Scenario Analysis of Web Service Composition based on Multi-Criteria Mathematical Goal Programming

LiYing Cui
Product Supply and Operations, KC International, Kimberly-Clark
1400 Holcomb Bridge Rd, Roswell, GA 300076, USA
LiYing.Cui@kcc.com

Soundar Kumara
Department of Industrial Engineering, The Pennsylvania State University
310 Leonhard Building, University Park, PA 16802, USA
skumara@psu.edu

Dongwon Lee
College of Information Sciences and Technology, The Pennsylvania State University
313A INFO SCI and TECH Building, University Park, PA 16802, USA
dongwon@psu.edu

This paper addresses the web service composition problem considering multi-criteria regarding quality of services (QoS). Three different scenarios of multi-criteria mathematical programming models are explored under the 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. The 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) Compromised optimal, and (3) Acceptable. This set of multi-criteria models have both advantages and disadvantages comparing 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 are 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):Compromised optimal and (3)Acceptable respectively. And they can find a compromised 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.

Key words: web service; service composition; multi-criteria mathematical programming; goal programming; manufacturing, enterprise integration
History: Received Oct. 16, 2010; Received in revised form Feb. 22, 2011; Accepted Feb. 14, 2011; Online first publication Dec. 12, 2011

1. Introduction

Web based software is fast becoming an integral part of e-business. In particular, the proliferation of e-commerce and the use of distributed computing in recent years have fueled the explosive growth of web services. Web service composition helps build web-based applications by composing as well as integrating existing web services. In general, web services are diverse in terms of both functionality and heterogeneity of the quality of services (QoS). Consequently, it is necessary to use a general web service composition method which considers a variety of services with diverse constraints. 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. In practice, multiple compositions may provide the same response to user's queries. 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.). In this paper, we refer to such a problem as web service composition problem with multi-criteria regarding QoS. For example, consider that we have three work flows which can meet a customer's query. These work-flows are: A-B-C, F-D-E-H, and A-M-N. Their prices are $500, $500 and

$700 respectively. The execution times are 10 seconds, 8 seconds and 7 seconds respectively. In addition, reliability of performance is 6, 8 and 9. We could assume that less cost, shorter execution time and higher reliability may form the metrics of selection of better work flows. Among the three solutions, it is impossible for the customer to choose a work flow that dominates the other two. If the customer gives the highest priority to cost, the second priority to execution time, and the third priority to reliability, obviously we will consider the second work flow as the optimal solution since it has the shortest execution time between the solutions with the least cost.

This paper addresses the web service composition problem considering multi-criteria regarding quality of services (QoS). In modern business environments, the ability to provide users personalized services has become pivotal competitive factor for enterprise. In addition, recently user preferences also have drawn more and more attention in the service-oriented research field. However, most of the traditional automated service composition approaches cannot meet the personalized user requirements in a flexible manner. The user requirements not only include the hard constraints, but also include the soft constraints. In traditional approaches, only users' initial requirements are considered as constraints to implement planning. However, this kind of hard constraints are not sufficient for describing user requirements in a real business scenario. For example, an user's ideal solution is the fastest conveyance at the lowest cost. Unfortunately, there may be no way to satisfy both of his preferences at all. Thus, he has to make a compromise to select a feasible conveyance. This kind of soft constraints are usually negotiable according to the trade-off. As a matter of fact, user-centered service composition should pay much more attention to flexible user preferences to improve the service level. In addition, soft-constraints will also improve the success rate of service composition thanks to the negotiation feature.

In this paper, we assume 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 web service composition problem by considering different QoS at the same time, we developed three different multi-criteria goal programming models 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 model is not only 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.

The structure of this paper is organized as follows: The literature on web service composition is explored in section 2. Background knowledge required for understanding the proposed method and the solution framework is briefly introduced in section 3. The proposed mathematical models are studied in section 4. The complexity analysis of the models is presented in section 5. Experimental results are shown in section 6. An application of web service computing in manufacturing process integration is shown, and the steps on how to apply the proposed models are explained in section 7. Finally, the conclusions and future work are discussed in section 8.

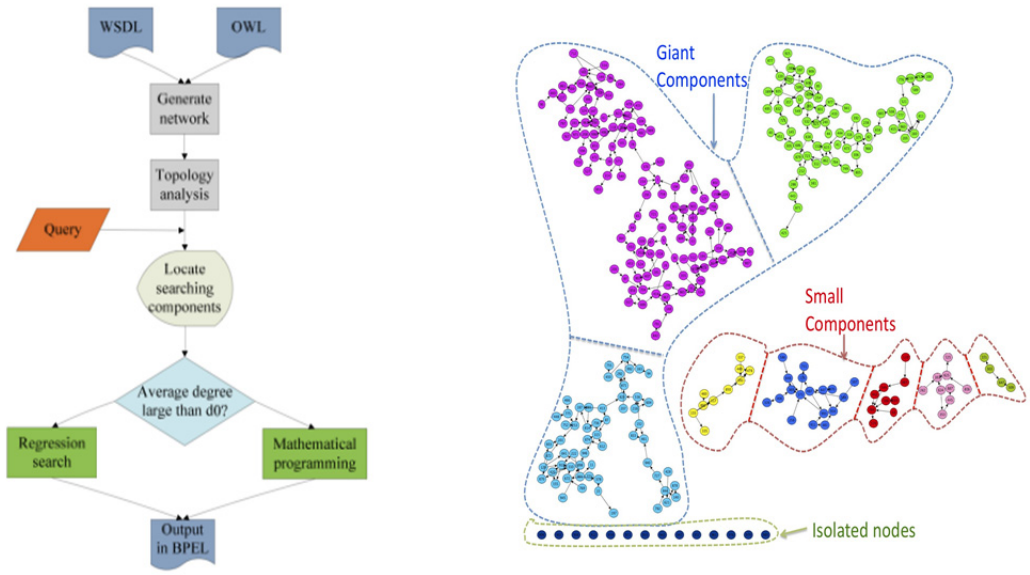
2. Web Service Composition: Background Literature

Considerable amount of research on service composition exists in literature. Zhang et al. (2003) proposed an XML-based UDDI exploration engine. In the same year, 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 in each step of 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. Xiao et al. (2004) proposed an automatic mapping framework based on XML Document Type Definition structure. Qiu (2004) proposed the concept of manufacturing grid by considering services in the scope of next generation collaborative and agile manufacturing model. Huang et al. (2005) proposed a progressive auction based market mechanism for resource allocation that allows users to differentiate service value and ensure the resource acquisition latency. The allocation rule, pricing rule, bidding strategy are explored. 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. (2006, 2007) published AI planning-based frameworks which enables the automatic composition of web services, and developed a novel web service benchmarking tool. Zhang et al. (2007) proposed an architectural framework which enables technology for a business services analyzer to compose and adapt heterogeneous services by pattern identification. Oh et al. (2008) applied large-scale network techniques in web service analysis. Montagut et al. (2008) addressed the security features on executing distributed web services. Lufei et al. (2008) proposed an adaptive secure access mechanism as well as an adaptive function invocation module in mobile environments. Phan et al.(2008) discussed a similarity-based SOAP multicast protocol to reduce bandwidth and latency in web services with high-volume transactions. Wei et al. (2008) studied secure information handling

in web service platform. Sheng et.al. (2009) proposed a web service composition framework and model to allow users to configure web service context and exceptions. Further, Hwang et al. (2008) also 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 services. Segev et.al (2009) introduced a two-step, context-based semantic method to match and rank web services. Qiu (2009) proposed a mechanism to capture users' requirements and satisfaction when decomposing an enterprise process of management and projects into multiple service modules. This concept made it possible that a manufacturing process can be decomposed as global services as we described in an application of our service composition models in section 7.

As for integer programming approaches, Vossen et al. (1999, 2000) and Kautz (1999) initiated the ILP-based approach to AI-planning problems. Zeng et al. (2004) reported a multiple criteria Linear Integer programming model that can be used for QoS features after web service composition. They proposed the issue of service composition by considering the trade-off among preferences of customers as a future work. Gao et al. (2005) modeled service composition by using Integer Programming, which provides a way for capacity planning and load control to be addressed by the service provider. In their work, services with the same function can be compared. In general, web services are diverse in terms of both functionality and homogeneity. Consequently, it is necessary to use a general web service composition method which considers varieties of services. Yoo et al. (2008) formulated the web service composition problem using the Integer Programming AI planning approach based on the methods proposed by Gao et al. (2005) and Briel et al (2005). This algorithm takes care of nonfunctional objectives and constraints compared with the previous work. Rao et al. (2003) applied Linear Logic theorem in the Web Service Composition problem. This approach assumes that core services are already selected by the users, and the functionality does not completely match the users' requirement. Therefore, the approach allows partial involvement of mathematical programming. Kritikos et al.(2009) studied a Mixed-Integer Programming for QoS based matchmaking and experimentally showed that it is better than constraint programming. The technique for finding compromised solution when the query contains multi-objectives is an important research area. Until now, the computational speed and solution quality of generalized mathematical programming models for both functional and QoS based service composition has not been studied thoroughly. This paper addresses the theoretical issues in this domain.

Figure 1: Composition process (a) Composition framework, (b) An example of service networks (different colors represent different subgraphs)



3. Prerequisite and Framework

We assume 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. In this paper, we focus on the topic of service composition.

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 shown in Figure 1(a). Figure 1(b) shows an example of service network. The nodes represent web services. There is a link between two web services if they share either an input or an output attribute. Next, the communities can be detected in the network. When a query comes in, a community or some communities can be assigned as a searching region according to what are given and what are needed in the query.

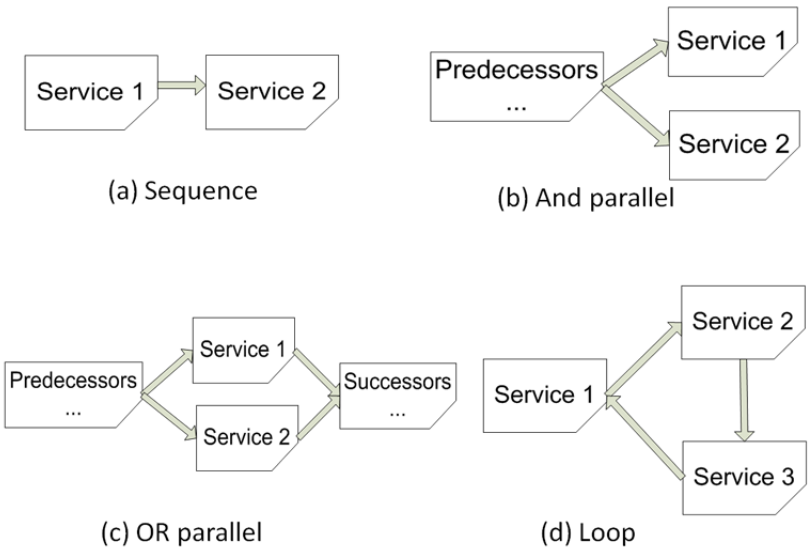
3.1 Quality of Service (QoS)

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 paper can handle many number of QoS metric elements. In this paper, cost, execution time and reliability are used to demonstrate the model. 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.

3.2 Flow Pattern in Service Networks

There are four types of Service flow patterns which include: Sequential, Switch, Parallel and Iterative in web service networks. In Figure 2(a), Service 1 is the predecessor of Service 2, so these two services are sequential. In Figure 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 switch (We call it OR Parallel in this context), with probability p_1, for using Service 1, and with probability p_2 for using Service 2. In Figure 2(c), both Service 1 and Service 2 are needed to provide input information for the successors, so the two services are parallel (We call it AND Parallel in this context). In Figure 2(d), Services 1, 2, and 3 are in a loop, *e.g.*, an iterative flow pattern. We design 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) - to accommodate the four flow patterns discussed.

Figure 2: Flow patterns in Service Networks



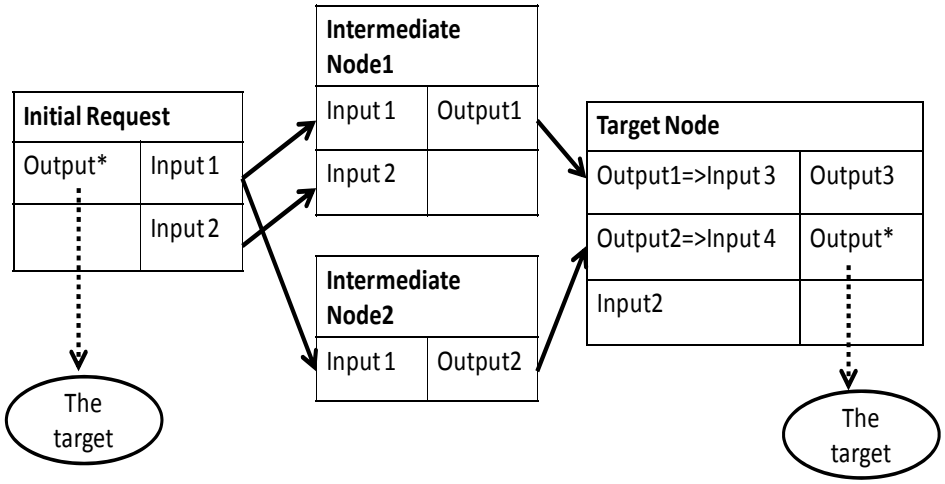
4. Mathematical Modeling

We model web service composition problem with multi-criteria regarding OoS as a multi-criteria program. Especially, goal programming solves the problem in accordance with the customers' preferences and can find a compromised solution when the regular multi-criteria programming model does not have a solution. We consider the customers' preferences on aggregating price, reliability of service, and total service time as goals. Both preemptive and non-preemptive goal programming models are built and tested.

4.1 Attributes of Services

We assume 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, and are categorized by a service interface in WSDL documents that contain 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. 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 3.

Figure 3: An example of service composition



A web service can have multiple inputs as well as outputs. When the service needs to be activated, we assume that all of the inputs are available. In the meantime, some extra outputs those 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.

4.2 Definition of Parameters and Variables of the Model

The entire list of variables used in the paper is shown in Table 1.

Table 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 $n = \max\{|I|,|O|\}$ 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, we give the following definition. 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 $0 < f < 1$ and $0 \le q(f) < +\infty$.

Here, we introduce reliability measure $q(f)$ 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 maximum depth, is important as larger or smaller L influences the computation and whether the optimal solution can be obtained or not.

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

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

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

4.3 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.

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

Service execution time (criterion No.2): The total process 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 1st level is $\max_j \{t_j \cdot Z_{1j}\}$;

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

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

Therefore, 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:

$$\min \sum_{l=1}^{L} \eta_l \tag{2}$$

subject to

$$\eta_l - t_j \cdot Z_{lj} \geq 0 \tag{3}$$

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

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.

$$\max \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot q_j \tag{4}$$

The validity of the first and the second criteria are obvious as we stated above. 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

$$\max \sum_{Z_j \in S} q_j \tag{5}$$

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

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

Now we have: (5) equals to

$$\min \prod_{Z_j \in S} f_j \tag{7}$$

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

Thus, (4) equals to (7).

Therefore, 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. Therefore, (4) is validated.

4.4 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,

$$\sum_{l=1}^{L}\sum_{j=1}^{m} I_{ij} \cdot Z_{lj} \geq I_{i0} \qquad i = 1, 2, \ldots, n. \tag{8}$$

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,

$$\sum_{l=1}^{L}\sum_{j=1}^{m} O_{ij} \cdot Z_{lj} \geq O_{i0} - I_{i0} \qquad i = 1, 2, \ldots, n. \tag{9}$$

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.

$$\sum_{j=1}^{m} I_{ij} \cdot Z_{1j} \leq I_{i0} \qquad i = 1, 2, \ldots, n. \tag{10}$$

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:

$$\sum_{j=1}^{m} I_{ij} \cdot Z_{k+1,j} - \sum_{l=1}^{k}\sum_{j=1}^{m} O_{ij} \cdot Z_{lj} \leq I_{i0} \tag{11}$$

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

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 (11).

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

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j \leq C_0 \tag{12}$$

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,

$$\eta_l \geq Z_{lj} t_j \tag{13}$$

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

$$\sum_{l=1}^{L} \eta_l \le T_0 \tag{14}$$

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

$$Z_{lj} \cdot (q_j - Q_0) \ge 0 \tag{15}$$

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

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.

$$\sum_{l=1}^{L} Z_{lj} \cdot q_j \ge Q_1 \tag{16}$$

8. Non negative and binary constraints:

$$Z_{lj} \ge 0 \qquad Z_{lj} \in \{0,1\} \tag{17}$$

$$\eta_l \ge 0 \tag{18}$$

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

The input-output constraints (8), (9), (10) and (11) can handle both sequential and parallel flow patterns in service network.

We formulate three multi-criteria scenarios, wherein the customers require: 1. Optimal solution, 2. Optimal solution under soe compromise of the QoS, and 3. an acceptable solution with both functional and nonfunctional attributes considered.

4.4.1 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.

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

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:

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j + d_1^- - d_1^+ = C_0$$

$$\sum_{l=1}^{L} \eta_l + d_2^- - d_2^+ = T_0$$

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j + d_3^- - d_3^+ = Q_1$$

$$d_i^-, d_i^+ \ge 0 \quad i = 1, 2, 3$$

and the real constraints (5), (6), (7), (8), (10), (12), (14) and (15).

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**.

We first define 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:

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot p_j + d_1^- - d_1^+ = C_0$$

$$\sum_{l=1}^{L} \eta_l + d_2^- - d_2^+ = T_0$$

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj} \cdot q_j + d_3^- - d_3^+ = Q_1$$

Step-4: 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.

4.4.3 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:

$$\min Z_4 = d_1^+$$
$$\min Z_5 = d_2^+$$
$$\min Z_6 = d_3^-$$

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:

$$\min Z = P_1 \cdot d_1^+ + P_2 \cdot d_2^+ + P_3 \cdot d_3^-$$

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:

$$\min Z = w_1 \cdot d_1^+ + w_2 \cdot d_2^+ + w_3 \cdot d_3^-$$

We 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.

5. Scenario Analysis of the Models

Next, let us discuss the worst case and the best case scenarios. As shown in Figure 4 (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.

5.1 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. $C_v^i \cdot 2^{v-i}$ 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 1 *The time complexity of the weighted average Mixed IP Model is:*

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

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

Table 2 shows the times of comparisons for the goal programming and exhaustive search. Table 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 2 Computational complexity of MILP compared with exhaustive search

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 3 Computational complexity of MILP compared with exhaustive search when $M = 10$ 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:

We present below a comparison of the time complexity of the mixed IP model and exhaustive search 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 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.

5.2 Solution Analysis of the proposed Models

In Section 4, we have introduced six composition models: (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) nonpreemptive 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. We can prove the following relation between the two methods:

Theorem 1: 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 4: 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

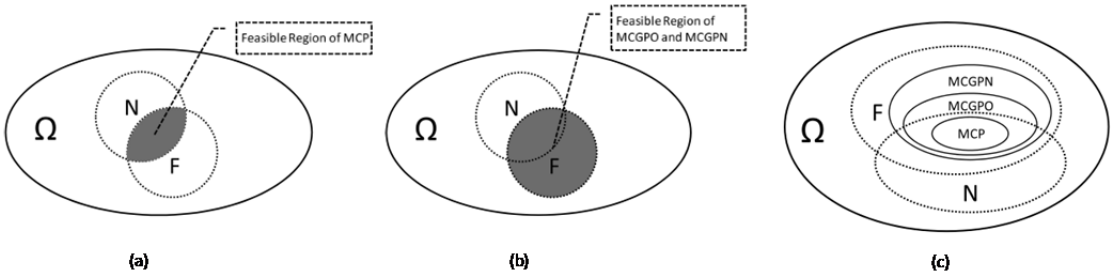
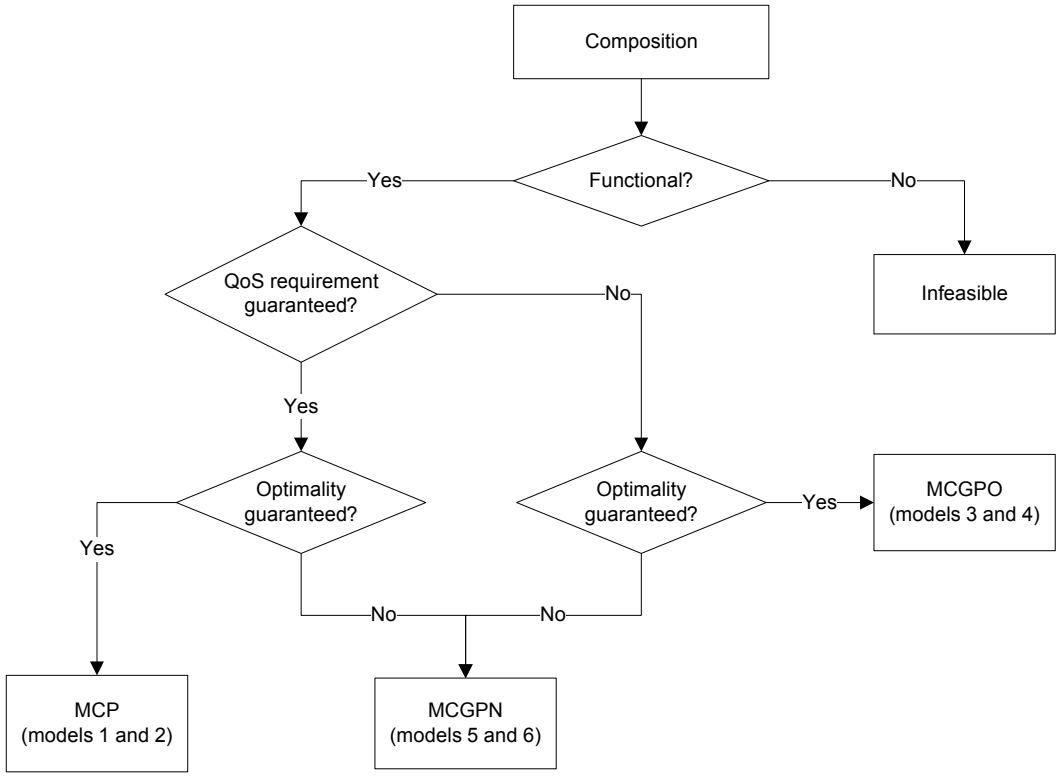


Figure 4 (a) shows the feasible region of MCP model, where Ω represents the entire space, F is the functional feasible space and N is the nonfunctional feasible space. Similarly, Figure 4 (b) shows the feasible region of MCGPO and MCGPN models. We notice that MCGPO and MCGPN models have the same feasible region which is F, and MCP model has a smaller feasible region which is the intersection of F and N. Figure 4 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 5, the quality of the solutions from the MCP, MCGPO and MCGPN models are explored.

Figure 5: 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 4 summarizes the characteristics of the proposed three types of models in the chapter.

Table 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

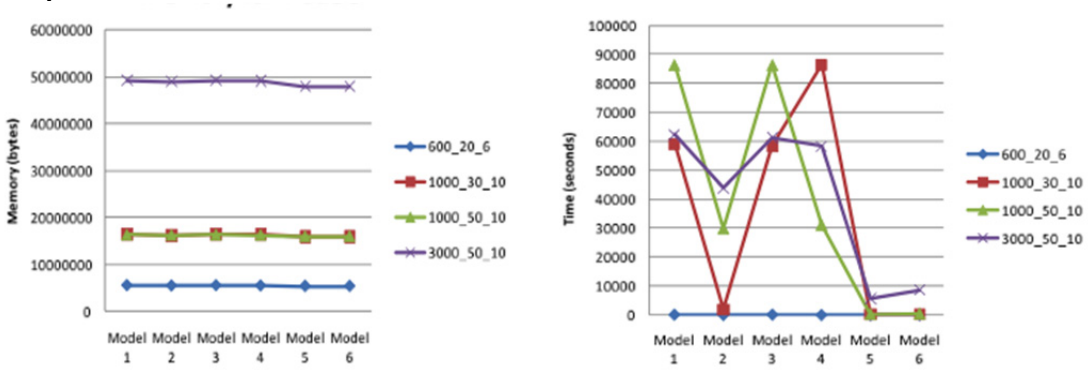
6. Experiments

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, we use the maximum memory use of all the models in all the experiments as 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 7 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.

We assume that the customers give both priorities and the weights to the objectives. Therefore, 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 (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 (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 Experiments



(a) Memory vs. Models **(b) Time vs. Models**

Figure 7 Memory usage performance of the six models in four different experimental sets

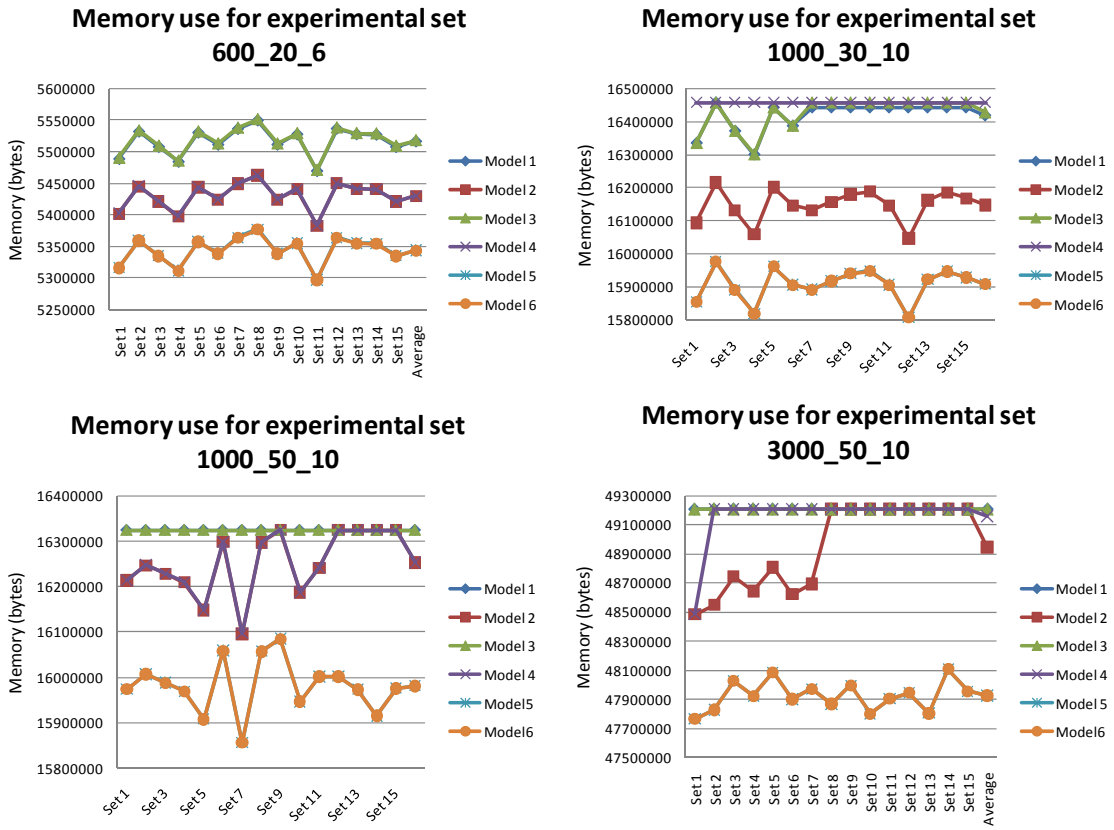
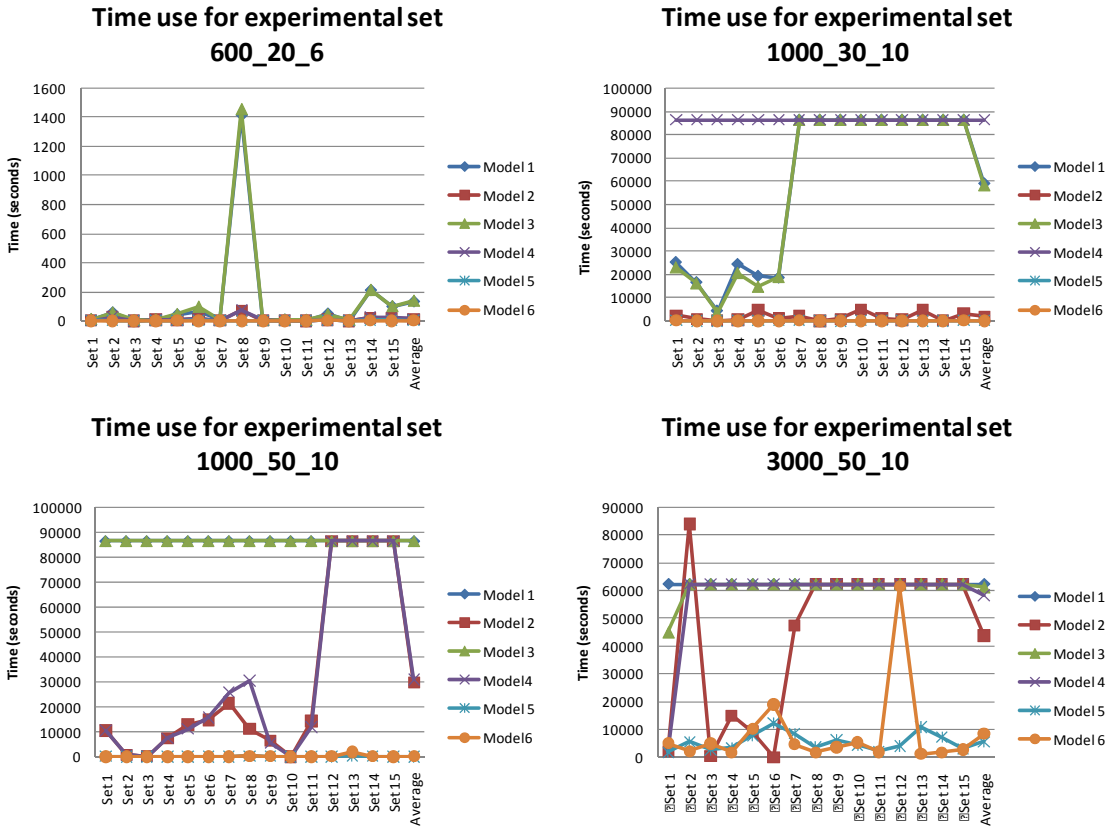


Figure 8 shows the computational time of the six models in four different designs of experiment. Some curves may overlap when their values are close to each other. From the four experiments, we can see that Models 4 and 5 always take less computational time than the other models do; and Models 3 and 1 always take more computational time than the other models do.

Figure 8 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 (b). 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 findings of a solution are important, and we do not care about computational time, we can use Models 3 and 4.

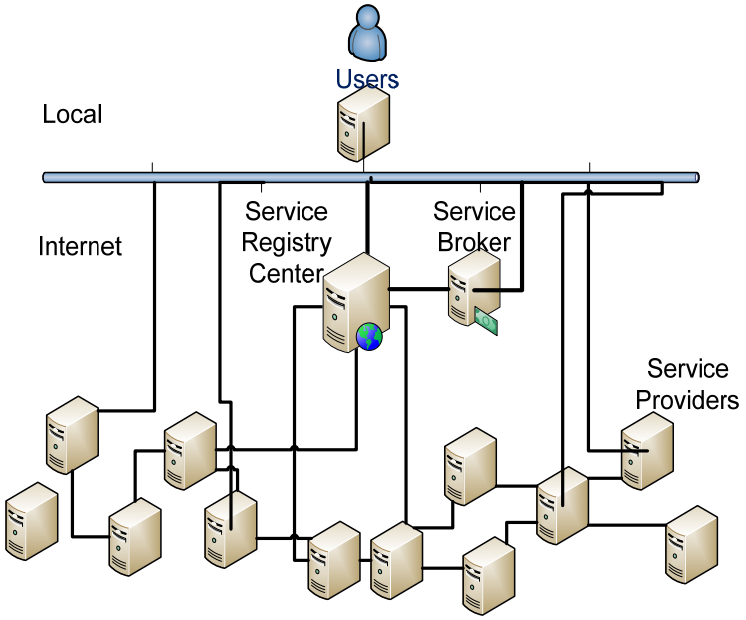
7. An Application to Manufacturing Process Integration

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 Enterprise 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 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 9 shows the web service topology mentioned here.

Figure 9 The topology of users, service providers, service brokers and service registry center



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., $m = 5$.

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

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 $n = 15$.

$$I_1 = [1,1,1,0,0,0,0,0,0,0,0,0,0,0,0];$$
$$O_1 = [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0];$$
$$I_2 = [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0];$$
$$O_2 = [0,0,0,0,0,0,1,1,1,0,0,0,0,0,0];$$
$$I_3 = [0,0,0,0,0,0,1,1,1,0,0,0,0,0,0];$$
$$O_3 = [0,0,0,0,0,0,0,0,0,1,1,1,0,0,0];$$
$$I_4 = [0,0,0,0,0,0,0,0,0,1,1,1,0,0,0];$$
$$O_4 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1];$$
$$I_5 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1];$$
$$O_5 = [0,0,1,0,0,0,0,0,1,0,0,0,0,0,1];$$

Therefore, 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:

$$\begin{aligned}[f_1, f_2, \ldots, f_5)] &= [0.02, 0.02, 0.01, 0.1, 0.1] \\ &= [-\log 0.02, -\log 0.02, -\log 0.01, -\log 0.1, -\log 0.1] \\ &= [1.699, 1.699, 2, 1, 1]\end{aligned}$$

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).

$$\begin{aligned}&[p_1, p_2, p_3, p_4, p_5, C_0] = [115, 114, 92, 101, 49, 550] \\ &\Rightarrow [\frac{115}{550}, \frac{114}{550}, \frac{92}{550}, \frac{101}{550}, \frac{49}{550}, \frac{550}{550}]\end{aligned}$$

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):

$$\begin{aligned}&[t_1, t_2, t_3, t_4, t_5, T_0] = [5, 8, 7, 9, 3, 35] \\ &\Rightarrow [\frac{5}{35}, \frac{8}{35}, \frac{7}{35}, \frac{9}{35}, \frac{3}{35}, \frac{35}{35}]\end{aligned}$$

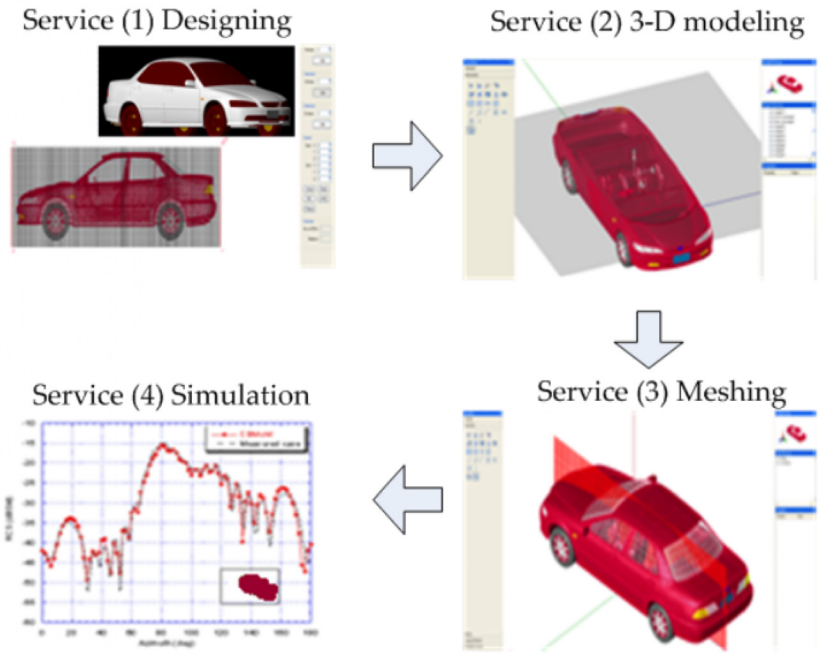
3. Normalize the reliability of the service Z_j, the reliability is defined by the failure rate and the total failure rate limit, by using the maximum value of the elements (which is the Q_1 value 1):

$$\begin{aligned}&[q_1, q_2, \ldots, q_5] = [1.699, 1.699, 2, 1, 1] \\ &\Rightarrow [\frac{1.699}{7.398}, \frac{1.699}{7.398}, \frac{2}{7.398}, \frac{1}{7.398}, \frac{1}{7.398}]\end{aligned}$$

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 10. In the solution, all the artificial variables related with the goal constraints are zero, and the optimal solution is found in the MCGPO model.

Figure 10: The composition chain of the application (car models simulated by GEMS, an EM simulation software)



8. Conclusions and Future Work

In this paper, we have built and analyzed multi-criteria 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 method and non-preemptive method. 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 compromised 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 4.

In the future, it is important to introduce uncertainty in the web services composition model. 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.

Acknowledgements

The authors are grateful to these who offered help and inspiration in writing the paper. We would like to thank Dr. Raj Mittra for proof-reading the first manuscript, and thank Dr. A. Ravi Ravindran for validating the goal programming models proposed in this paper.

References

Arthur, J., A. Ravindran. 1980. A Partitioning Algorithm for (Linear) Goal Programming Problems. *ACM Transactions on Mathematical Software*, **6**, **3**, 378–386.

Cui, L., S. Kumara, R. Albert. 2010. Complex Networks: An Engineering View. *IEEE Circuits and Systems Magazine*, **10**, **3**, 10–25.

Cui, L., S. Kumara, J. Yoo, F. Cavdur. 2009. Large-Scale Network Decomposition and Mathematical Programming Based Web Service Composition. *E-Commerce Technology, IEEE International Conference on*, 511–514.

Den Briel, M.V., S. Kambhampat. 2005. Optiplan: Unifying IP-based and Graph-based Planning. *J. of Artificial Intelligence Research*, **24**, 919–931.

Den Briel, M.V., T. Vossen, S. Kambhampati. 2005. Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework *Proc. of Int'l Conf. on Automated Planning and Scheduling*, 310–319.

Gao, A., D. Yang, S. Tang, 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*.

Hwang, S., E. Lim; C. Lee; C. Chen. 2008. Dynamic Web Service Selection for Reliable Web Service Composition. *IEEE Transactions on Services Computing*, **1**, **1**, 104–116.

Kautz, H., J. P. Walser. 1999. State-Space Planning by Integer Optimization *Proc. of American Association of Artificial Intelligence*, 526–533.

Lufei, H., W. Weisong, V. Chaudhary. 2008. Adaptive Secure Access to Remote Services in Mobile Environments. *IEEE Transactions on Services Computing*, **1**, **1**, 49–61.

Singh, M.P., M.N. Hugns. 2005. Service-Oriented Computing: Semantics, Processes, Agents. *John Wiley and Sons, Ltd, West Sussex, England*.

Montagut, F., R. Molva. 2008. Bridging Security and Fault Management within Distributed Workflow Management Systems *IEEE Transactions on Services Computing*, **1**, **1**, 33–48.

Oh, S.-C., D. Lee, 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, 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, A. Youssef. 2005. Performance Management for Cluster-Based Web Services. *IEEE journal on selected areas in communications*, **23**, **12**, 247–261.

Phan, K.A., Z. Tari, 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.

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.

Rao, J., P. Kungas, M. Matskin. 2003. Application of Linear Logic to Web Service Composition. *Proc. of the 1st International Conference on Web Services, 2003*.

Salkin, H. M., K. Mathur. 1989. Foundations of Integer Programming *Elsevier Science Publishing Co. Inc.*.

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.

Wadhwa, V., A. R. Ravindran. 2007. Vendor selection in outsourcing. *Computers & Operations Research*, **34**, 3725–3737.

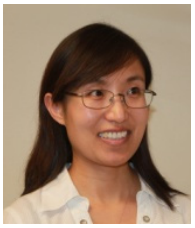
Wei,J., L. Singaravelu; C. Pu. 2008. A Secure Information Flow Architecture for Web Service Platforms. *IEEE Transactions on Services Computing*, **1**, **1**, 75-87.

W3C. 2003. Web Services Description Language (WSDL) Version 2.0. Working Draft. http://www.w3.org/TR/wsdl20.

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.

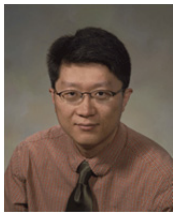
Zhang, L., S. Cheng, Y. Chee, A. Allam, Q. Zhou. 2007. Pattern Recognition Based Adaptive Categorization Technique and Solution for Services Selection. *Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference* 535–543.

Yoo, J., S. Kumara, D. Lee, S.-C. Oh. 2008. A Web Service Composition Framework Based on Integer Programming with Non-Functional Objectives and Constraints. *the IEEE CEC&EEE, Washington DC., USA, 2008*.



LìYing Cūi is an Operations Research Specialist in Kimberly-Clark. She received her B.S. degree in Mathematics and M.S. degree in Operations Research and Control from Tianjin University, China, and Ph.D. degree in Industrial Engineering and Operations Research from the Pennsylvania State University. She is a recipient of Bronze Medal in 1995 at the National Math Competition "Hope Cup" during her 10th grade in China. She received Kimberly Clark Excellence Award in Aug 2010 and the IERC best paper award in CIS track in May 2011. Her research interests are in Service Computing, Retail Intelligence, Supply Chain, Optimization and Complex Networks.

Soundar Kumara is the Allen, E., and Allen,M., Pearce Professor of Industrial Engineering at Penn State. He also holds a joint appointment with the Department of Computer Science. He is an Adjunct Professor position with C.R. Rao Institute of Advanced Mathematics, Statistics and Computer Science, University of Hyderabad, India. He received his undergraduate degree in Mechanical Engineering from SVUCE, Tirupati, M. Tech in IE from IIT Madras, and Ph.D., from Purdue University. His research interests are in complex networks, sensor networks, crowdsourcing and process monitoring and diagnostics. He is a Fellow of Institute of Industrial Engineers and Fellow of the International Academy of Production Engineering (CIRP).



Dongwon Lee is an associate professor of College of Information Sciences and Technology (IST) at The Pennsylvania State University. He obtained a B.S. from Korea University in 1993, an M.S. from Columbia University in 1995, and a Ph.D. from UCLA in 2002, all in Computer Science. In-between M.S. and Ph.D., he has worked at AT&T Bell Labs from 1995 to 1997. His research interests include Database, Data Mining, Information Retrieval, Web Search and Analysis, and Web Services and has (co-)authored over 100 scholarly articles in competitive conferences or journals.