QoS-Aware Dynamic Composition of Web Services Using Numerical Temporal Planning

Guobing Zou, Qiang Lu, Yixin Chen, *Senior Member*, *IEEE*, Ruoyun Huang, You Xu, and Yang Xiang

Abstract—Web service composition (WSC) is the task of combining a chain of connected single services together to create a more complex and value-added composite service. Quality of service (QoS) has been mostly applied to represent nonfunctional properties of web services and differentiate those with the same functionality. Many research has been done on QoS-aware service composition, as it significantly affects the quality of a composite service. However, existing methods are restricted to predefined workflows, which can incur a couple of limitations, including the lack of guarantee for the optimality on overall QoS and for the completeness of finding a composite service solution. In this paper, instead of predefining a workflow model for service composition, we propose a novel planning-based approach that can automatically convert a QoS-aware composition task to a planning problem with temporal and numerical features. Furthermore, we use state-of-the-art planners, including an existing one and a self-developed one, to handle complex temporal planning problems with logical reasoning and numerical optimization. Our approach can find a composite service graph with the optimal overall QoS value while satisfying multiple global QoS constraints. We implement a prototype system and conduct extensive experiments on large web service repositories. The experimental results show that our proposed approach largely outperforms existing ones in terms of solution quality and is efficient enough for practical deployment.

Index Terms—WSC, QoS, automated planning, temporal reasoning, numerical optimization

1 Introduction

Web services are modular, self-describing, self-contained, platform-independent software components that can be published by service providers over the Internet. Since web services became available, many organizations prefer to only keep their principal business, but outsource other application services over the Internet [1]. Web service composition (WSC) has been widely applied, allowing construction and sharing of independent and autonomous software [2]. As the number of web services proliferates, automated WSC is motivated by the need to improve the effectiveness and efficiency of integrating services [3].

WSC is the task of combining a set of single web services together to create a more complex, value-added and cross-organizational business process. WSC requires a computer program to automatically select, integrate, and invoke multiple web services to achieve a user-defined objective [3]. For those web services providing the same functionality, quality of service (QoS) has been mostly applied to represent their nonfunctional properties and differentiate them for service composition. QoS is a broad concept that encompasses a group of nonfunctional properties, such as execution price, execution duration, availability, execution success rate, and reputation [4]. Given a set of multiple global QoS constraints and user preferences, the challenge is how to efficiently construct a composite service such that its overall QoS is optimal, while all the QoS constraints are satisfied.

Existing QoS-aware WSC approaches fall short on finding solutions with globally optimal QoS, because it is a very difficult optimization problem with logical reasoning, discrete decisions, temporal constraints, and numerical optimization. In particular, when the number of web services becomes large, there is a huge search space. As a result, most existing QoS-aware WSC methods are restricted to predefined workflows [4], [5], [6], [7], [8], [9]. That is, it has a predefined workflow model to support service selection, as the one shown in Fig. 1. A predefined workflow consists of a set of tasks. For each task, it corresponds to a group of candidate web services so that each of them can perform the task. Fig. 2 illustrates the candidate services for a workflow model with p tasks. Since these conventional approaches are based on predefined workflows, their search space is reduced to a smaller one, which results in two limitations. One is that they cannot make sure its overall QoS is optimal, considering other workflows. Another is that these approaches do not guarantee finding a solution satisfying the global QoS constraints for a composition task, even if there exists one under a different workflow.

Automated planning has been a popular method for WSC [10], [11], [12], [13], [14], [15], [16], [17], [18], [19].

- *G. Zou is with the School of Computer Engineering and Science, Shanghai University, Room 919, Computer Building, 99 Shangda Road, Baoshan District, Shanghai 200444, China. E-mail: guobingzou@gmail.com.*
- *Q. Lu is with the Department of Computer Science and Technology, School of Computer Science and Technology, University of Science and Technology of China, No. 96, JinZhai Road, Hefei, Anhui 230026, China. E-mail: rczx@mail.ustc.edu.cn.*
- *Y. Chen is with the Department of Computer Science and Engineering, Washington University in St. Louis, Campus Box 1045, One Brookings Dr., St. Louis, MO 63130. E-mail: chen@cse.wustl.edu.*
- *R. Huang is with Google Inc., 747 6th Street South, Kirkland, WA 98033. E-mail: rh11@cse.wustl.edu.*
- *Y. Xu is with the Washington University in St. Louis, 20 W Kinzie St., Chicago, IL 60611. E-mail: yx2@cse.wustl.edu.*
- *Y. Xiang is with the Tongji University, 4800 Caoan Road, Shanghai 201804, China. E-mail: shxiangyang@tongji.edu.cn.*

Manuscript received 20 Dec. 2011; revised 23 June 2012; accepted 26 Sept. 2012; published online 12 Oct. 2012.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2011-12-0120.
Digital Object Identifier no. 10.1109/TSC.2012.27.

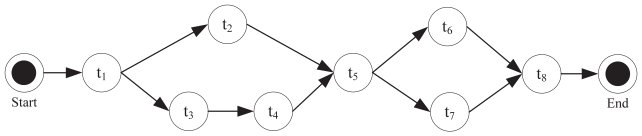


Fig. 1. A predefined workflow model for a WSC. It consists of eight business process tasks.

However, these approaches only try to find a composite service satisfying the functionality requirement, but do not consider QoS at all.

To address the above issues, we propose a novel planning-based approach to WSC with QoS optimization. One can specify multiple global QoS constraints and user preferences, and our method finds a composite service that optimizes the overall QoS, while satisfying those specified global QoS constraints.

Instead of predefining a workflow composition model for the selection of web services, our approach transforms a composition task with multiple global QoS constraints and preferences to a planning problem with temporal and numeric features. We leverage advances in temporal numerical planning to optimally and efficiently solve the resulting planning problems by our temporally numeric planner.

Our new approach has several advantages. It ensures that a composition solution can be found if one exists, while existing work may not when no solution within a predefined workflow satisfies global QoS constraints. Furthermore, our new approach can optimize the QoS, while conventional approaches only find the optimal QoS under a predefined workflow, which may not be globally optimal. The experimental results show that, our approach significantly extends the capability of prior work by ensuring global satisfiability and optimality without assuming a predefined workflow. The drawback of our approach is in its computational cost for a composition task. However, although our approach is slower than existing approaches, it can solve large instances in a few seconds and is still fast enough for practical deployment, thanks to efficient search engines in state-of-the-art automated planners.

2 Related Work

2.1 QoS-Aware WSC

Conventional QoS-aware WSC approaches can be classified into the following five categories, all of which assume a predefined workflow, which constrains their solution space. As a result, they are not globally complete or optimal.

- *Exhaustive search.* This approach [8] tries to enumerate all possible combinations by using candidate web services for each task. As a consequence, a composite service with the optimal QoS value for a predefined workflow model can be selected, if one exists and satisfies all global QoS constraints. However, the time complexity of this approach is high, i.e., $O(m^p)$, where m and p are, respectively, the maximum number of candidate services for a task and the number of tasks in a workflow.
- *Local optimization.* This is a locally optimal QoS service selection process [4] for WSC. A QoS vector is used to represent QoS of each service and a multiple criteria decision-making (MCDM) process [2], [8] is applied to calculate QoS value of a service, using the weights assigned by a user to each QoS criterion. For each task in a workflow, the service with the optimal QoS value is selected from its candidate service group, as shown in Fig. 2. Although this approach is locally optimal and efficient with a low time complexity of $O(m * p)$, it does not guarantee to satisfy global QoS constraints.
- *Integer programming.* Since high complexity makes exhaustive search impractical in real applications and local optimization does not take global QoS constraints into account, Zeng et al. [4] proposed a method based on integer programming (IP). Based on a predefined workflow model, it transforms a QoS-aware WSC problem to an IP problem. An IP solver is then used to find a solution.
- *Approximative algorithm.* To decrease the time complexity, Zhang et al. [7] model the QoS-aware WSC problem as a multidimensional multichoice knapsack problem. It still requires a predefined workflow. Although it is an NP-complete problem, a heuristic algorithm can be used. An approximative algorithm of constructing the convex hull of related points [7] was applied to generate a composite service with a suboptimal QoS value.
- *Situation calculus and Golog.* Considering rich user preferences as a key component, McIlraith's group proposed two ways of automated WSC with users' customized preferences [3], [20]. They first proposed a means of performing automated WSC by exploiting the agent programming language Golog to represent generic procedures and a first-order preference language to represent qualitative temporal user preferences [20]. Subsequently, a middle-ground execution engine [3] has been presented to generate high-quality compositions based on a hierarchical task network (HTN) WSC system, HTNWSC-P [21], which recursively decomposes a composition task into subtasks, and stops when it reaches primitive tasks that can be performed directly by planning operators.

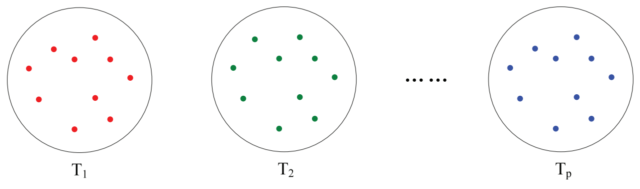


Fig. 2. The candidate services for a composite service workflow model containing p tasks. Each task $t_i (1 \le i \le p)$ corresponds to a set of candidate services T_i.

In addition to above hard QoS contracts and user preferences of WSC, Rosario et al. [22] argued that users would find it very natural to "soften" contracts so that probabilistic QoS and soft contracts should be taken for transaction-based web service orchestration.

2.2 Planning-Based WSC

For automated WSC, some approaches use AI search techniques to find a composition solution, such as heuristic forward and regression search [13], [16] and planning graph construction [14]. Rahmani et al. [23] proposed a novel backward search algorithm for automatically directing the composition process of web services. Although it considers nonfunctional user preferences for optimizing quality of WSC, multiple hard global QoS constraints are not taken into account when finding a composition solution. Based on an aggregated metric proposed by Blanco et al. [24] to estimate quality of service composition, two algorithms called DP-BF and PT-SAM are presented to select the best compositions, respectively. DP-BF combines a best first strategy with a dynamic programming technique and PT-SAM adapts a Petri net unfolding algorithm trying to find a desired marking from an initial state. By extending colored Petri net (CPN) formalism, Cardinale et al. [25] presented a CPN transactional web service selection (CPN-TWS) algorithm to address the issue of selecting and composing web services, considering functional and QoS requirements combined with transactional properties. The result is a CPN corresponding to a transactional composite service whose components locally optimize the QoS. For composition-oriented discovery of web services, an extensive matchmaking algorithm called service aggregation matchmaking (SAM) [26] features a more flexible matching by accounting for service composition.

There are also some composition approaches, where automated planners have been applied to select web services in real applications [27], [28] [29], [30] [31], [32], [33] for building operational business processes. The service composition planner OWLS-XPlan [27], [28] has been applied in an agent-based mobile eHealth system for emergency medical assistance (EMA) tasks. Another important application area for automated planner is the creation of new processes in business process management (BPM) at the SAP corporation. In this application [29], [30], a status and action management (SAM) model was developed to employ planning in a real-time BPM process modeling environment, SAP NetWeaver platform. O-Plan [31], [32], another automated planner, is used in a wide variety of WSC applications [32], [33], including air campaign planning, noncombatant evacuation operations, and biological pathway discovery.

3 Problem Formulation

In this section, we first focus on the understanding of QoS-aware service composition problem by a set of formal definitions, and then clearly demonstrate what a composition solution is to a QoS-aware WSC problem.

Definition 1 (Web Service). *A web service w consists of a finite set of operations, denoted as $w = \{op_1, op_2, \ldots\}$, where each $op \in w$, it is a three-tuple (I, O, Q), where $I = \{I^1, I^2, \ldots\}$ is a set of input interface parameters, $O = \{O^1, O^2, \ldots\}$ is a set of output interface parameters, $Q = (Q^1, Q^2, \ldots)$ is a set of QoS values for a group of QoS criteria $\{q_1, q_2, \ldots\}$. We use op.I, op.O, and $Q(op)$ to denote I, O, and Q in op, respectively.*

Each web service plays a role that can perform a set of operations. A web service repository is a set of disjoint services. We denote it as $W = \{w_1, w_2, \ldots\}$.

Definition 2 (Functionality Request). *A user's functionality request, r, is a two-tuple (r_{in}, r_{out}), where $r_{in} = \{r_{in}^1, r_{in}^2, \ldots\}$ is an interface parameter set provided as request inputs, and $r_{out} = \{r_{out}^1, r_{out}^2, \ldots\}$ is a goal specification provided as desired results.*

A functionality request (r_{in}, r_{out}) is specified by a user who provides a set of input parameters as request condition and goal facts as desired results.

QoS criteria can be divided into two categories: positive and negative. Positive QoS criteria denote better quality with higher values, while negative ones correspond to lower quality with higher values. Based on widely used QoS criteria [4], [34], we use a QoS vector $Q(op)$ to represent QoS values of each operation op.

$$Q(op) = (q_{price}(op), q_{time}(op), q_{succ}(op), q_{avail}(op), q_{rep}(op)),$$

where it models the values of a group of QoS criteria {execution price, execution time, probability of success, availability, reputation} in an operation op.

Definition 3 (Global QoS Constraints). *Given a group of QoS criteria $\{q_1, q_2, \ldots\}$, global QoS constraints, denoted as C, are a set of QoS values $(c_1, c_2, \ldots)$. Each $c_i \in C$ is a lower bound on a positive QoS criterion q_i or an upper bound on a negative QoS criterion q_i.*

Each global QoS constraint in C is used to restrict on its corresponding QoS criterion as global QoS value of a composite service.

Definition 4 (User Preferences). *Given a group of QoS criteria $\{q_1, q_2, \ldots\}$, user preferences, denoted as P, are a set of QoS weights $(p_1, p_2, \ldots)$. Each $p_i \in P$, it denotes a user's preference on the QoS criterion q_i. The preferences must satisfy $\sum_{i=1}^{|P|} p_i = 1$, and $0 \le p_i \le 1$.*

For a user preference in P, it denotes a bias on its corresponding QoS criterion by a user.

Given a set of services, a functionality request, a set of global QoS constraints and preferences, we define the problem of QoS-aware service composition as below.

Definition 5 (QoS-Aware WSC Problem). *A QoS-aware WSC problem, denoted as Q-WSC, is defined by $(W, C, P, r_{in}, r_{out})$, where*

1. *$W = \{w_1, w_2, \ldots\}$ is a web service repository,*
2. *$C = (c_1, c_2, \ldots)$ is a set of global QoS constraints,*
3. *$P = (p_1, p_2, \ldots)$ is a set of user preferences,*
4. *$r_{in} = \{r_{in}^1, r_{in}^2, \ldots\}$ is an input parameter set, and*
5. *$r_{out} = \{r_{out}^1, r_{out}^2, \ldots\}$ is a goal specification.*

The above Q-WSC problem defines a composition problem where a user can specify multiple global QoS constraints, a set of user preferences, and a functionality request based on a service repository.

Example 1. Consider a QoS-aware WSC problem $(W, C, P, r_{in}, r_{out})$. For the service repository W, it has three services $\{w_1, w_2, w_3\}$, where $w_1 = \{op_1, op_2, op_3\}$, $w_2 = \{op_4, op_5\}$ and $w_3 = \{op_6, op_7, op_8\}$. Table 1 shows all the

TABLE 1
Input and Output Parameters of Each Operation

Operation	Service	$op.I$	$op.O$
op_1	w_1	$\{par1, par2\}$	$\{par3, par4\}$
op_2	w_1	$\{par3\}$	$\{par5, par6\}$
op_3	w_1	$\{par4\}$	$\{par7, par8, par9\}$
op_4	w_2	$\{par7, par8, par9\}$	$\{par10\}$
op_5	w_2	$\{par5, par6, par10\}$	$\{par11, par12, par13\}$
op_6	w_3	$\{par11, par12\}$	$\{par14\}$
op_7	w_3	$\{par13\}$	$\{par15, par16\}$
op_8	w_3	$\{par14, par15, par16\}$	$\{par17, par18, par19\}$

Column "Service" denotes the web service that an operation belongs to. Columns "op.I" and "op.O" are input and output parameters set.

TABLE 2
The QoS Values of Each Operation Shown in Table 1

Operation	Exec. Price	Exec. Time	Probability of success	Availability	Reputation
op_1	26	15	0.85	0.93	4.6
op_2	34	22	0.90	0.88	3.3
op_3	18	36	0.84	0.89	5.0
op_4	49	19	0.96	0.95	3.7
op_5	37	20	0.91	0.97	4.8
op_6	15	11	0.98	0.86	3.1
op_7	35	28	0.93	0.92	3.5
op_8	19	23	0.82	0.75	4.1

operations in W, and their input and output parameters. Table 2 shows five QoS criteria as well as their QoS values for each operation in W. Assume that a user submits $C = \{240, 150, 0.40, 0.35, 3.8\}$ as global QoS constraints on the five QoS criteria, as shown in Table 2. Also, suppose that the user provides five QoS preferences $P = \{0.25, 0.3, 0.15, 0.2, 0.1\}$. Finally, an input parameters set $r_{in} = \{par1, par2\}$ and a goal specification $r_{out} = \{par17, par18, par19\}$ are also specified as functionality request.

Given a request $r = (r_{in}, r_{out})$, we need to find a sequence of operations $L = (op_1, op_2, \ldots, op_m)$ from W, such that it satisfies the functionality request. We define a *service state* as a set of input and/or output interface parameters $R = \{x^1, x^2, \ldots\}$.

Definition 6 (Operation Applicability). *Given an operation $op = (I, O, Q)$, it is applicable at a service state R if $op.I \subseteq R$. We denote this as $R \oplus op$.*

The above definition describes the applicability of an operation to a service state by checking whether input requirements of the operation are subsumed in the state.

When an applicable operation op is applied to R, the resulting service state $R' = R \oplus op$ is $R' = R \cup op.O$, in which the values of the parameters in $op.O$ are set by executing op. An *operation sequence* is an ordered list $L = (op_1, op_2, \ldots, op_m)$, where each element is an operation op. Applying a sequence L to a service state R results in $R' = R \oplus L = (\cdots((R \oplus op_1) \oplus op_2) \cdots \oplus op_m)$ if every step is applicable (otherwise $R \oplus L$ is undefined).

Definition 7 (Solution Satisfiability). *Given two service states $X = \{x^1, x^2, \ldots\}$, $Y = \{y^1, y^2, \ldots\}$ and a set of services W, if $X \oplus op_i \oplus \cdots \oplus op_m \supseteq Y$, $1 \le i, m \le |W|$, we say $(op_i, \ldots, op_m)$ is a solution sequence and denote this satisfiability as $(op_i \otimes \cdots \otimes op_m) \propto (X \to Y)$.*

Solution satisfiability identifies the applicability of an ordered sequence of operations from a service state.

Definition 8 (Composition Solution). *Given a Q-WSC problem $(W, C, P, r_{in}, r_{out})$, a composition solution to the problem is a solution sequence, $L^* = (op_1, \ldots, op_m)$, such that $(op_1 \otimes \cdots \otimes op_m) \propto (r_{in} \to r_{out})$ is satisfiable.*

Each composition solution corresponds to a composite service graph, which reflects the invocation order of operations including sequential and parallel ones between two operations, as defined below.

Our objective is not to find any composition solution, but the one that leads to the composite service graph with the optimal global QoS. As to global constraints and QoS optimization, we define composite service graph and then discuss them in the subsequent section.

4 QoS Model

Given each operation op and its QoS values $Q(op) = (q_{price}(op), q_{time}(op), q_{succ}(op), q_{avail}(op), q_{rep}(op))$, we now define composite service graph and its QoS model.

4.1 Composite Service QoS and Global Constraints

To describe global constraints and QoS optimization of a composite service, we first define operation dependence and composite service graph.

Definition 9 (Operation Dependence). *Given a composition solution, $L^* = (op_1, op_2, \ldots, op_m)$, an operation op_j depends on op_i (denoted as $op_i \prec op_j$) if and only if $i < j$ and there exists at least one output interface parameter $O^k \in op_i.O$, such that op_i is the last operation in $op_1, \ldots, op_{j-1}$ that satisfies $O^k \in op_j.I$.*

Operation dependence describes the connectivity of two operations during the invocation. Based on the operation dependence relationship, the composite service graph is defined as follows:

Definition 10 (Composite Service Graph). *Given a composition solution, $L^* = (op_1, op_2, \ldots, op_m)$, its composite service graph is a directed acyclic graph $G = (V, E)$, such that $V = L^*$ and there is an edge $(op_i, op_j) \in E$ if and only if $op_i \prec op_j$.*

A composite service graph G describes an execution process that provides the partial orders among the operations in L^*. For example, Fig. 3 illustrates an example of a composite service graph G, including 8 operations and 11 operation dependences.

Given a G and a group of global QoS constraints $C = (c^*(price), c^*(time), c^*(succ), c^*(avail), c^*(rep))$ on {*execution price, execution time, probability of success, availability, reputation*}, the QoS value of G on each criterion and its global constraint are as follows:

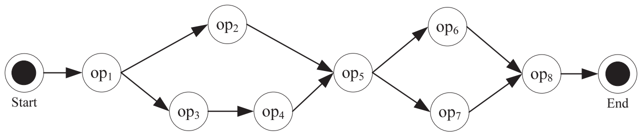


Fig. 3. A composite service graph consists of eight operations. Each operation has a set of QoS values on their corresponding QoS criteria, as shown in Table 2.

- *Execution price.* Given a composite service graph G, its execution price $Q_{price}(G)$ is the sum of execution prices of all the operations in G. Taking the given global QoS constraint $c^*(price)$ as its upper bound, we have an inequality for the global constraint:

$$\sum_{i=1}^{N} q_{price}(op_i) \leq c^*(price), \quad (1)$$

where N is the number of operations in G.

- *Execution time.* Given a composite service graph G, its execution time $Q_{time}(G)$ depends on the maximum span of execution time of operations in G. It is determined by a directed acyclic path starting from an initial state and ending at a goal state, called *critical operation path*, which consists of a subset of operations in G such that the sum of their execution time is maximized. That is, it is computed by the expression:

$$Q_{time}(G) = \sum_{op_i \in CP} q_{time}(op_i),$$

where $q_{time}(op_i)$ is the execution time of op_i, and op_i is an operation in the critical operation path CP. For example, Table 2 shows the execution time of each operation in Fig. 3. The critical operation path in Fig. 3 corresponds to $CP = \{op_1, op_3, op_4, op_5, op_7, op_8\}$. Based on the given $c^*(time)$ as its upper bound, the global constraint on execution time is:

$$\sum_{op_i \in CP} q_{time}(op_i) \leq c^*(time). \quad (2)$$

- *Probability of success.* Given a composite service graph G, its probability of success $Q_{succ}(G)$ is the product of probability of success of all the operations in G. Using $c^*(succ)$ as a lower bound, we have global constraint on probability of success:

$$\prod_{i=1}^{N} q_{succ}(op_i) \geq c^*(succ).$$

Since the above inequality is a nonlinear one that cannot be easily handled by AI planners, we transform it into a linear global constraint as follows:

$$\log_{\frac{1}{2}} \prod_{i=1}^{N} q_{succ}(op_i) \leq \log_{\frac{1}{2}} c^*(succ),$$

or equivalently,

$$\sum_{i=1}^{N} \log_{\frac{1}{2}} q_{succ}(op_i) \leq \log_{\frac{1}{2}} c^*(succ). \quad (3)$$

- *Availability.* Given a composite service graph G, its availability $Q_{avail}(G)$ is the product of availability of all the operations in G. Taking $c^*(avail)$ as its lower bound, the global constraint on availability is

$$\prod_{i=1}^{N} q_{avail}(op_i) \geq c^*(avail).$$

Similarly, we transform it into a linear constraint:

$$\sum_{i=1}^{N} \log_{\frac{1}{2}} q_{avail}(op_i) \leq \log_{\frac{1}{2}} c^*(avail). \quad (4)$$

- *Reputation.* Given a composite service graph G, its reputation $Q_{rep}(G)$ is the average reputation of all the operations in G. Using $c^*(rep)$ as its lower bound, the global constraint on reputation is

$$\frac{\sum_{i=1}^{N} q_{rep}(op_i)}{N} \geq c^*(rep). \quad (5)$$

Notice that we calculate the reputation of a composite service graph G by the mean of components. However, reputation is a very subjective term and can be evaluated by multiple rating standards. Another possibility is to use the minimal function min() among a group of operations. Our approach can still be applied with this choice.

Example 2. Consider the composite service graph G in Fig. 3. Table 2 shows QoS values of each operation. Reconsider the five global QoS constraints in Example 1, where $c^*(price) = 240, c^*(time) = 150, c^*(succ) = 0.40$, $c^*(avail) = 0.35$ and $c^*(rep) = 3.8$. The global constraints on five QoS criteria in G are as follows:

1. $\sum_{i=1}^{8} q_{price}(op_i) = 233 \leq c^*(price) = 240$;
2. $\sum_{op_i \in CP} q_{time}(op_i) = 141 \leq c^*(time) = 150$;
3. $\sum_{i=1}^{8} \log_{\frac{1}{2}} q_{succ}(op_i) = 1.253 \leq \log_{\frac{1}{2}} c^*(succ) = 1.322$;
4. $\sum_{i=1}^{8} \log_{\frac{1}{2}} q_{avail}(op_i) = 1.328 \leq \log_{\frac{1}{2}} c^*(avail) = 1.515$; and
5. $\frac{\sum_{i=1}^{8} q_{rep}(op_i)}{8} = 4.013 \geq c^*(rep) = 3.8$.

After checking above global constraints on five QoS criteria, G satisfies the given global QoS constraints.

4.2 QoS Normalization and Graph Optimization

When calculating the QoS score of a single operation, we adopted a weighted sum of values on QoS criteria. Since they have different ranges, we first normalize the QoS values to the range of [0, 1] before using them in the weighted sum. QoS normalization applied to each operation is designed to avoid frequent case, where several high scores on some QoS criteria in an operation reduce the discrimination of those low scores on some other QoS criteria within the same operation. Depending on the features of QoS criteria, normalization strategy is classified for positive QoS criteria and negative ones.

For positive QoS criteria, such as probability of success, availability, and reputation, we denote better quality by higher values. Since our formulation is a minimization problem, each positive QoS criterion, its QoS value of an operation is normalized to

$$q_i^j = \begin{cases} \dfrac{Q_i^{max} - q_i(op_j)}{Q_i^{max} - Q_i^{min}}, & \text{if } Q_i^{max} \neq Q_i^{min}, \\ 1, & \text{otherwise,} \end{cases}$$

where $q_i(op_j)$ and q_i^j represent the QoS values on the ith QoS criterion in op_j before and after QoS normalization, respectively. Q_i^{max} and Q_i^{min} are, respectively, the maximum and minimum QoS values on the ith QoS criterion among all the operations in a repository.

For negative QoS criteria, such as execution price and execution time, we denote lower quality by higher values. Each negative QoS criterion, we normalize its original QoS value in an operation by

$$q_i^j = \begin{cases} \dfrac{q_i(op_j) - Q_i^{min}}{Q_i^{max} - Q_i^{min}}, & \text{if } Q_i^{max} \neq Q_i^{min}, \\ 1, & \text{otherwise.} \end{cases}$$

We use a QoS vector $Q'(op)$ to represent the normalized values of an operation op as

$$Q'(op) = (q'_{price}(op), q'_{time}(op), q'_{succ}(op), q'_{avail}(op), q'_{rep}(op)).$$

Given a $Q'(op)$, the overall QoS value $qos(op)$ is calculated by a weighted sum of $Q'(op)$:

$$qos(op) = \sum_{i=1}^{n} (Q'(op)[i] * p_i),$$

where $Q'(op)[i]$ is the normalized value of op on the ith QoS criterion, n is the number of QoS criteria in op, and p_i is a user's preference on the ith QoS criterion. The weights satisfy $\sum_{i=1}^{n} p_i = 1$ and $0 \leq p_i \leq 1$.

Given a G, its QoS value $QoS(G)$ is calculated by the sum of the QoS values of all the operations in G.

$$QoS(G) = \sum_{i=1}^{N} qos(op_i).$$

Example 3. Reconsider the G illustrated in Fig. 3. We use Table 2 as the original QoS of each operation. After QoS normalization, Table 3 shows the normalized QoS values. The user preferences on these five QoS criteria are {0.25, 0.3, 0.15, 0.2, 0.1}, as shown in Example 1. The QoS value of each operation can be calculated. For example, the QoS of operation op_1 is calculated by $qos(op_1) = \sum_{i=1}^{5} Q'(op_1)[i] * p_i = 0.308$. As a result, the QoS value of G in Fig. 3 is calculated by $QoS(G) = \sum_{i=1}^{8} qos(op_i) = 3.442$.

To put all pieces together, the problem we solve is the following. Given a Q-WSC problem $(W, C, P, r_{in}, r_{out})$ (Definition 5), our goal is to find a composite service graph (Definition 10), G, such that it minimizes the overall QoS, while functionality request (r_{in}, r_{out}) and all global QoS constraints in C are satisfied. The objective function of the Q-WSC problem is

$$\arg\min_{G \in G^s} QoS(G), \tag{6}$$

where G^s represents all of the possible composite service graphs to the Q-WSC problem.

To solve a given Q-WSC problem, we transform it into a CSTE planning problem solved by our developed SCP solver in the following.

TABLE 3
The Normalized QoS Values of Each Operation

Operation	Exec. Price	Exec. Time	Probability of success	Availability	Reputation
op_1	0.324	0.160	0.813	0.182	0.211
op_2	0.559	0.440	0.500	0.409	0.895
op_3	0.088	1.000	0.875	0.364	0.000
op_4	1.000	0.320	0.125	0.091	0.684
op_5	0.647	0.360	0.438	0.000	0.105
op_6	0.000	0.000	0.000	0.500	1.000
op_7	0.588	0.680	0.313	0.227	0.789
op_8	0.118	0.480	1.000	1.000	0.474

The original QoS values of each operation are shown in Table 2.

5 QoS-Aware Service Composition Using Automated Planning

We develop a planning-based approach to optimally solve the Q-WSC problem. Fig. 4 illustrates an overview of our approach. It has a couple of major steps: 1) Translate a Q-WSC problem into a cost sensitive temporally expressive (CSTE) planning problem [35], [36], [37], which is a numeric planning problem with action duration and cost optimization features. 2) Solve the CSTE planning problem by our developed SAT-based cost planning (SCP) solver, which not only takes logical reasoning and temporal planning into account, but also optimizes overall QoS of a composite service graph.

For a restricted class of Q-WSC problems where there are no global QoS constraints on temporal restriction (e.g., execution time) and average-based constraint (e.g., reputation), we can also solve the planning problem using a numeric planner (Metric-FF [38]).

5.1 CSTE Planning Problem

We first formally define the CSTE planning problem [35], [36], [37] and CSTE action as follows:

Definition 11 (CSTE Planning Problem). *A cost sensitive temporally expressive planning problem is defined as (A, F, V, s_0, g), where A is a set of CSTE actions, F is a set of logical facts, V is a set of numeric variables, s_0 is the initial state, and g is a goal specification.*

A CSTE planning problem describes a planning problem with logical reasoning, temporal constraints planning, and numerical optimization.

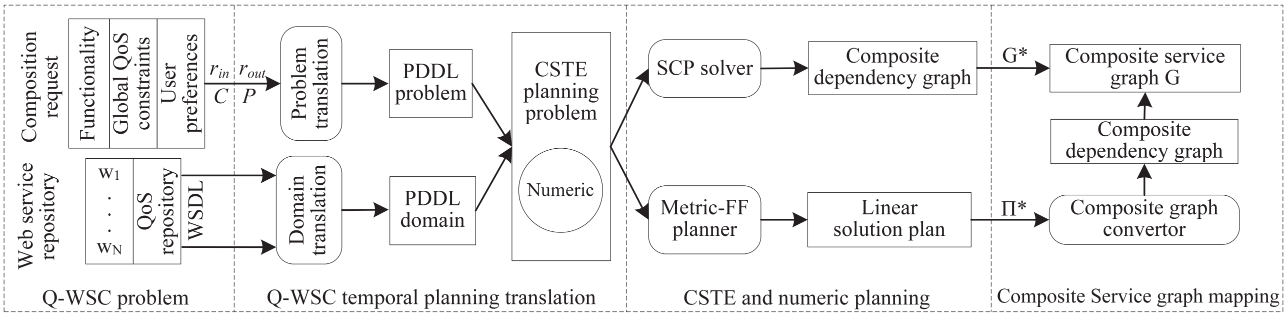


Fig. 4. Overview of our approach of QoS-aware WSC using numerical temporal planning.

Definition 12 (CSTE Action). *A CSTE action a is defined by a tuple (pre, eff, μ, ρ), where*

- *$pre(a)$ is the action precondition that consists of a set of numeric constraints based on numeric variables V and a set of logical facts, each of which is an atomic proposition $f \in F$.*
- *$eff(a)$ is the action effect that consists of a set of numeric effects based on V and a set of logical facts from F.*
- *$\mu(a)$ is the action duration.*
- *$\rho(a)$ is the action cost.*

A solution to a CSTE planning problem (A, F, V, s_0, g), is a composite dependence graph $G^* = (V, E)$, which transforms the initial state s_0 to a goal state g^*, such that all the logical facts in g are subsumed in g^*, i.e., $g \subseteq g^*$. Moreover, an optimal solution to a CSTE planning problem (A, F, V, s_0, g) minimizes the cost of all CSTE actions in G^*, denoted as $\min \sum_{a_i \in G^*} \rho(a_i)$.

5.2 CSTE Planning Formulation of Q-WSC

Given a Q-WSC problem $(W, C, P, r_{in}, r_{out})$, we translate it into a CSTE planning problem (A, F, V, s_0, g).

QoS numeric variables V. We divide V into four groups to represent objective QoS value function, QoS numeric constraints and numeric effects.

1. An objective QoS value variable V_o. It represents the overall QoS value by the costs of all the actions in a CSTE planning state.
2. A set of QoS value variables V_d. Each $V\_d[i]$, it represents the sum of QoS values for the ith QoS criterion in all the operations, which correspond to the CSTE actions in a CSTE planning state.
3. A set of global QoS constraint variables V_g. Each $V\_g[i]$, it is transformed from the ith global QoS constraint $c_i \in C$.
4. A set of user preference variables V_w. Each $V\_w[i]$, it represents preference $p_i \in P$ on the ith QoS criterion.

CSTE actions A. For each web service $w \in W$, we transform each of its operation $op = (I, O, Q)$ into a CSTE action $a = (pre, eff, \mu, \rho)$.

Precondition $pre(a)$. As to the numeric constraints in $pre(a)$, we specify a QoS expression as an arithmetic expression over V and the rational numbers with a set of arithmetic operators $\{+, -, *, /\}$. Then, by using this, we define a QoS numeric constraint as follows.

Definition 13 (QoS Numeric Constraint). *Given a set of V and a QoS expression, a QoS numeric constraint is a triple $(v, comp, exp)$, where $v \in V$ is a numeric variable, exp is a QoS expression, and $comp \in \{<, \leq, >, \geq\}$ is a comparative operator.*

A QoS numeric constraint restricts a numeric variable v to satisfy a QoS expression by a comparative operator. We use a set of QoS numeric constraints for numeric representation in $pre(a)$.

1. For each input parameter $I^i \in op.I$, we introduce a precondition fact $(yes\ I^i)$ in $pre(a)$. Here, we represent a precondition fact by a predicate $(yes\ ?p)$ which indicates the availability of an input or output parameter p in a CSTE planning state.
2. For each QoS value variable $V\_d[i]$, if it corresponds to a negative QoS criterion by addition (e.g., execution price), we develop a QoS numeric constraint $(V\_g[i], \geq, V\_d[i] + Q(op)[i])$ in $pre(a)$; otherwise, if it represents a positive QoS criterion by product (e.g., probability of success, availability), we introduce a QoS numeric constraint $(V\_g[i], \geq, V\_d[i] + \log_{\frac{1}{2}} Q(op)[i])$ in $pre(a)$.

Effect $eff(a)$. For the numeric representation in $eff(a)$, we define a QoS numeric effect over a set of V and a QoS expression.

Definition 14 (QoS Numeric Effect). *Given a set of V and a QoS expression, a QoS numeric effect is a triple (v, ass, exp), where $v \in V$ is a variable, $ass \in \{:=, +=\}$ is an assignment operator, and exp is a QoS expression.*

A QoS numeric effect updates QoS value of a numeric variable v by assigning or adding the value of a QoS expression. This way, we use a set of QoS numeric effects to represent the value changes of QoS numeric variables on each QoS criterion in $eff(a)$.

1. For each output parameter $O^i \in op.O$, we introduce an effect fact $(yes\ O^i)$ in $eff(a)$.
2. For each QoS value variable $V\_d[i]$, if it represents a QoS criterion by addition or average (e.g., execution price, reputation), we develop a QoS numeric effect $(V\_d[i], +=, Q(op)[i])$ in $eff(a)$; otherwise, it is a QoS criterion by product (e.g., probability of success, availability), we introduce $(V\_d[i], +=, \log_{\frac{1}{2}} Q(op)[i])$ in $eff(a)$.

Duration $\mu(a)$. We set the duration of action a as the QoS value of execution time in *op*. Assume that execution time is the ith QoS criterion in *op*, we have $\mu(a) = Q(op)[i]$.

Action cost $\rho(a)$.

1. We set action cost by a weighted sum of normalized QoS values of *op*. Thus, we have
$$\rho(a) = \sum_{i=1}^{n}(Q'(op)[i] * V\_w[i]),$$
where n is the number of QoS criteria in *op*. The weights satisfy $\sum_{i=1}^{n} V\_w[i] = 1$, where $0 \leq V\_w[i] \leq 1$.
2. For the objective QoS value variable V_o, we increase its QoS value by action cost $\rho(a)$ after the invocation and execution of action a. So we introduce a QoS numeric effect $(V\_o, +=, \rho(a))$ in $eff(a)$.

Facts F. The facts include all precondition and effect facts by input and output parameters from every operation $op \in w \in W$. That is, for each $I^i \in op.I$ or $O^i \in op.O$, we add a fact $(yes\ I^i)$ or $(yes\ O^i)$ to the facts F.

Initial state s_0. We set initial state s_0 by the initial facts and initial QoS values of the numeric variables V.

1. For each $r_{in}^i \in r_{in}$, we add an initial fact $(yes\ r_{in}^i)$.
2. We set the objective QoS value variable V_o as 0.
3. For each QoS value variable $V\_d[i]$, we set its initial QoS value as 0.
4. For each QoS constraint variable $V\_g[i]$, if it is used to restrict a QoS criterion by product (e.g., probability of success, availability), we set its initial value as $\log_{\frac{1}{2}} C[i]$; otherwise, its value is directly initialized as $C[i]$.
5. For each user preference variable $V\_w[i]$, we directly set its value as the corresponding preference in P. That is, we set each $V\_w[i]$ as user preference $P[i]$.

Goal specification g. We set g as a set of goal facts. For each goal parameter $r_{out}^i \in r_{out}$, we add a goal fact $(yes\ r_{out}^i)$ in g.

Example 4. Reconsider the QoS-aware WSC problem $(W, C, P, r_{in}, r_{out})$, as shown in Example 1. After Q-WSC temporal planning translation, it is transformed into a CSTE planning problem (A, F, V, s_0, g).

The QoS numeric variables V:

1. $V\_o = qos\_total$;
2. $V\_d = (price\_d, time\_d, succ\_d, avail\_d, rep\_d)$;
3. $V\_g = (price\_g, time\_g, succ\_g, avail\_g, rep\_g)$; and
4. $V\_w = (price\_w, time\_w, succ\_w, avail\_w, rep\_w)$.

After the translation, an operation in W is translated to a CSTE action. We take the operation op_1 as an example, its corresponding CSTE action a_1 is as follows:

For the precondition $pre(a_1)$:

1. Precondition facts: $\{(yes\ par1), (yes\ par2)\}$.
2. QoS numeric constraints:
$$\{(price\_g, \geq, price\_d + 26), (succ\_g, \geq, succ\_d + \log_{\frac{1}{2}} 0.85), (avail\_g, \geq, avail\_d + \log_{\frac{1}{2}} 0.93)\}.$$

For the effect $eff(a_1)$:

1. Effect facts: $\{(yes\ par3), (yes\ par4)\}$.
2. QoS numeric effects:
$$\{(price\_d, +=, 26), (succ\_d, +=, \log_{\frac{1}{2}} 0.85), (avail\_d, +=, \log_{\frac{1}{2}} 0.93), (rep\_d, +=, 4.6)\}.$$

For the action duration $\mu(a_1)$:

It is the QoS value on execution time of the operation op_1, so we have $\mu(a_1) = 15$.

For the action cost $\rho(a_1)$:

1. We set cost $\rho(a_1)$ as the QoS value of operation op_1. Thus, we have $\rho(a_1) = \sum_{i=1}^{5}(Q'(op)[i] * V\_w[i]) = 0.308$, where $V\_w[i]$ equals with the ith user preference $P[i]$.
2. Add $\rho(a_1)$ to $V\_o : \{(qos\_total, +=, 0.308)\}$.

After the translation, we get eight CSTE actions in $A = \{a_1, a_2, \ldots, a_8\}$, each of which is transformed from its corresponding operation in Example 1.

The facts F consists of all the atomic propositions by input and output parameters of each operation. Thus, we have $F = \{(yes\ par1), (yes\ par2), \ldots, (yes\ par19)\}$.

The initial state s_0 consists of five parts:

1. Initial facts: $\{(yes\ par1), (yes\ par2)\}$.
2. Initial objective QoS value: $qos\_total = 0$.
3. Initial value on each QoS criterion: $(price\_d = 0, time\_d = 0, succ\_d = 0, avail\_d = 0, rep\_d = 0)$.
4. Global QoS constraints: $(price\_g = 240, time\_g = 150, succ\_g = 1.322, avail\_g = 1.515, rep\_g = 3.8)$.
5. User preferences: $(price\_w = 0.25, time\_w = 0.3, succ\_w = 0.15, avail\_w = 0.2, rep\_w = 0.1)$.

Finally, goal specification consists of a set of goal facts by r_{out}, so $g = \{(yes\ par17), (yes\ par18), (yes\ par19)\}$.

5.3 Time Complexity Analysis of CSTE Translation

The CSTE planning formulation of Q-WSC consists of a CSTE domain and a CSTE problem. The former part models each operation $op \in w$ in W as a CSTE action $a \in A$. The latter translates a functionality request (r_{in}, r_{out}) and QoS requirements (C, P) to an initial state s_o and a goal state g.

Since CSTE domain translation is dominated by the conversion time from all of the operations in W to their CSTE actions in A, its time complexity is calculated by $O(\sum_{w \in W} \sum_{op \in w}((|op.I| + |V\_d| + |op.O| + |V\_d|) + (n + 3)))$, where n is the number of QoS criteria in an operation. We use N and M to denote the number of services in W and the maximum number of operations in a web service. Furthermore, $K = \max_{op \in w}\{|op.I| + |op.O|\}$ is an upper bound on the number of parameters in an operation in W. Meanwhile, $|V\_d|$ is equal to n, so the time complexity is $O(NM(K + 2n) + NM(n + 3)) = O(NM(K + 3n + 3))$. For a large scale W, we have $N \gg M, N \gg K, N \gg n$, and $K \gg n$. As a result, the time complexity of CSTE domain translation is $O(NMK)$.

The time complexity of CSTE problem translation is dominated by the number of numeric variables. The time complexity is bounded by $O(|V\_d| + |V\_g| + |V\_w| + |r_{in}| + |r_{out}|)$. Since the number of numeric variables is equal to the number of QoS criteria n, the time complexity is $O(3n + |r_{in}| + |r_{out}|)$. In a composition request, we have $n > |r_{in}|, n > |r_{out}|$. Thus, the time complexity of CSTE problem translation is $O(n)$, which is linear to the number of QoS criteria.

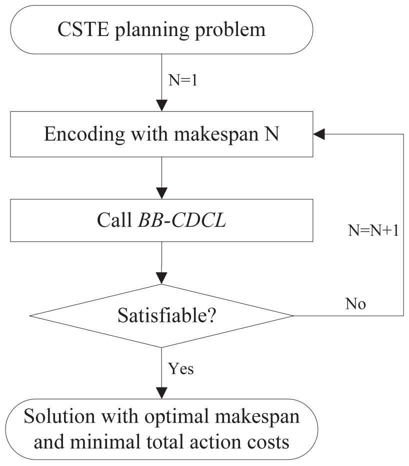


Fig. 5. The architecture of the SCP planner.

5.4 CSTE Planning by the SCP Solver

Given a CSTE planning instance transformed from a Q-WSC problem, we solve it by a CSTE planner, called the SCP solver [35], [36], [37]. SCP translates a CSTE problem into an optimization problem with satisfiability (SAT) constraints and multiple global constraints, denoted as QoS-MinCost SAT problem and defined as follows.

Definition 15 (QoS-MinCost SAT Problem). *A QoS-MinCost SAT problem is a tuple $\Phi^c = (U, L, \delta)$, where U is a set of Boolean variables, L is a set of clauses, and δ is a set of cost function $\{\delta_{qos}, \delta_{price}, \delta_{succ}, \delta_{avail}, \delta_{req}\}$, where each $\delta_i : U \to \mathbb{N}$. A solution to Φ^c is a variable assignment ψ that minimizes the objective function:*

$$QoS(\psi) = \sum_{x\in U} \delta_{qos}(x) v_\psi(x),$$

subject to:

$$v_\psi(p) = 1, \forall p \in L,$$

and multiple global constraints on QoS criteria:

$$\sum_{x\in U} \delta_{price}(x) v_\psi(x) \leq c^*(price);$$
$$\sum_{x\in U} \delta_{succ}(x) v_\psi(x) \leq \log_{\frac{1}{2}} c^*(succ);$$
$$\sum_{x\in U} \delta_{avail}(x) v_\psi(x) \leq \log_{\frac{1}{2}} c^*(avail);$$
$$\sum_{x\in U} \delta_{req}(x) v_\psi(x) \geq c^*(rep).$$

Fig. 5 shows the architecture of the SCP planner. Based on our previous work [36], we turn a CSTE instance into an optimization problem with SAT-based constraints, which is called a MinCost SAT instance as defined above. To solve the encoded MinCost SAT instance, we develop BB-CDCL, a Branch-and-Bound algorithm based on the conflict driven clause learning (CDCL) procedure. The planning algorithm follows the bounded SAT solving strategy, originally proposed in SATPlan. It starts from a lower bound of the makespan (N = 1), encodes the CSTE planning problem as a MinCost SAT instance, either proves it unsatisfiable and increase the makespan by 1, or finds an optimal solution to the MinCost SAT instance.

Given a Q-WSC problem $(W, C, P, r_{in}, r_{out})$, after our CSTE planning formulation and problem solving by the SCP solver, we can generate a composite dependence graph which describes the correct invocation and execution order of CSTE actions, and can be directly mapped to a composite service graph by a simple mapping from planning actions to operations.

5.5 Numeric Planning by Metric-Based Planner

For a class of restricted Q-WSC problems, we transform a Q-WSC problem $(W, C, P, r_{in}, r_{out})$ into a *numeric planning problem* and solve it by a numeric planner.

5.5.1 Finding a Linear Solution Plan

A *numeric planning problem* is a CSTE planning problem (A, F, V, s_0, g), except that, 1) there are no temporal and average-based global constraints in s_0, and 2) every action $a \in A$ is a *numeric action*.

Definition 16 (Numeric Action). *A numeric action is a CSTE action without action duration, which is denoted as a triple $a = (pre, eff, \rho)$. The precondition is $pre(a) = (p(pre), q(pre))$, where $p(pre)$ are precondition facts and $q(pre)$ are QoS numeric constraints. The effect is $eff(a) = (p(eff), q(eff))$, where $p(eff)$ are effect facts and $q(eff)$ are QoS numeric effects.*

Given a numeric planning problem (A, F, V, s_0, g), we define a *numeric planning state* $S = (S_p, S_q)$ in its solution space, where S_p is a fact state with a set of facts $\{(yes\ p_1), (yes\ p_2), \ldots\}$, and S_q is a numeric state with a set of QoS values $(V\_o, V\_d[1], V\_d[2], \ldots, V\_d[n])$.

Definition 17 (Action Applicability). *A numeric action $a = (pre, eff, \rho)$ is applicable to S (denoted as $S \succ a$), if it can satisfy: 1) $p(pre)$ is subsumed in S, $p(pre) \subseteq S_p$, and 2) QoS numeric constraints in $q(pre)$ must be satisfied by using the QoS value $V\_d[i]$ $(1 \leq i \leq n)$ in S_q, where n is the number of QoS criteria.*

When a numeric action a is applicable to a state S, the resulting planning state is $S' = S \succ a$, where $S'_p = S_p \cup p(eff)$, and $S'_q = S_q \triangleright q(eff)$, meaning that the QoS values $(V\_o, V\_d[1], V\_d[2], \ldots, V\_d[n])$ in S'_q are updated by applying the corresponding QoS values in S_q to QoS numeric effects in $q(eff)$.

An *action sequence*, $\Pi = (a_1, a_2, \ldots, a_m)$, is an ordered list of numeric actions. Applying a Π to a numeric planning state S results in a new planning state $S' = S \succ \Pi = (((S \succ a_1) \succ a_2) \succ \cdots \succ a_m)$ if every step is applicable (otherwise $S \succ \Pi$ is undefined).

Definition 18 (Plan Satisfiability). *Given a numeric planning state $S = (S_p, S_q)$, an action sequence $\Pi = (a_i, a_j, \ldots, a_k)$, and a set of propositional facts $X = \{(yes\ x_1), (yes\ x_2), \ldots\}$, if $S' = S \succ \Pi$ is defined and $X \subseteq S'_p$, we denote it as $(a_i \succ a_j \succ \cdots \succ a_k) \bowtie (S \to X)$, where $1 \leq i, j, k \leq |\Pi|$.*

Plan satisfiability describes the applicability of an ordered sequence of actions to a numeric planning state.

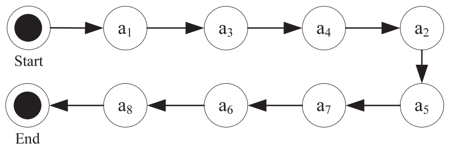


Fig. 6. The linear solution plan with suboptimal QoS value found by Metric-FF. Each numeric action a_i corresponds to an operation op_i from the Q-WSC problem in Example 1.

Definition 19 (Linear Solution Plan). *Given a numeric planning problem (A, F, V, s_0, g), a linear solution plan is an action sequence, $\Pi^* = (a_i, a_j, \ldots, a_k)$, such that $(a_i \succ a_j \succ \cdots \succ a_k) \bowtie (s_0 \rightarrow g)$ is satisfiable.*

A linear solution plan, $\Pi^* = (a_i, a_j, \ldots, a_k)$, is optimal, if the objective variable V_o in S_q^* has the minimal QoS value, where $S^* = (S_p^*, S_q^*) = s_o \succ \Pi^*$ is a resulting numeric planning state after applying every action $a \in \Pi^*$ to the initial state s_0.

Example 5. Reconsider the QoS-aware WSC problem $(W, C, P, r_{in}, r_{out})$, as shown in Example 1. Here, we only consider global QoS constraints $C = (240, 0.40, 0.35)$ on execution price, probability of success, and availability. After translation, we generate a numeric planning problem (A, F, V, s_0, g). We then use Metric-FF [38] to find a linear solution plan, as shown in Fig. 6. The solution plan satisfies: $(a_1 \succ a_3 \succ a_4 \succ \cdots \succ a_8) \bowtie (s_0 \rightarrow g)$, where s_0 and g are initial state and goal state.

We use Metric-FF [38] as an automated optimizer to address the problem, because it is a well-known numeric planner with the best performance in numeric track of the International Planning Competition. Note that, unlike SCP, it is a suboptimal planner that cannot guarantee optimality of the solution. However, it is very efficient and gives high-quality solutions in practice.

5.5.2 Constructing a Composite Dependence Graph

Given a linear solution plan, we convert it into a composite dependence graph, which includes both sequential and parallel order among numeric actions.

Definition 20 (Action Dependence). *Given a numeric planning problem (A, F, V, s_0, g), and a linear solution plan $\Pi^* = (a_1, \ldots, a_m)$, a numeric action a_j depends on a_i (denoted as $a_i \vdash a_j$), if and only if $i < j$ and there exists at least one precondition fact $f \in p(pre)$ in a_j, such that $f \notin s_0$ and a_i is the last action in $a_1, \ldots, a_{j-1}$ that satisfies $f \in p(eff)$ in a_i.*

By performing action dependence on a given $\Pi^* = (a_1, \ldots, a_m)$, we convert it into a composite dependence graph G^* as follows.

Definition 21 (Composite Dependence Graph). *Given a linear solution plan $\Pi^* = (a_1, \ldots, a_m)$, the composite dependence graph is a directed acyclic graph, $G^* = (V, E)$, such that $V = \Pi^*$ and there is an edge $(a_i, a_j) \in E$ if a_j depends on $a_i (a_i \vdash a_j)$.*

Example 6. Reconsider the linear solution plan Π^*, as shown in Fig. 6. After performing action dependence on the Π^*, Fig. 7 illustrates the generated composite dependence graph, where each numeric action a_i corresponds to an operation op_i in Example 1.

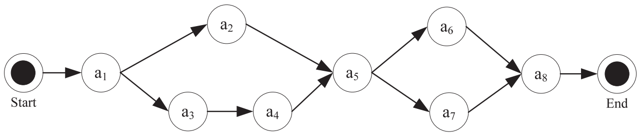


Fig. 7. The composite dependence graph for the linear solution plan shown in Fig. 6.

Once a linear solution plan Π^* with suboptimal QoS value is found by Metric-FF, we only need to convert this plan into a composite dependence graph G^* once, which still keeps suboptimal in terms of comprehensive QoS. In fact, the conversion from Π^* to G^* can be done fast enough because we only need to make a conversion within a very small finite number of numeric actions involved in Π^*. After the conversion, we directly map the G^* to a composite service graph G by replacing actions with their corresponding operations, as shown in Fig. 3.

6 Experimental Evaluation

6.1 Experimental Setup and Data Sets

Experiments are conducted on a DELL PC with Intel Dual Core 3.1 GHZ CPU and 4G RAM. We implemented a prototype system in Java. We also implemented the IP-based approach [4] by automatically generating AMPL model and data from web service repositories, which can be solved by an IP solver.

We have conducted extensive experiments on 30 web service repositories with 7,275 number of web services. Table 4 shows the number of operations in each of these 30 web service repositories. They are from six simulated predefined composition workflow models with the number of tasks 9, 16, 18, 7, 14, and 20, respectively. We mark these six workflow models as A, B, C, D, E, and F. By using our developed module *Q-services generator* to generate randomly specified number of operations, every workflow corresponds to five web service repositories. These repositories are generated, respectively, by the number of 5, 10, 15, 20, and 25 candidate operations for each task in the workflow, as well as randomly adding certain number of operations outside of the workflow.

For the QoS values of each operation in a web service repository, we use the module *Q-services generator* to randomly generate five QoS values on a group of QoS criteria, including execution price, execution time, probability of success, availability and reputation. Every QoS value on a QoS criterion is generated by *Q-services generator* with a specified value domain. Specifically, the range of five QoS values for each operation in the workflow models A, B, and C is {5-100, 1-50, 0.65-1, 0.65-1, 3.5-5} on the five QoS criteria. On the other hand, we have the range of five QoS values {5-50, 1-15, 0.65-1, 0.65-1, 3.5-5} for those operations in the workflow models D, E, and F.

TABLE 4
The 30 Web Service Repositories for Testing Metric-FF, SCP Solver and the IP-Based Approach

Tasks	5		10		15		20		25	
No.	WO	TO	WO	TO	WO	TO	WO	TO	WO	TO
A	45	55	90	110	135	165	180	220	225	275
B	80	90	160	180	240	270	320	360	400	450
C	90	100	180	200	270	300	360	400	450	500
D	35	45	70	90	105	135	140	180	175	225
E	70	80	140	160	210	240	280	320	350	400
F	100	115	200	230	300	345	400	460	500	575

The number of candidate operations for a task in a workflow are 5, 10, 15, 20, and 25. "WO" is the number of operations covered by tasks in a workflow. "TO" is the number of operations in a service repository.

6.2 QoS Value of a Composite Service Graph

6.2.1 The Results of Comprehensive QoS

The 30 web service repositories are classified into two groups to evaluate the QoS value of a composite service generated by our approach and the IP-based approach [4]. The first group of 15 web service repositories is from the workflow models A, B, and C. They are used to test our proposed approach using Metric-FF (called Metric-FF) and the IP-based approach (called IP-Linear), where each composition request in a service repository has three global QoS constraints without temporal and average-based features. The second group of 15 web service repositories from the workflows D, E, and F are applied to our approach using SCP solver (called SCP Solver) and the IP-based approach (called IP-SAT) with five global QoS constraints in each request.

Table 5 shows global QoS constraints of the 30 composition requests for these 30 web service repositories. On the one hand, we take the 15 composition requests on the first group of 15 web service repositories from workflow models A, B, and C. Each of these composition requests corresponds to a web service repository and has three global QoS constraints in the order of execution price, probability of success, and availability. On the other hand, we take another 15 composition requests on the second group of 15 web service repositories from the workflow models D, E, and F. Each of them has five global QoS constraints in the order of execution price, execution time, probability of success, availability, and reputation. We set QoS preferences $P = (0.2, 0.2, 0.2, 0.2, 0.2)$ on the five QoS criteria. After Q-WSC planning formulation, each Q-WSC problem is translated to a CSTE or numeric planning problem that is fed to Metric-FF or SCP solver to find a composite service graph. Also, we transform the Q-WSC problem to an IP problem with an AMPL model and AMPL data. Then, we use an IP solver CPLEX to find a composite service graph. Table 6 summarizes the comprehensive QoS of each composite service generated by Metric-FF, SCP solver, and the IP-based approach.

6.2.2 Comparison and Analysis

Based on the above experimental results on comprehensive QoS, we compare our approach against the IP-based approach and analyze the results.

1. Our proposed approach using planning does not depend on a predefined workflow model for QoS-aware service composition problem. However, the IP-based approach requires a predefined workflow and selects a sequence of services for the tasks defined in that workflow. Although these approaches under predefined workflows can efficiently find a composite service graph, they cannot ensure global satisfiability and optimality for a Q-WSC problem.
2. Our approach using Metric-FF planner can generate a linear solution plan with a globally sub-optimal QoS value. Experimental results show that the comprehensive QoS of a composite service found by our approach using Metric-FF outperforms that generated by the IP-based approach on all the test cases.
3. To deal with global QoS constraints with temporal and average-based features, we use our SCP solver to handle planning problems with temporal expressiveness and numeric reasoning. As shown in the experimental results, the QoS value of a composite dependence graph found by our approach using SCP solver is always better than that of a composite service found by the IP-based approach.

TABLE 5
Global QoS Constraints within a Composition Request for Each Repository

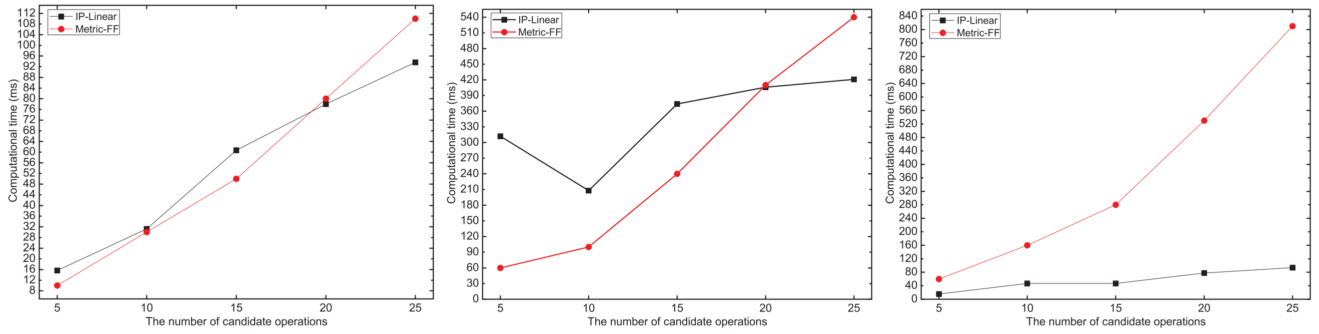
Tasks	Global QoS constraints				
No.	5	10	15	20	25
A	{209,0.228,0.274}	{176,0.275,0.255}	{183,0.289,0.240}	{192,0.295,0.281}	{163,0.225,0.299}
B	{416,0.032,0.038}	{372,0.047,0.042}	{359,0.042,0.078}	{406,0.191,0.074}	{348,0.179,0.102}
C	{409,0.052,0.063}	{362,0.049,0.136}	{417,0.055,0.087}	{348,0.122,0.079}	{394,0.063,0.087}
D	{146,58,0.196,0.291,4.15}	{127,39,0.209,0.253,4.15}	{90,64,0.257,0.206,3.9}	{138,43,0.291,0.235,4.4}	{105,82,0.248,0.289,4.35}
E	{378,90,0.099,0.090,4.2}	{350,102,0.079,0.148,4.3}	{279,116,0.128,0.115,4.05}	{248,109,0.104,0.074,4.25}	{215,80,0.147,0.141,4.26}
F	{459,178,0.067,0.074,4.25}	{364,161,0.073,0.052,4.10}	{462,170,0.078,0.084,3.95}	{397,150,0.075,0.052,4.0}	{482,162,0.072,0.081,4.17}

In the first group of 15 composition requests for the workflows A, B, and C, each composition request has three global QoS constraints in the order of execution price, probability of success, and availability. In the second group of 15 composition requests for the workflows D, E, and F, it has five global QoS constraints in each request in the order of execution price, execution time, probability of success, availability, and reputation.

TABLE 6
Experimental Results on the Comprehensive QoS Value (the Lower the Better) Found by Metric-FF, SCP Solver and the IP-Based Approach

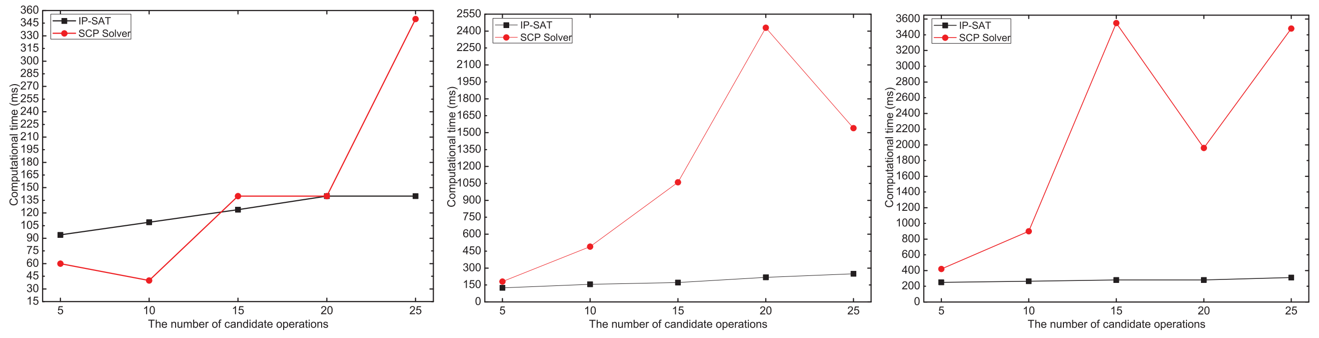
Tasks	5			10			15			20			25		
No.	Constraint	QoS value		Constraint	QoS value		Constraint	QoS value		Constraint	QoS value		Constraint	QoS value	
		FF	IP-Linear		FF	IP-Linear		FF	IP-Linear		FF	IP-Linear		FF	IP-Linear
A	C_A^5	4.326	5.155	C_A^{10}	3.930	5.263	C_A^{15}	4.017	5.061	C_A^{20}	3.763	5.022	C_A^{25}	3.946	4.718
B	C_B^5	7.892	10.260	C_B^{10}	7.377	8.743	C_B^{15}	7.286	8.515	C_B^{20}	6.864	8.703	C_B^{25}	6.812	8.423
C	C_C^5	9.036	10.863	C_C^{10}	8.423	10.007	C_C^{15}	8.869	10.103	C_C^{20}	7.810	9.499	C_C^{25}	8.122	9.747
No.	Constraint	SCP	IP-SAT	Constraint	SCP	IP-SAT	Constraint	SCP	IP-SAT	Constraint	SCP	IP-SAT	Constraint	SCP	IP-SAT
D	C_D^5	3.011	4.486	C_D^{10}	2.892	4.077	C_D^{15}	2.613	3.725	C_D^{20}	3.312	3.839	C_D^{25}	2.618	3.777
E	C_E^5	6.478	8.299	C_E^{10}	6.401	8.193	C_E^{15}	7.092	7.724	C_E^{20}	5.932	7.811	C_E^{25}	7.377	7.492
F	C_F^5	10.130	12.000	C_F^{10}	10.193	11.983	C_F^{15}	9.674	11.278	C_F^{20}	9.722	10.926	C_F^{25}	9.549	11.012

Column "Constraint" represents global QoS constraints specified in a composition request, as shown in Table 5. For a composition request on each web service repository, it corresponds to a set of global QoS constraints C_p^m, where p is the workflow model and m is the number of candidate operations for each task in the model. For example, C_A^5 represents the global QoS constraints {209,0.228,0.274} in Table 5. Column "FF" represents Metric-FF. Column "SCP" represents SCP solver.



(a) The number of workflow tasks = 9 (b) The number of workflow tasks = 16 (c) The number of workflow tasks = 18

Fig. 8. Experimental results on the computational time of Q-WSC of our approach using Metric-FF planner and the IP-based method IP-Linear. There are 15 composition requests tested on 15 web service repositories.



(a) The number of workflow tasks = 7 (b) The number of workflow tasks = 14 (c) The number of workflow tasks = 20

Fig. 9. Experimental results on the computational time of Q-WSC of our approach using SCP solver and the IP-based method IP-SAT. There are 15 composition requests tested on 15 web service repositories.

6.3 Time of Generating a Composite Service

We compare the computational time of our approach using Metric-FF and SCP to the IP-based approach. The experiments are tested on 30 composition requests, as shown in Table 5. Our approach using Metric-FF and SCP finds a linear solution plan and a composite dependence graph for a composition request, respectively. The IP-based approach using an IP solver finds a composite service graph for a composition request. Figs. 8 and 9 show computational time of each request.

From Figs. 8 and 9, we can see that, for the IP-based approach, its computing time of each composition request is in no more than 0.421 seconds. Our approach using Metric-FF can find a Q-WSC solution within 0.81 seconds for any composition request. For more complex problems with temporal numeric planning, our approach using SCP finds the optimal solution in no more than 3.55 seconds for all of the 30 composition requests.

Our approach is slower than the IP-based approach. The reason is that the IP-based approach only optimizes the QoS value under a given predefined workflow model (and hence suboptimal), while our approach using the SCP solver finds the optimal Q-WSC solution under all possible workflows, a service composition task with a much larger search space.

When the problem size is within the range specified by our experiments, the developed SCP planner can be integrated by corporations to help workflow creators build their operational business processes in real applications, such as airline ticket ordering, electronic commerce, and enterprise business workflow management.

From the above experimental results on computational time, we conclude that our approaches with Metric-FF and SCP, although slower than the IP-based approach, are still fast enough for practical use. Since our approach delivers solutions with the optimal QoS, it should be preferred by users who concern about QoS values such as execution price and time.

7 Conclusions

This paper presents an AI planning-based method for Q-WSC. The method first compiles a Q-WSC problem into a CSTE planning problem. Then, the method applies our recently developed SCP planner to handle the CSTE planning problem using temporal planning and numerical optimization and find a composite service graph with the optimal QoS. For a restricted class of Q-WSC problems, we transform them into a numeric planning problem, which can be solved by a metric-based planner Metric-FF. Our approach significantly extends the capability of prior work by ensuring global satisfiability and optimality without assuming a predefined workflow. The experimental results demonstrate that our proposed approach is fast enough for practical deployment, thanks to the highly efficient automated planners.

Acknowledgments

This work was partially supported by US National Science Foundation grant IIS-0713109, a Microsoft Research New Faculty Fellowship, a National Natural Science Foundation of China (61303096), and a Shanghai Natural Science Foundation (13ZR1454600).

References

[1] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," *Proc. First Int'l Conf. Semantic Web Services and Web Process Composition*, vol. 3387, pp. 43-54, 2005.

[2] J. Haddad, M. Manouvrier, and M. Rukoz, "TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition," *IEEE Trans. Services Computing*, vol. 3, no. 4, pp. 73-85, Jan.-Mar. 2010.

[3] S. Sohrabi and S. McIlraith, "Preference-Based Web Service Composition: A Middle Ground between Execution and Search," *Proc. Int'l Semantic Web Conf. (ISWC '10)*, 2010.

[4] L. Zeng et al., "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311-327, May 2004.

[5] D.A. Menascé, "Composing Web Services: A QoS View," *IEEE Internet Computing*, vol. 8, no. 6, pp. 88-90, Nov./Dec. 2004.

[6] M. Jaeger, G. Rojec-Goldmann, and G. Mühl, "QoS Aggregation for Web Service Composition Using Workflow Patterns," *Proc. Int'l Enterprise Distributed Object Computing Conf.*, 2004.

[7] W. Zhang et al., "QoS-Driven Service Selection Optimization Model and Algorithms for Composite Web Services," *Proc. Ann. Int'l Computer Software and Applications Conf. (COMPSAC)*, 2007.

[8] B. Wu, C. Chi, and S. Xu, "Service Selection Model Based on QoS Reference Vector," *Proc. IEEE Congress Services*, 2007.

[9] S. Hwang et al., "Dynamic Web Service Selection for Reliable Web Service Composition," *IEEE Trans. Services Computing*, vol. 1, no. 2, pp. 104-116, Jan. 2008.

[10] M. Falou et al., "A Distributed Planning Approach for Web Services Composition," *Proc. Int'l Conf. Web Services (ICWS '10)*, 2010.

[11] P. Bertoli et al., "Continuous Orchestration of Web Services via Planning," *Proc. Int'l Conf. Automated Planning and Scheduling (ICAPS '09)*, 2009.

[12] J. Hoffmann et al., "Message-Based Web Service Composition, Integrity Constraints, and Planning under Uncertainty: A New Connection," *J. Artificial Intelligence Research*, vol. 35, no. 1, pp. 49-117, 2009.

[13] S. Oh, D. Lee, and S. Kumara, "Effective Web Service Composition in Diverse and Large-Scale Service Networks," *IEEE Trans. Services Computing*, vol. 1, no. 1, pp. 15-32, Jan.-Mar. 2008.

[14] X. Zheng and Y. Yan, "An Efficient Syntactic Web Service Composition Algorithm Based on the Planning Graph Model," *Proc. Int'l Conf. Web Services (ICWS '08)*, 2008.

[15] J. Hoffmann, P. Bertoli, and M. Pistore, "Web Service Composition as Planning, Revisited: In between Background Theories and Initial State Uncertainty," *Proc. AAAI Conf. Artificial Intelligence (AAAI)*, 2007.

[16] S. Oh, D. Lee, and S. Kumara, "Web Service Planner (WSPR): An Effective and Scalable Web Service Composition Algorithm," *Int'l J. Web Services Research*, vol. 4, no. 1, pp. 1-23, 2007.

[17] M. Pistore, P. Traverso, and P. Bertoli, "Automated Composition of Web Services by Planning in Asynchronous Domains," *Proc. Int'l Conf. Automated Planning and Scheduling (ICAPS '08)*, 2005.

[18] M. Pistore et al., "Automated Composition of Web Services by Planning at the Knowledge Level," *Proc. Int'l Joint Conf. Artificial Intelligence*, 2005.

[19] E. Sirin et al., "HTN Planning for Web Service Composition Using SHOP2," *J. Web Semantics*, vol. 1, no. 4, pp. 377-396, 2004.

[20] S. Sohrabi, N. Prokoshyna, and S. McIlraith, "Web Service Composition via Generic Procedures and Customizing User Preferences," *Proc. Int'l Semantic Web Conf. (ISWC '06)*, 2006.

[21] S. Sohrabi and S. McIlraith, "Optimizing Web Service Composition While Enforcing Regulations," *Proc. Int'l Semantic Web Conf. (ISWC '09)*, 2009.

[22] S. Rosario et al., "Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations," *IEEE Trans. Services Computing*, vol. 1, no. 4, pp. 187-200, Oct. 2008.

[23] H. Rahmani, G. GhasemSani, and H. Abolhassani, "Automatic Web Service Composition Considering User Non-Functional Preferences," *Proc. Int'l Conf. Next Generation Web Services Practices (NWESP '08)*, 2008.

[24] E. Blanco, Y. Cardinale, and M. Vidal, "Aggregating Functional and Non-Functional Properties to Identify Service Compositions," *Engineering Reliable Service Oriented Architecture: Managing Complexity and Service Level Agreements*, pp. 145-174, IGI Global, 2011.

[25] Y. Cardinale et al., "CPN-TWS: A Coloured Petri-Net Approach for Transactional-Qos Driven Web Service Composition," *Int'l J. Web and Grid Services*, vol. 7, no. 1, pp. 91-115, 2011.

[26] A. Brogi, S. Corfini, and R. Popescu, "Semantics-Based Composition-Oriented Discovery of Web Services," *ACM Trans. Internet Technology*, vol. 8, no. 4, pp. 1-39, 2008.

[27] M. Klusch, A. Gerber, and M. Schmidt, "Semantic Web Service Composition Planning with OWLS-XPlan," *Proc. AAAI Fall Symp. Semantic Web and Agents*, 2005.

[28] M. Klusch and A. Gerber, "Fast Composition Planning of OWL-S Services and Application," *Proc. European Conf. Web Services (ECOWS '06)*, 2006.

[29] J. Hoffmann, I. Weber, and F. Kraft, "SAP Speaks PDDL: Exploiting a Software-Engineering Model for Planning in Business Process Management," *J. Artificial Intelligence Research,* vol. 44, pp. 587-632, 2012.

[30] J. Hoffmann, I. Weber, and F.M. Kraft, "Planning@SAP: An Application in Business Process Management," *Proc. Int'l Scheduling and Planning Applications Workshop (SPARK '09),* 2009.

[31] K. Currie and A. Tate, "O-Plan: The Open Planning Architecture," *Artificial Intelligence,* vol. 52, no. 1, pp. 49-86, 1991.

[32] A. Tate and J. Dalton, "O-Plan: A Common Lisp Planning Web Service," *Proc. Int'l Lisp Conf. (ILC '03),* 2003.

[33] S. Khan et al., "A Multi-Agent System-Driven AI Planning Approach to Biological Pathway Discovery," *Proc. Int'l Conf. Automated Planning and Scheduling (ICAPS '03),* 2003.

[34] Y. Liu, A. Ngu, and L. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection," *Proc. Int'l World Wide Web Conf. (WWW '04),* 2004.

[35] Q. Lu et al., "A SAT-Based Approach to Cost Sensitive Temporally Expressive Planning," to be published in *ACM Trans. Intelligent Systems and Technology,* 2012.

[36] R. Huang, Y. Chen, and W. Zhang, "An Optimal Temporally Expressive Planner: Initial Results and Application to P2P Network Optimization," *Proc. Int'l Conf. Automated Planning and Scheduling (ICAPS '09),* 2009.

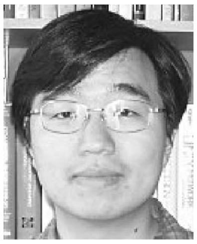
[37] Q. Lu et al., "Temporal Planning for Co-Design of Host Scheduling and Workflow Allocation in Mobile Environments," *Proc. Int'l Scheduling and Planning Applications Workshop (SPARK '11),* 2011.

[38] J. Hoffmann et al., "The Metric-FF Planning System: Translating Ignoring Delete Lists to Numeric State Variables," *J. Artificial Intelligence Research,* vol. 20, no. 1, pp. 291-341, 2003.

Guobing Zou received the joint PhD degree in computer science from Tongji University and Washington University in St. Louis, Missouri, in 2012. He has been an assistant professor in the School of Computer Engineering and Science at Shanghai University since 2012. His research interests include web service composition, automated planning, Semantic Web, and information retrieval. He has published 17 papers in international journals and conferences, including AAAI '12, *Soft Computing,* the *Journal of Tongji University,* FSKD '09, and CCV '10. He currently serves as a program committee member for CIT '12 and UUMA '12. He has also worked as a reviewer for the *Journal of Artificial Intelligence Research*, KDD '09, AAAI '10, and ICDM '10.

Qiang Lu received the BE and PhD degrees from the School of Computer Science and Technology, University of Science and Technology of China (USTC), in 2007 and 2012, respectively. He is currently a postdoctoral researcher in the School of Computer Science and Technology at USTC. His research interests include automated planning and scheduling, parallel and distributed computing, and cloud computing. He received the Joint PhD Training Scholarship from the China Scholarship Council in 2009. He has published more than 10 papers in journals and conference proceedings, including the *ACM Transactions on Intelligent Systems and Technology*, ICAPS '11, CloudCom '11, and IPC '11. He is a student member of the ACM and the CCF.

Yixin Chen received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2005. He is an associate professor of computer science at the Washington University in St. Louis, Missouri. His research interests include nonlinear optimization, constrained search, planning and scheduling, data mining, and data warehousing. His work on planning won first-place prizes in the International Planning Competitions (2004 and 2006). He won the Best Paper Award at AAAI (2010) and ICTAI (2005), and a best paper nomination at KDD (2009). He received an Early Career Principal Investigator Award from the US Department of Energy (2006) and a Microsoft Research New Faculty Fellowship (2007). He serves as an associate editor for the *IEEE Transactions on Knowledge and Data Engineering* and *ACM Transactions on Intelligent Systems and Technology*. He is a senior member of the IEEE.



Ruoyun Huang received the PhD degree in computer science from Washington University in St. Louis, Missouri, in August 2011. Before receiving his PhD degree, he worked for BearingPoint Consulting. He is currently a software engineer at Google, working with large-scale intelligent systems. His research interests include automated planning, large-scale intelligent systems, web service composition, and probabilistic inference. He has published more than 12 papers in top journals and conference proceedings, including the *Journal of Artificial Intelligence Research*, *Artificial Intelligence*, AAAI (2008, 2010, 2012), and ICAPS '09. He won the AAAI '10 Outstanding Paper Award.

You Xu received the BSc degree in mathematics from Nanjing University in 2006 and the MSc degree in computer science from the Washington University in St. Louis, Missouri, in 2009. He is currently working toward the PhD degree in the Department of Computer Science and Engineering at Washington University in St. Louis, Missouri. He is currently working for Google in Chicago, Illinois. His research interests include large-scale nonlinear optimization, constrained search, and partial-order reduction for planning, and automated planning in cloud computing. He has published 12 papers, including some appearing in conference proceedings such as RTAS '12, RTAS '11, MobiHoc '10, RTSS '10, and IJCAI '09.

Yang Xiang received the PhD degree in management science and engineering from the Harbin Institute of Technology in 1999. He is a professor in the Department of Computer Science and Technology at Tongji University. His research interests include data warehousing, data mining, intelligent decision support system, service computing and e-commerce. He has published more than 100 papers in international journals and conferences, including *Expert Systems with Applications, Science China Information Sciences,* and the *Chinese Journal of Electronics*. He has published four books on intelligent decision support system in complex problems.