A Web Service Composition Framework Using Integer Programming with Non-Functional Objectives and Constraints

John Jung-Woon Yoo, Soundar Kumara, and Dongwon Lee
The Pennsylvania State University, University Park, PA 16802, U.S.A.
{jwyoo, skumara, dongwon}@psu.edu

Seog-Chan Oh
General Motors R&D Center, Warren, MI 48090, U.S.A.
seog-chan.oh@gm.com

Abstract

In this paper, we propose a Web service composition framework that uses Integer Linear Programming with non-functional objectives and constraints, in addition to the syntactic matching of Web services features. We envision that when Web services are fully deployed and commercialized in the near future, the criteria of Web service composition to achieve objectives will vary depending on users' needs or preferences from the number of Web services to non-functional objectives, such as costs, time, and/or reputation. Such non-functional attributes cannot be readily considered in planning-graph, constraint satisfaction, or propositional satisfiability techniques, which are predominantly logic-based. This paper shows how the proposed Integer Linear Programming framework can be utilized to compose Web services with non-functional attributes. This framework enables our composition software agent to identify the best composition result that satisfies both non-functional requirements as well as functional ones, namely, parameter matching. A preliminary implementation of the proposed idea and further research directions are also discussed.

1. Introduction

This paper proposes an *Integer Linear Programming* (ILP) based Web service composition framework that can incorporate not only functional requirements, such as parameter matching between Web services, but also non-functional ones, such as cost, lead time, and/or reputation. Our approach differs from most of the other approaches from the recent Web Services Challenge (WSC) that focus on functional aspects [5, 8]. As Web services become more popular and better utilized by many users and software agents, they will inevitably be commercialized. The commercialization of Web services requires a significant change in evaluating how to compose Web services. Traditionally, parameter name/type matching with the *minimal* number of Web services (i.e., a functional objective and constraints) has been used as one of the main criteria in the Web service composition problem. However, we believe that considering both functional and non-functional attributes together in solving the Web services composition problem would produce superior outputs.

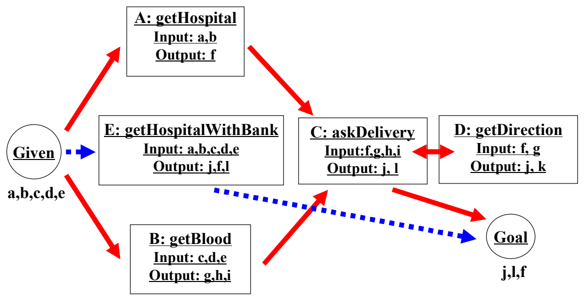

Figure 1. A use case with non-functional attributes

Figure 1 illustrates an example of a Web service composition case with non-functional attributes. The example is a complex Web service request, from a 911 center, for urgently finding a way to deliver a required amount of blood of a specific blood type from a blood bank which has the required blood availability, to an available, transfusion-capable hospital that is nearest to an accident site. In a virtual medical industry UDDI (Universal Description, Discovery and Integration), there are five relevant Web services, WS-A: getHospital ($10), WS-B: getBlood ($10), WS-C: askDelivery ($10), WS-D: getDirection ($10), and

WS-E: getHospitalwithBank ($50), where each Web service charges a nominal fee for the service request as indicated in the parentheses.

If the main objective of the composition problem is to minimize the number of Web services to be invoked, satisfying the parameter matching requirements, the best solution is to invoke only one Web service, WS-E at the cost of $50. The invocation of a single Web service, WS-E, is sufficient since it is provided by a hospital with a blood bank. On the other hand, if the main objective for the composition is to minimize the cost for Web service invocation, the best solution is to invoke four Web services, WS-A and WS-B concurrently and then invoke WS-D and WS-C sequentially, at a total cost of $40. Once a hospital and a blood bank having the required blood type are matched, we need to identify a transportation company and deliver blood from the blood bank.

One can consider several other objectives, such as minimal processing time or maximum reputation if the situation is very urgent or needs high credibility of the service providers, respectively. This example clearly explains our claim that depending on the user's objectives, which are non-functional in nature, the best solution varies. Therefore, in this paper, we propose a framework to address how to incorporate such "non-functional" attributes into a software agent for Web service composition.

The paper is organized as follows. Section 2 surveys related research work. In Section 3, the proposed Integer Linear Programming based framework is introduced. Implementation details are discussed in Section 4. Section 5 extends this research to Web service platform identification and con and consolidation 6 concludes this paper.

2. Background Research

An AI planning problem, P, can be represented as $P = (\Sigma, s_0, g)$ and $\Sigma = (S, A, \gamma)$, where S is the set of states, A is the set of actions, $\gamma : S \times A \rightarrow 2^S$ is a state transition function, s_0 is the initial state, and g is the goal state. Web service composition problems can be represented as an AI-Planning problem since the given parameters correspond to the initial state, the goal parameters to the goal state, the set of Web services to the set of actions, and the set of known parameters to the set of states, and finally the currently known parameters (input), an "invokable" Web service with the known parameters, and the newly known parameters (output) through the Web service invocation correspond to the state transition function. The AI-Planning problem has been addressed by a variety of techniques, such as planning-graph techniques, propositional satisfiability techniques, constraints satisfaction techniques, situational calculus, rule-based planning, theorem proving, among others [4, 9, 10]. In 1999, Vossen et al.[14] and Kautz and Walser [6] initiated the ILP-based approach to AI-Planning problems. In 2005, Van Den Briel and Kambhampati [12,13] reported that the ILP approach showed relatively good or even better performance than the most efficient SAT-based planners. Yoo and Kumara [15] formulated the Web service composition problem using the ILP-based AI Planning approach based on Van Den Briel and Kambhampati's Optiplan [12]. They showed that the number of variables and constraints does not increase exponentially as the number of Web services increases in the formulation.

There have been a few research works related to ILP-based Web service composition in order to incorporate quality of service (QoS) factors [3, 16], which are equivalent to non-functional attributes. However, in these approaches Web service composition is not performed on a parameter level but on a task level. In these methods, first multiple alternative Web services per task are selected and using ILP methods compositions that deliver the best QoS are arrived at.

3. Methodology

3.1. Integer linear programming

A linear program [2] is a mixed integer linear program (MILP) when some, but not all, of the variables need to be integer. If all the variables of the program are integer, it is called an *Integer Linear Programming (ILP)* [11].

The most significant benefit of the ILP approach is the capability of incorporating not only *functional* attributes (e.g., parameter matching between Web services), but also *non-functional* ones, (e.g., cost or time spent in invoking Web services). Note that most of the previous approaches tend to support only functional attributes. Furthermore, we can incorporate multiple objectives in the objective function, such as both cost and time, so that we can find alternative Web service composition solutions that are on the *efficient frontier* [1] of the multiple objectives. Such multiple objective ILP problems can be solved using various algorithms [1, 7].

3.2. Formulation

[1] An efficient frontier is a solution set where efficient solutions exist. An efficient solution can improve an objective only at the expense of at least one other objective.

Figure 2 outlines the ILP formulation of the Web service composition problem. Due to space constraint, further details are elaborated in [15]. Instead, here, we briefly explain the intuitive idea of the ILP formulation.

First, the definition of domain is as follows: W is the set of Web services in a domain specific UDDI system; P is the set of all parameters of Web services in W; $P_{In} \subseteq P$, is the set of parameters that are used as input in any Web service; $P_{Out} \subseteq P$, is the set of parameters that are used as output in any Web service; $P_{Initial} \subseteq P$, is the set of parameters that are initially given; $P_{Goal} \subseteq P$, is the set of parameters that are goal; $W_p^{input} \subseteq W, \forall p \in P$, is the set of Web services that have parameter p as input; $W_p^{output} \subseteq W, \forall p \in P$, is the set of Web services that have parameter p as output; Stage (s): $1 \leq s \leq S$, where S is the maximal number of stages for Web service composition.

Second, the brief explanation on the constraints is as follows: (1) is the objective function to represent any numerical, non-functional objective; both (2) and (3) are initial constraints, i.e., the given parameters; (4) is goal constraints, the parameters to find; (5) ~ (8) are Web service invocation constraints; (9) and (10) are non-concurrency constraints; (11) is sequence constraints; and (12) ~ (16) are the definition of variables, where all variables are binary.

4. Implementation

Figure 3 illustrates the architecture of our prototype Web service composer for the CEC/EEE 08 Web Services Challenge (WSC-08). The software composer agent consists of three steps, *bootstrapping*, *query processing*, and *execution*. The bootstrapping procedure first reads all the input WSDL (Web Service Description Language) files and OWL (Web Ontology Language) type hierarchy file before receiving queries. Next, the query processing procedure is initiated by client's query and generates the ILP formulation referring to the stored WSDLs and type hierarchy. Finally, the execution procedure processes the ILP formulation using an ILP solver and generates a WSBPEL (Web Service Business Process Execution Language) file that includes solutions.

5. Research Extensions

The proposed ILP-based Web service composition framework with non-functional attributes can be extended further. We discuss two such extensions regarding Web service packaging, pricing, and discounting for commercial Web services. Note that this cannot be done in previous approaches without non-functional attributes.

$$\textit{Minimize} \sum_{w \in W} \sum_{s \in S} c_w \cdot y_{w,s} \tag{1}$$

$\textit{Subject to}:$

$$x_{p,0}^{output} = 1, x_{p,0}^{input} = x_{p,0}^{known\text{-}unused} = 0 \;\; \forall p \in P_{Initial} \tag{2}$$

$$x_{p,0}^{input}, x_{p,0}^{output}, x_{p,0}^{known\text{-}unused} = 0, \forall p \notin P_{Initial} \tag{3}$$

$$x_{p,S}^{output} + x_{p,S}^{known\text{-}unused} + x_{p,S}^{input} \geq 1 \;\; \forall p \in P_{Goal} \tag{4}$$

$$\sum_{w \in W_p^{output} \setminus W_p^{input}} y_{w,s} \geq x_{p,s}^{output} \;\; \forall p \in P, s \in 1,\ldots,S \tag{5}$$

$$y_{w,s} \leq x_{p,s}^{output} \;\; \forall w \in W_p^{output} \setminus W_p^{input}, \forall p \in P, s \in 1,\ldots,S \tag{6}$$

$$\sum_{w \in W_p^{input}} y_{w,s} \geq x_{p,s}^{input} \;\; \forall p \in P, s \in 1,\ldots,S \tag{7}$$

$$y_{w,s} \leq x_{p,s}^{input} \;\; \forall w \in W_p^{input}, \forall p \in P, s \in 1,\ldots,S \tag{8}$$

$$x_{p,s}^{output} + x_{p,s}^{known\text{-}unused} \leq 1 \;\; \forall p \in P, s \in 1,\ldots,S \tag{9}$$

$$x_{p,s}^{input} + x_{p,s}^{known\text{-}unused} \leq 1 \;\; \forall p \in P, s \in 1,\ldots,S \tag{10}$$

$$x_{p,s}^{input} + x_{p,s}^{known\text{-}unused} \leq x_{p,s-1}^{input} + x_{p,s-1}^{output} + x_{p,s-1}^{known\text{-}unused} \;\; \forall p \in P, s \in 1,\ldots,S \tag{11}$$

$$x_{p,s}^{input}, x_{p,s}^{output}, x_{p,s}^{known\text{-}unused} \in \{0,1\} \;\; \forall p \in P, s \in 1,\ldots,S \tag{12}$$

$$y_{w,s} \in \{0,1\} \;\; \forall w \in W, s \in 1,\ldots,S \tag{13}$$

$$x_{p,s}^{known-unused} = \left\{ \begin{array}{l} 1 \text{ if parameter } p \text{ is known but not used in stage } s, \\ 0 \text{ otherwise.} \end{array} \right\} \tag{14}$$

$$x_{p,s}^{input} = \left\{ \begin{array}{l} 1 \text{ if Web services } w \text{ is invoked in stage } s \text{ such that } w \in W_p^{input}, \\ 0 \text{ otherwise.} \end{array} \right\} \tag{15}$$

$$x_{p,s}^{output} = \left\{ \begin{array}{l} 1 \text{ if Web services } w \text{ is invoked in stage } s \text{ such that } w \notin W_p^{input} \wedge w \in W_p^{output}, \\ 0 \text{ otherwise.} \end{array} \right\} \tag{16}$$

Figure 2. Integer Linear Programming formulation for the Web service composition problem

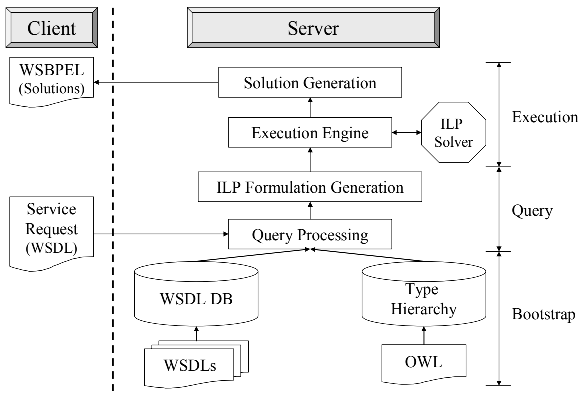

Figure 3. System architecture

Figure 4 illustrates how the ILP framework enables Web service providers to identify a Web service platform, so-called, "core services." Suppose a UDDI can provide Service Request A~Z for Web service users. The ILP approach with a particular objective function generates the best composition solution for each service request. For multiple service requests, if there exist common Web services in the composition solutions obtained by the ILP approach, the common Web services can be regarded as Web service platform (or core services) for the particular objective. Such a Web service platform can be utilized in Web service product packaging, pricing, and discounting.

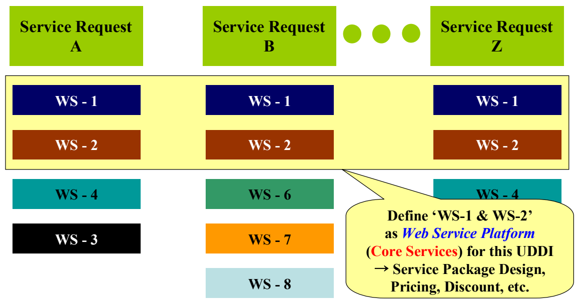
Service Request A
Service Request B
Service Request Z
WS - 1
WS - 2
WS - 4
WS - 3
WS - 6
WS - 7
WS - 8
Define 'WS-1 & WS-2' as *Web Service Platform* (Core Services) for this UDDI → Service Package Design, Pricing, Discount, etc.

Figure 4. Web service platform identification

Similarly, finding the best K Web service composition for a particular objective can be solved by using a *K best solutions* algorithm. If K best solutions contain common Web services, Web service providers need to consider whether they are to consolidate the common Web services or not. Through c them and setting slightly lower price than the sum of the individual Web services price of the best composition, the provider can outbid other competition for future composition requests.

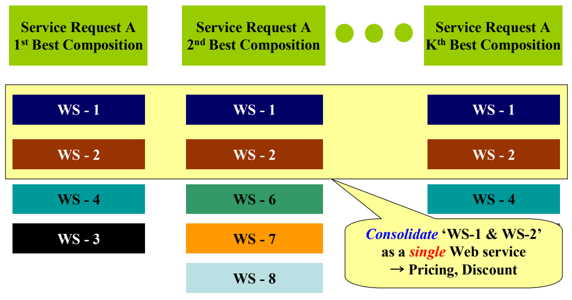
Service Request A 1st Best Composition
Service Request A 2nd Best Composition
Service Request A Kth Best Composition
WS - 1
WS - 2
WS - 4
WS - 3
WS - 6
WS - 7
WS - 8
Consolidate 'WS-1 & WS-2' as a *single* Web service → Pricing, Discount

Figure 5. Web service consolidation

6. Conclusions

An Integer Linear Programming (ILP) based Web service composition framework is proposed in this paper. This proposed approach can incorporate not only functional but also non-functional attributes into its model, so that both functional feasibility as well as the optimality in non-functional attributes can be achieved. In addition, this paper described the architecture of an ILP-based Web service composer and introduced potential research extensions for future work.

7. References

[1] A. I. Al-Rafai, "A Priori Interactive Methods for Multiple Objective Linear Programming Problems", *Ph.D. Dissertation*, The University of Oklahoma, 1993.
[2] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, "Linear Programming and Network Flows", 3rd edition, Wiley, 2004.
[3] R. Berbner, M. Spahn, N. Repp, O. Heckmann, R. Steinmetz, "Heuristics for QoS-aware Web Service Composition", *Proceedings of IEEE Int'l Conf. on Web Services*, 2006.
[4] M. Ghallab, D. Nau, and P. Traverso, "Automated Planning: theory and practice", Morgan Kaufmann Publisher, 2004.
[5] Z. Gu, B. Xu, J. Li, "Inheritance-Aware Document-Driven Service Composition", *Proc. of IEEE Int'l Conf. on E-Commerce Technology and on Enterprise Computing, E-Commerce, and E-Services*, pp. 513-516, 2007.
[6] H. Kautz and J. P. Walser, "State-Space Planning by Integer Optimization", *Proc. of American Association of Artificial Intelligence*, pp. 526-533, 1999.
[7] D. Klein and E. Hannan, "An Algorithm for the Multiple Objective Integer Linear Programming Problem", *European Journal of Operational Research*, **9**(4), pp. 378-385, 1982.
[8] S.-C. Oh, J.-W. Yoo, H. Kil, D. Lee, and S. Kumara, "Semantic Web-Service Discovery and Composition Using Flexible Parameter Matching", *Proc. of IEEE Int'l Conf. on E-Commerce Technology and on Enterprise Computing, E-Commerce, and E-Services*, pp. 533-536, 2007.
[9] S.-C. Oh, D. Lee, and S. Kumara, "A Comparative Illustration of AI Planning-based Web Services Composition", *ACM SIGecom Exchanges*, **5**(5), pp. 1-10, 2005.
[10] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods", *Lecture Note in Computer Science*, 3387, pp. 43-54, 2005.
[11] H. M. Salkin and K. Mathur, "Foundations of Integer Programming", North-Holland, 1989.
[12] M. Van den Briel and S. Kambhampati, "Optiplan: Unifying IP-based and Graph-based Planning", *J. of Artificial Intelligence Research*, **24**, pp. 919-931, 2005.
[13] M. Van den Briel, T. Vossen, and S. Kambhampati, "Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework", *Proc. of Int'l Conf. on* Automated Planning and Scheduling, pp. 310-319, 2005.
[14] T. Vossen, M. Ball, A. Lotem, and D. Nau, "On the Use of Integer Programming Models in AI Planning", *Proc. of Int'l Joint Conf. on Artificial Intelligence*, pp. 304-309, 1999.
[15] J.-W. Yoo and S. Kumara, "Integer Programming Formulation for Web Service Composition", *Technical Memorandum*, No. PSU-IE-LISQ-Yoo-2008-001-v1.0 (http://www2.ie.psu.edu/Kumara/Research/lisq/), 2008.
[16] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition", *IEEE Trans. on Software Engineering*, **30**(5), pp. 311-327, 2004.