Y. Lespérance. 2004. IG-JADE-PKSlib: An Agent-Based Framework for Advanced Web Service Composition and Provisioning. Proceedings of the AAMAS-2004 Workshop on Web Services and Agent-Based Engineering, 2-10.

McAllester, D. and D. Rosenblitt. 1991. Systematic nonlinear planning, AAAI (1991), Menlo Park, CA, 634-639.

McCarthy, J. and P. J. Hayes. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence, 4, 463-502.

McDermott, D.V. 2002. Estimated-Regression Planning for Interactions with Web Services. Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, 2002, 204-211.

McIlraith, S. and R. Fadel. 2002. Planning with Complex Actions. Proceedings of the 9th International Workshop on Non-Monotonic Reasoning, 2002, 356-364.

McIlraith, S. and T. C. Son. 2002. Adapting Golog for Composition of Semantic Web Services. Proceedings of the 8th International Conference on Principles and Knowledge Representation and Reasoning, 2002, 482-496.

Milo, R., S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon. 2002. Network motifs: Simple building blocks of complex networks. Science, 298, 5594, 824-827.

Montagut, F. and R. Molva. 2008. Bridging Security and Fault Management within Distributed Workflow Management Systems IEEE Transactions on Services Computing, 1, 1, 33-48.

Moon, S.K., T. W. Simpson, L.Y. Cui and S. R. T. Kumara. 2010. A Service based Platform Design Method for Customized Products. Proceedings of CIRP Integrated Production and Service Systems –II, Sweden, April, 2010.

Morrison, A., P. Lynch and N. Johns. 2004. International tourism networks. International Journal of Contemporary Hospitality Management, 16, 3, 197-202.

Motter, A. E. 2004. Cascade control and defense in complex networks. Phys. Rev. Lett. 93, 098701.

Motter, A. E. 2010. Improved network performance via antagonism: From synthetic rescues to multi-drug combinations. BioEssays, 32, 236-245.

Motter, A. E. and Y.C. Lai. 2002. Cascade-based attacks on complex networks. Phys. Rev. E 66, 065102(R).

Newman, M .E. J. 2000. Models of small world. J. Statistical Physics, 101, 819-841.

Nguyen, D., T. Nguyen and X. Yang. 2007. Multimedia Wireless Transmission with Network Coding. Packet Video 2007, Nov. 2007, 326-335.

Oh, S.-C., D. Lee and S. R. T. Kumara. 2006. WSBen: A Web Services Discovery and Composition Benchmark. IEEE Int'l Conf. on Web Service (ICWS), Chicago, USA, September, 2006.

Oh, S.-C., D. Lee and S. R. T. Kumara. 2008. Effective Web Service Composition in Diverse and Large-Scale Service Networks. Transactions on Services Computing, 1, 1, 15-32.

Oh, S.-C., H. Kil, D. Lee and S. R. T. Kumara. 2006. WSBen: A Web Services Discovery and Composition Benchmark. IEEE Int'l Conf. on Web Service (ICWS), Chicago, USA, September, 2006.

Pacifici, G., M. Spreitzer, A. N. Tantawi and A. Youssef. 2005. Performance management for Cluster-Based Web Services. IEEE journal on selected areas in communications, 23, 12.

Pacifici, G., M. Spreitzer, A. N. Tantawi, A. Youssef. 2005. Performance Management for Cluster-Based Web Services. IEEE journal on selected areas in communications, 23, 12, 247-261.

Paolucci, M., K. Sycara and T. Kawamura. 2003. Delivering Semantic Web Services. Proceedings of the 12th International World Wide Web Conference, 2003, 111-118.

Paolucci, M., T. Kawamura, T. Payne and K. Sycara. 2002. Semantic matching of web services capabilities. Proceedings of the First International Semantic Web Conference, LNCS 2342, Springer-Verlag, 333-347.

Percus, A., G. Istrate and C. Moore. 2006. Computational Complexity and Statistical Physics. Oxford.

Phan, K. A., Z. Tari and P. Bertok. 2008. Similarity-Based SOAP Multicast Protocol to Reduce Bandwith and Latency in Web Services. IEEE Transactions on Services Computing, 1, 1, 88-103.

Pistore, M., A. Marconi, P. Bertoli and P. Traverso. 2005. Automated Composition of Web Services by Planning at the Knowledge Level. International Joint Conference on Artificial Intelligence, 2005.

Pistore, M., P. Roberti and P. Traverso. 2005. Process-level composition of executable web services: "on-the-fly" versus "once-for-all" composition. ESWC (2005), 62-77.

Pistore, M., P. Traverso and P. Bertoli. 2004. Planning and Monitoring Web Service Composition. Presented at AIMSA 2004, Varna, Bulgaria.

Pistore, M., P. Traverso and P. Bertoli. 2005. Automated Composition of Web Services by Planning in Asynchronous Domains. Proceedings of the 15th International Conference on Automated Planning and Scheduling, 2005, 2-11.

Puterman, M. L. 1994. Markov Decision Processes Discrete Stochastic Dynamic Programming. John Wiley and Sons, Inc., New York, NY.

Qiu, R.G. 2004. Manufacturing Grid: A Next Generation Manufacturing Model. Proceeding of 2004 IEEE International Conference on Systems, man and Cybernetics, 4667-4672.

Qiu, R.G. 2009. Computational Thinking of Service Systems: Dynamics and Adaptiveness Modeling. Service Science, 1, 1, 42-45.

Raghavan, U.N., R. Albert and S. R.T. Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. 76, 3: 036106, pp 1-11.

Rao, J., P. Kungas and M. Matskin. 2003. Application of Linear Logic to Web Service Composition. Proc. of the 1st International Conference on Web Services, 2003.

Reichardt, J. 2010. Mesoscopic structure in complex networks. NetSci conference, Boston, USA, May, 2010.

Reiter, R. 2001. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge, Massachusetts.

Russell, S. and P. Norvig. 1995. Artificial Intelligence: a modern approach. Prentice-Hall.

Salkin, H. M. and K. Mathur. 1989. Foundations of Integer Programming. Elsevier Science Publishing Co. Inc.

Sinatra, R. 2010. Networks of motifs from sequences of symbols. NetSci conference, Boston, May, 2010.

Singh, M.P., M.N. Hugns. 2005. Service-Oriented Computing: Semantics, Processes, Agents. John Wiley and Sons, Ltd, West Sussex, England.

Sirin, E., B. Parsia, D. Wu, J. Hendler and D. Nau. 2004. HTN Planning for Web Service Composition Using SHOP2. Journal of Web Semantics, 1, 4, 377-396.

Sirin, E., J. Hendler and B. Parsia. 2003. Semi-Automatic Composition of Web Services Using Semantic Descriptions. Proc. of the International Workshop on Web Services: Modeling, Architecture and Infrastructure Workshop in conjunction with ICEIS, 2003.

Slijepcevic, S. and M. Potkonjak. 2001. Power efficient organization of wireless sensor networks. IEEE International Conference on Communications, 2, 472-476.

Song, C., Z. Qu, N. Blumm and A.L. Barabási. 2010. Limits of predictability in human mobility. Science, 327, 5968, 1018-1021.

Thadakamalla, H.P., S.R.T. Kumara and R. Albert. 2008. Complexity and Large-Scale Networks. Operations Research and Management Science Handbook(edited by A. Ravindran), Chapter 11, CRC Press.

Thadakamalla, H.P., U.N. Raghavan, S.R.T. Kumara and R. Albert. 2004. Survivability of multiagent based supply networks: A Topological perspective. IEEE Intelligent Systems, 19, 5, 24-31.

The Language of the Fourth International Planning Competition. ls5-www.cs.uni-dortmund.de/~edelkamp/ipc-4/, retrieved in Dec. 2005.

Trainotti, M., M. Pistore, F. Barbon, P. Bertoli, A. Marconi, P. Traverso and G. Zacco. 2006. ASTRO: Supporting Web Service Development by Automated Composition, Monitoring and Verfication. Proceedings of the 16th International Conference on Automated Planning and Scheduling (Software Demonstration), 2006, 28-31.

Trainotti, M., M. Pistorem, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli and P. Traverso. 2005. ASTRO: Supporting Composition and Execution of Web Services. Proceedings of the ICSOC 2005, 495-501.

Vossen, T., M. Ball, A. Lotem, and D. Nau. 1999. On the Use of Integer Programming Models in AI Planning Proc. of Int'l Joint Conf. on Artificial Intelligence, 304-309.

Vossen, T., M. Ball, A. Lotem, and D. Nau. 2000. Applying Integer Programming to AI Planning. it The Knowledge Engineering Review, 15, 1, 85-100.

W3C. 2003. Web Services Description Language (WSDL) Version 2.0. W3C Working Draft, March 2003, http://www.w3.org/TR/wsdl20.

Wadhwa, V. and A. R. Ravindran. 2007. Vendor selection in outsourcing. Computers& Operations Research 34, 3725-3737.

Watts, D.J. 2004. The "New" Science of Networks. Annual Review of Sociology 30, 243-270.

Watts, D.J., P.S. Dodds and M.E.J. Newman. 2002. Identity and search in social networks. Science, 296, 17.

Watts, D.J. and S.H. Strogatz. 1998. Collective dynamics of "small-world" networks. Nature 393 (6684): 409–10. doi:10.1038/30918.

Wei, J., L. Singaravelu and C. Pu. 2008. A Secure Information Flow Architecture for Web Service Platforms. IEEE Transactions on Services Computing, 1, 1, 75-87.

White, D. J. 1993. Markov Decision Processes. Wiley, 1-96.

Whitney, D. E. 2009. A dynamic model of cascades on random networks with a threshold rule. Arxiv preprint, arxiv: 0911.4499.

Wu, D. and B. Parsia. 2003. Automating DAML-S Web Services Composition Using SHOP2. Proceedings of ISWC 2003, Sanibel, Island, FL, USA, October 2003, 195-210.

Wu, Z., A. Ranabahu, K. Gomadam, A. P. Sheth and J. A. Miller. Automatic Semantic Web Services Composition. Online sharing document http://www.cs.uga.edu/~jam/papers/zLSDISpapers/zixin.doc

Xiao, L., L. Zhang, G. Huang and B. Shi. 2004. Automatic Mapping from XML Documents to Ontologies. Proceedings of the Fourth International Conference on Computer and Information Technology, September, 2004, 321-325.

Xu, X., Y Wang, S, Panwar and K. W. Ross. 2004. A peer-to-peer video-on-demand system using multiple description coding and server diversity. Image Processing, 2004. ICIP '04. 2004 International Conference, 3, 1759- 1762.

Yan, Y., Y. Liang and H. Liang. 2006. Composing Business Processes with Partial Observable Problem Space in Web Services Environments. 4th IEEE International Conference on Web Services, 2006.

Yikldirim, M.B. 2008. Network optimization. Operations Research and Management Science Handbook (edited by A. Ravindran), Chapter 4, CRC Press pp (4:1)-(4:20).

Yoo, J., S. Kumara, D. Lee and S.-C. Oh. 2008. A Web Service Composition Framework Based on Integer Programming with Non-Functional Objectives and Constraints. IEEE CEC&EEE, Washington DC., USA, 2008.

Zeng, L., B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang. 2004. QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering, 30, 5, 311-327.

Zeng, L., B. Benatallah, M. Dumas, J. Kalagnanam and Q. Z. Sheng. 2003. Quality Driven Web Services Composition. Proceedings of the 12th international conference on World Wide Web (WWW), Budapest, Hungary. ACM Press, May 2003.

Zhang, J., S. Zhang, J. Cao and Y. Mou. 2004. Improved HTN Planning Approach for Service Composition. Proceedings of the 2004 IEEE International Conference on Service Computing, 609-612.

Zhang, L., S. Cheng, Y. Chee, A. Allam and Q. Zhou. 2007. Pattern Recognition Based Adaptive Categorization Technique and Solution for Services Selection. Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, December, 2007, 535-543.

Zhao, H. and P. Doshi. 2006. A Hierarchical Framework for Composing Nested Web Processes. Lecture Notes In Computer Science, NUMB 4294, 116-128.

VITA

Liying Cui

Liying Cui received her B.S. degree in Mathematics and M.S. degree in Operations Research and Control from Tianjin University, China. Liying's professional experience includes: Marketing intelligence with Kimberly Clark, in USA; and Enterprise Resource Management of Tianjin company of Sinopec Group with Crisen Tech in China. Her research experience includes service computing, fraud detection, hotel management, binary code heredity detection, mesh algorithm for large scale EM simulation and revenue management. She received Bronze Medal at the "Hope Cup" National Mathematical Competition in China in 1995. Recently, she received Kimberly Clark Excellence Award and INFORMS Service Science Finalist Best Paper Award in 2010, and IERC best paper award in computer and information systems in 2011.