Economic Model-Driven Cloud Service Composition

ZHEN YE and ATHMAN BOUGUETTAYA, Royal Melbourne Institute of Technology
XIAOFANG ZHOU, University of Queensland

This article considers cloud service composition from a decision analysis perspective. Traditional QoS-aware composition techniques usually consider the qualities available at the time of the composition because compositions are usually immediately consumed. This is fundamentally different in the cloud environment where the cloud service composition typically lasts for a relatively long period of time. The two most important drivers when composing cloud service are the long-term nature of the composition and the economic motivation for outsourcing tasks to the cloud. We propose an economic model, which we represent as a Bayesian network, to select and compose cloud services. We then leverage influence diagrams to model the cloud service composition. We further extend the traditional influence diagram problem to a hybrid one and adopt an extended Shenoy-Shafer architecture to solve such hybrid influence diagrams that include deterministic chance nodes. In addition, analytical and simulation results are presented to show the performance of the proposed composition approach.

Categories and Subject Descriptors: H.3.5 [**Online Information Services**]: Web-based services

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Cloud service, service composition, economic model

ACM Reference Format:
Zhen Ye, Athman Bouguettaya, and Xiaofang Zhou. 2014. Economic model-driven cloud service composition. ACM Trans. Internet Technol. 14, 2-3, Article 20 (October 2014), 19 pages.
DOI: http://dx.doi.org/10.1145/2651420

1. INTRODUCTION

Cloud computing is becoming increasingly popular as the next-generation platform for conducting business [Motahari-Nezhad et al. 2009]. Companies such as Amazon, Microsoft, Google, and IBM are already offering cloud computing solutions in the market.

Cloud computing has been intertwined with Service-Oriented Computing (SOC) since its inception [Yu et al. 2008]. Service composition is an active research topic in service-oriented computing [Medjahed et al. 2003]. Because of the volatile and distributed nature of the cloud and the ever-increasing number of cloud players in the market, providing an efficient approach for composing cloud services will become central to growing and sustaining the expected large demand. Current techniques for composing services are not well suited for the cloud environment [Dustdar and Schreiner 2005].

Traditional QoS-aware composition techniques usually consider the qualities at the time of the composition [Zeng et al. 2004]. This is fundamentally different in cloud environments where the cloud service composition typically lasts for a long period.

This research was made possible by NPRP 7-481-1-088 grant from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.
Authors' addresses: Z. Ye (corresponding author) and A. Bouguettaya, Royal Melbourne Institute of Technology, 124 Little La Trobe Street, Melbourne VIC 3000, Australia; email: jayyezh@gmail.com; X. Zhou, University of Queensland, Brisbane, St. Lucia QLD 4072, Australia.

DOI: http://dx.doi.org/10.1145/2651420

Compared to traditional service composition, cloud service composition is long-term based and economically driven [Armbrust et al. 2009]. Specifically, we identify the following problems in existing solutions.

—*Lack of Long-Term Quality-of-Service-based Service Selection*. End-users are usually large companies and organizations that aim to construct long-term business relationships with cloud service providers [Armbrust et al. 2009]. This aspect is largely lacking in existing service composition solutions such as those of Medjahed et al. [2003] and Zeng et al. [2004]. Although some existing work (e.g., Liu et al. [2010]) have conducted research in the area of change management in long-term composed services, few have considered the problem of making long-term QoS-aware selection.
—*Lack of an Economic Model Leveraging Users and Multiple Cloud Providers*. End-users and service providers participate in service composition according to their economic models [Stonebraker et al. 1994; Dash et al. 2009]. Existing models addressing economic aspects do not consider service composition but mostly focus on resource provision to specific applications [Kantere et al. 2011] (e.g., cloud cache, scientific applications). In addition, there is a tremendous amount of cloud service providers in the cloud environment thus end-users have the flexibility to select different cloud service providers. Existing work such as Dash et al. [2009] and Kantere et al. [2011] mostly focus on a single service provider scenario.
—*Lack of Composition Approach across Multiple Levels*. Cloud service composition happens at multiple levels [Ye et al. 2011] (e.g., SaaS and IaaS levels). Existing composition approaches only consider service composition at the same level [Medjahed et al. 2003; Zeng et al. 2004]. It is important to note that the nonfunctional properties of composite cloud services are determined not only by the selected SaaS providers but also by the PaaS or IaaS providers. Therefore, a cross-level composition approach is needed.

This article presents a novel QoS-aware cloud service composition approach. This approach enables long-term quality-driven composition of cloud services by addressing the preceding key issues. Our main contributions include the following.

—*Cloud Economic Model*. Economic models [Haque et al. 2011] have been used to provision large-scale heterogeneous resources in the recent ten years. Economic models describe the long-term preferences and behaviors for each actor in the cloud environment. In this article, cloud service composition is considered from the end-user's perspective. A Bayesian network-based technique is used for end-users to represent their preferences and strategies in their economic models.
—*Decision Analysis Approach*. Our research considers cloud service composition problems from a decision analysis perspective. For an end-user, the composition framework needs to make decisions on which cloud service providers should be contracted for all the abstract services in the composite services. We adopt the *influence diagram* [Shachter 1988] to represent and solve this decision problem.
—*Experiments*. We conduct several experiments to illustrate the efficiency of the proposed composition approach. An example scenario is considered where the composition approach composes cloud services to process the real-time traffic in big cities.

The remainder of the article is structured as follows: a motivating scenario is detailed in Section 2. Section 3 provides an overview of the cloud computing framework and introduces the preliminaries of cloud service composition. Section 4 analyses the research challenges and elaborates on the proposed composition approach. Section 5 presents evaluations and shows the results. Related work is presented in Section 6 and, finally, Section 7 concludes and highlights some future work.

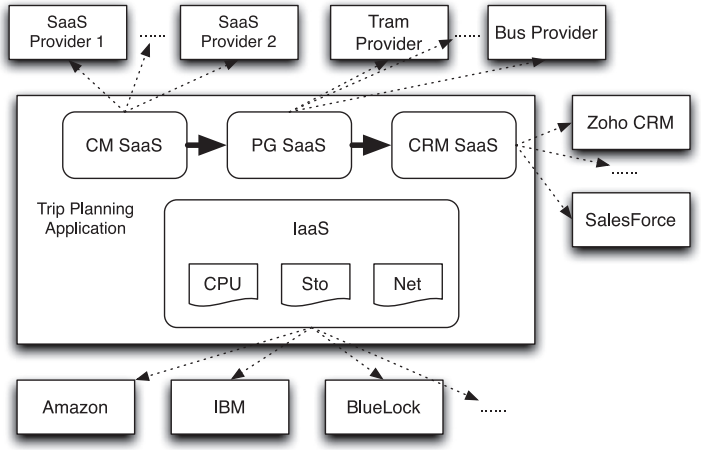


Fig. 1. Trip planning application for a company.

2. MOTIVATING SCENARIO

Let us consider a travel planning company, EasyPlanner, that aims to provide services to commuters in Melbourne, Australia. Customers of EasyPlanner can access its services through the company's Web site or through the EasyPlanner application in their smartphones. Suppose Michael is planning to travel from A to B on a weekday morning. He wants to get information about the public transport (i.e., buses, trams, and trains) in the city to plan his journey. After Michael issues his request though the mobile app, EasyPlanner will construct a composite service to fulfill this request. First, EasyPlanner will seek to update the public transport information. It then generates the best commute plan between start point A and end point B. Lastly, it will update Michael's usage information for further recommendation. Therefore, in this motivating scenario, the composite cloud service has three component abstract SaaS, namely a Content Management (CM) SaaS, a Plan Generater (PG) SaaS, and a Customer Relationship Management (CRM) SaaS. (Figure 1).

It is assumed that several SaaS providers would supply SaaSs for each abstract SaaS. Selecting which specific SaaS provider to implement each abstract SaaS depends on the company's QoS requirements. The content management application is to ensure that the traffic information is up to date. Usually, there are multiple options to travel between two locations, for instance, by tram, bus, or train. Traffic changes in any of these options will trigger the application to update the information in the cloud. To help with the collection of information, EasyPlanner relies on the deployment of sensors in trams, buses, and trains. The content management application would constantly check the updated data from these sensors and update traffic information in the cloud. One important QoS attribute is the degree of reliability of the CM SaaS. Here we consider that a CM SaaS with a higher scoring reputation is better with reliability. The plan generator application is to identify the best travel plan. Besides reputation, another vital QoS attribute for this abstract SaaS is the cost to compute the best travel plan. The customer relationship management application is responsible for managing customer information in the cloud. It is possible that there are tens of thousands of customers per day who want to enquire about their traveling plans and that each customer may use the trip planning application multiple times. One of the most important QoS attributes is the service rate, namely how many customers the CRM SaaS can support. Hence,

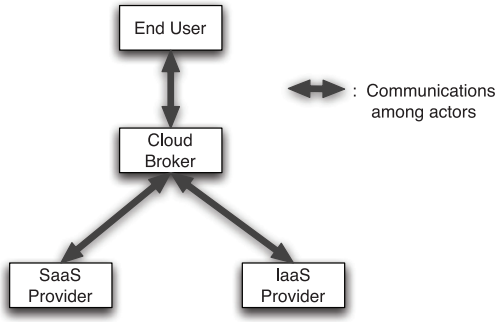


Fig. 2. Four actors in cloud computing.

when EasyPlanner does composition in the cloud, there are at least three QoS attributes to consider: reputation, cost, and service rate.

In addition to the selections of SaaS providers, the composite cloud service also needs CPU, network, and storage resources to support its executions. CPU services (denoted as CPU) are used to do computations on data. Storage services (denoted as Sto) are used to keep intermediate data. Network services (denoted as Net) are needed to transfer data between end-users and the application and also between the three components in the composite application.

Note that EasyPlanner will most likely have fluctuating QoS requirements on the composite trip planning application when we consider a long period of time $[t_i, t_k]$. Suppose the company presents these preferences through a score function as shown in Zeng et al. [2004]. We assume composite services with higher score are more preferable. EasyPlanner changes the preferences by changing the weights in the score function for different QoS attributes. For example, within $[t_i - t_j]$, the company could prefer a composite service that has a higher scoring reputation. For example, the weights for the three QoS attributes could be: reputation (0.5), service rate (0.3), and cost (0.2). While in the period from $[t_j - t_k]$, EasyPlanner may decide that saving cost is more important, therefore the weights for each QoS attribute would be: reputation (0.1), service rate (0.2), and cost (0.7). To obtain an optimal composition, a long-term economic-driven model is therefore needed to model the preferences of the end-users and cloud service providers. Based on the user's economic model, decisions are made to select concrete SaaS and IaaS providers for the company.

3. PRELIMINARIES

This section presents the preliminaries of the cloud service composition problem. First, the cloud environment is presented, followed by the composition procedure. The basic concepts and definitions used in this research are then explained in this section.

3.1. Cloud Environment

In this research, we consider four actors in the cloud environment (Figure 2): *end-users*, *cloud broker* [Liu et al. 2011], *SaaS (Software-as-a-Service) Providers*, and *IaaS (Infrastructure-as-a-Service) Providers*. For the purpose of this research, the Platform-as-a-Service (PaaS) provider is omitted as we assume we can package the applications and deploy them directly to virtual machines. End-users are usually large companies and organizations that outsource their tasks to the clouds. A cloud broker is an entity that manages the use, performance, and delivery of cloud services and negotiates relationships between cloud providers and end-users [Liu et al. 2011]. IaaS providers

supply infrastructure resources [Armbrust et al. 2009], namely *CPU services, storage services, and network services*, to SaaS providers and end-users.

Cloud service composition is the process of doing compositions on both SaaS level and IaaS level to form composite services. Similar to traditional service composition [Milanovic and Malek 2004], cloud service composition is conducted in two steps. First, a composition schema is constructed on the SaaS level. Second, SaaS providers are selected to supply SaaS to the abstract services in the composition schema by making Service-Level Agreements (SLAs) between end-users and SaaS providers, while an IaaS provider is selected to support the composite service on the IaaS level. This research focuses on the selection of cloud service providers based solely on nonfunctional (QoS) attributes. Therefore, we assume existing composition techniques for matching functional properties will be used (e.g., Medjahed et al. [2003]) to generate composition schema.

3.2. QoS Model

To differentiate composition plans during selection, their nonfunctional properties need to be considered. For this purpose, we adopt a QoS model applicable to all the SaaS and IaaS. Without loss of generality, we only consider the QoS attributes listed as follows. Although the adopted QoS models have a limited number of attributes, they are extensible and new QoS attributes can be added.

3.2.1. QoS Model for Elementary Services. Five QoS attributes are considered for component services: accuracy, reputation, throughput, response time, and cost.

—*Reputation*. Given an SaaS provider SP, the reputation $q_{rep}(SP)$ is a measure of its trustworthiness. It mainly depends on the end-user's experiences of using SaaS from the provider SP.

—*Throughput (i.e., Service Rate)*. Given an SaaS provider SP, the throughput of its SaaS $q_{tp}(SP)$ is the number of requests the SaaS provider is able to process per second. Given an IaaS provider IP, the throughput of its IaaS $\overrightarrow{q_{tp}(IP)} = [q_{tp}^{CPU}(IP), q_{tp}^{Net}(IP), q_{tp}^{Sto}(IP)]$ is a three-attribute vector, where $q_{tp}^{CPU}(IP)$ ($q_{tp}^{Net}(IP)$, $q_{tp}^{Sto}(IP)$) represents the number of CPU (network, storage) requests the IaaS provider is able to process per second.

—*Cost*. Given an SaaS provider, the execution cost $q_{cost}(SP)$ is the fee that a customer needs to pay for a single request if the SaaS provider agrees to supply SaaS at throughput $q_{tp}(SP)$. The total execution cost is computed using the expression $cost = q_{tp}(SP) \cdot q_{cost}(SP)$. Given an IaaS provider IP, the cost for using unit IaaS for one second is denoted as a three-attribute vector $\overrightarrow{q_{cost}(IP)} = [q_{cost}^{CPU}(IP), q_{cost}^{Net}(IP), q_{cost}^{Sto}(IP)]$, where $q_{cost}^{CPU}(IP)$, $q_{cost}^{Net}(IP)$ and $q_{cost}^{Sto}(IP)$ are the price for using unit CPU IaaS, unit network IaaS, and unit storage IaaS for one second correspondingly. For a CPU request, the cost of computing output data from task t_i is represented as $q_{cost}^{CPU}(t_i) = q_{cost}^{CPU}(IP) \cdot q_{cap}^{CPU}(IP) \cdot q_{rt}^{CPU}(t_i)$. The cost to transfer input data for task t_i is calculated using $q_{cost}^{IN}(t_i) = q_{cost}^{Net}(IP) \cdot q_{cap}^{Net}(IP) \cdot q_{rt}^{IN}(t_i)$, whereas that to transfer output data for task t_i can be calculated as $q_{cost}^{OUT}(t_i) = q_{cost}^{Net}(IP) \cdot q_{cap}^{Net}(IP) \cdot q_{rt}^{OUT}(t_i)$. The cost to store intermediate data Sto_i is computed as $q_{cost}^{STO}(Sto_i) = q_{cost}^{Sto}(IP) \cdot Sto_i \cdot time$, where $time$ denotes the seconds for which the intermediate data should be stored.

3.2.2. QoS Model for Composite Services. The quality criteria defined earlier are in the context of elementary cloud services. Aggregation functions are used to compute the QoS of the composite service. Table I presents these aggregation functions.

Table I. Aggregation Functions for Each QoS Attribute

QoS Attribute	Aggregation Functions
Reputation (q_{rep})	$\Pi_{i=1}^{N} q_{rep}(SP_i)$
Throughput (q_{tp})	$min(q_{tp}(SP_1), q_{tp}(SP_2), \ldots, q_{tp}(SP_N), q_{tp}^{CPU}(IP), q_{tp}^{Net}(IP), q_{tp}^{Sto}(IP))$
Cost (q_{cost})	$\sum_{i=0}^{N}(q_{cost}(SP_i) + q_{cost}^{CPU}(\tau_i) + q_{cost}^{IN}(\tau_i) + q_{cost}^{OUT}(\tau_i) + q_{cost}^{Sto}(Sto_i)) \cdot q_{tp}(aCS)$

—*Reputation*. For an abstract composite service aCS, the reputation $q_{rep}aCS$ is the product of the reputation of all the selected SaaS.
—*Throughput*. The throughput of a composite service denotes the number of requests it serves per second. For an abstract composite service aCS, the throughput $q_{tp}(aCS)$ is the minimal throughput of the selected SaaS providers $q_{tp}(SP)$ and the IaaS provider $q_{tp}(IP)$.
—*Cost*. The cost of an abstract service is the sum of execution costs of all the selected SaaS and IaaS.

4. LONG-TERM CLOUD SERVICE COMPOSITION

In this section, the definition of the cloud service composition problem is first presented, followed by how to model cloud service composition as an influence diagram problem. In this section, we use the motivating scenario as a walk-through example.

4.1. Cloud Economic Model

To enable long-term cloud service composition, economic models are needed to represent the long-term preferences of the end-users and the long-term QoS values of cloud service providers. An economic model is defined as "a theoretical construct that represents economic processes by a set of variables and a set of logical and quantitative relationships between them." [Baumol and Blinder 2011]. More informally, an economic model represents the expected performance of a set of economic indicators over a period of time.

Our research focuses on using probabilistic theory to explicitly represent and reason about the relationship among variables in economic models. The essential idea of using probabilistic theory is that the relationships we are interested in will not be predictable in advance, but rather will exhibit an inherent variation that should be taken into account by the model. This is usually accomplished by allowing the model to be probabilistic in nature. We adopt Bayesian Networks (BN) [Jensen 1996] to represent the economic model in this reseach. BN is a probabilistic graphical model that represents a set of random variables and their conditional dependency using a directed acyclic graph. A BN consists of a set of random variables as nodes connected through directed links (arcs). Each node has a conditional probability table that quantifies the effects the parents have on the nodes.

End-users represent their preferences over multiple QoS attributes through a score function, hence the variables in the end-user's economic model are the weights for all the QoS attributes. If we represent these weights at different periods as nodes in a BN, we can then leverage the network as a means to represent the economic model for the end-users. For end-users, we make the assumption that all requests initialized at the same period have a score function with the same weight however, those initialized at different periods have different score functions.

Similarly, cloud service providers represent their performance using QoS values across a long period. The variables in the cloud service providers' economic model are the QoS values at different periods. For each cloud service provider, we further assume all the services invoked at the same period have the same QoS performance, however,

those invoked at different periods have different performance. Economic models for end-users and cloud service providers are constructed based on historical data.

4.2. Problem Definition

Based on the previous analysis this section presents the general definition of the cloud service composition problem.

Suppose an end-user has a series of requests including N component abstract SaaSs during a long period T. The execution path of the requests is presented as

$$Path = \{Sto_0, NIn_1, \tau_1, NOut_1, Sto_1, CPU_1, \ldots\ldots NIn_N, \tau_N, NOut_N, Sto_N\}, \quad (1)$$

where τ_i represents the tasks (abstract SaaS) in the abstract composite service, NIn_i and $NOut_i$ denote the network units needed to transfer input and output data for SaaS τ_i, Sto_i represents the storage units needed for SaaS τ_i, and CPU_i denotes the CPU units needed to adapt output data from SaaS τ_i. Here τ_0 is the starting point of the composite service, while Sto_0 denotes the storage IaaS needed for storing the input data of the composite service. Abstract SaaS are ordered as $[\tau_1, \ldots, \tau_N]$ and for every τ_i $(1 < i < N)$.

The end-user defines score functions for all the requests during period T. We denote the weights for each application requested by the end-user as

$$W(Path) = \{W(1), W(2), \ldots, W(T)\}. \quad (2)$$

To illustrate the composition problem, we use the three QoS attributes discussed earlier, but other QoS attributes can be used instead without any fundamental changes. The QoS dimensions are numbered from 1 to 3, with 1 equalling reputation, 2 throughput, and 3 cost. W^t is further denoted as a vector

$$W(t) = < w_1(t), w_2(t), w_3(t) >, \quad (3)$$

where $w_a(t)$ $(a = 1, 2, 3)$ denotes the weight for attribute a in execution path $Path$ during period $period_t$.

Suppose there is a set of K_i candidate SaaS providers that can be used to implement the task τ_i in an execution path $Path$. There is also a set of P_p candidate IaaS providers that supply IaaS to the end-users. A candidate composition plan is formed by selecting a single SaaS provider for each task and a single IaaS provider for the execution path. The composition plan is denoted as

$$Plan[SP_1(k_1), SP_2(k_2), \ldots, SP_n(k_n), IP(p)], \quad (4)$$

where task τ_i in execution path $Path$ is executed by SaaS provider $SP_i(k_i)$ and where the abstract IaaS in the execution path $Path$ is executed by IaaS provider $IP(p)$.

The QoS for a composition plan $q(Path)$ for the end-user is denoted as a matrix

$$q(Path) = \begin{bmatrix} q^{\tau_0}(1), & q^{\tau_0}(2), \ldots, & q^{\tau_0}(T) \\ q^{\tau_1}(1), & q^{\tau_1}(2), \ldots, & q^{\tau_1}(T) \\ \ldots\ldots & & \\ q^{\tau_N}(1), & q^{\tau_N}(2), \ldots, & q^{\tau_N}(T) \end{bmatrix}, \quad (5)$$

where each $q^{\tau_i}(t)$ represents the QoS values for the component service τ_i at period $period_t$. Each $q^{\tau_i}(t)$ is further denoted as a vector

$$q^{\tau_i}(t) = \left[q_1^{\tau_i}(t), \quad q_2^{\tau_i}(t), \ldots, \quad q_3^{\tau_i}(t) \right], \quad (6)$$

where $q_a^{\tau_i}(t)$ denotes the realized QoS value for attribute a of abstract SaaS τ_i in execution path $Path$ during period $period_t$. For the sake of illustration, we denote τ_0 as the task of selecting the IaaS provider for the execution path $Path$. The realized QoS

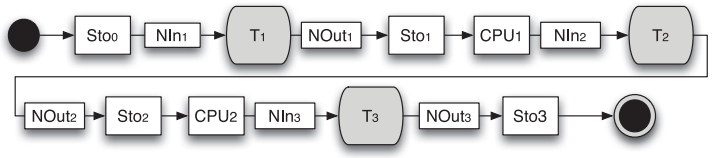


Fig. 3. Execution path for the tenure application.

values for each execution path are computed using the aggregation functions shown in Table I. We denote the QoS values for QoS attribute a of the execution path $Path$ in period t as $q_a(t)$, where $a = 1, 2, 3$ are

$$q_a(t) = aggre\left(q_a^{\tau_0}(t), q_a^{\tau_1}(t), \ldots, q_a^{\tau_N}(t)\right). \tag{7}$$

Each composition plan has a score to the end-user. A commonly used score function is the sum of the weighted QoS values defined based on the realized QoS values and the weights assigned by the end-user. For a realized execution path ($Path$), the score of this execution path during period $period_t$ is denoted as

$$s(t) = \sum_{a=1}^{3} w_a(t) \cdot q_a(t). \tag{8}$$

For a composition plan, the score is the sum of all the QoS scores over the long period.

$$S(Plan) = \int_1^T s(t)dt \tag{9}$$

Different combinations of SaaS providers and IaaS providers generate multiple composition plans for the end-user. The cloud service composition problem is to find an optimal composition plan for the end-user.

For our motivating example, the company EasyPlanner has an anstract composite service (or execution path) as showm in Figure 3. There are three component abstract SaaSs in this execution path. The composition should be last from 2013 to 2023. Suppose each period is a week, hence there are $T = 520$ periods in this example. In the following sections, an influence diagram technique is used to model this composition problem and an evaluation solution is provided to the influence diagram problem.

4.3. Problem Analysis

This research follows the idea of Zeng et al. [2003] in considering the composition problem from a decision-making perspective. Cloud service composition can be considered as a decision process involving different levels of knowledge in time. For an end-user, the composition framework needs to decide which concrete SaaS providers and IaaS provider the end-user should select. These decisions include those for each component abstract SaaS and the abstract IaaS.

A brute-force approach to compute an optimal composition plan is to generate all the possible candidate composition plans, consider all the possible scenarios for each plan regarding the weights, and compute the score and the probability rate for each scenario. The optimal composition plan is the one having the highest weighted sum score. This brute-force approach causes a high computation complexity when there is a large amount of (or infinite) possible scenarios, therefore a scalable solution is needed to compute the optimal composition plan. This research adopts the *influence diagram* to represent and solve the cloud service composition problem.

Influence Diagrams (IDs) [Shachter 1988] are graphical models for representing and solving complex decision-making problems based on uncertain information. IDs are

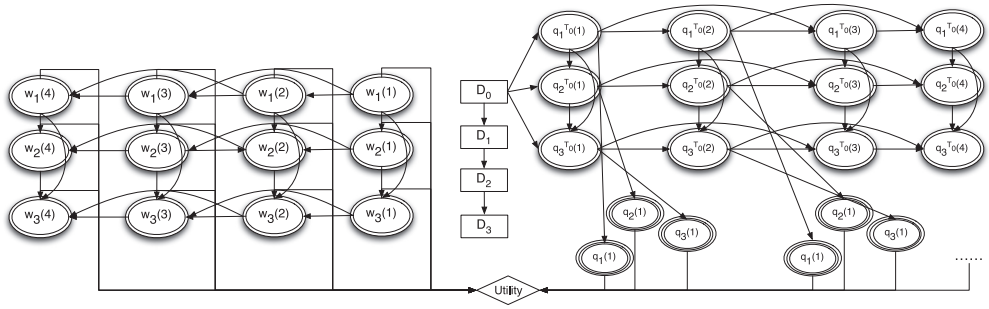


Fig. 4. Influence diagram for the trip planning example.

directed acyclic graphs seen as a BN augmented with decision and value nodes. An ID is a triplet (N, A, P): $N = C \cup D \cup U$, where C is the set of chance nodes, D a set of decision nodes, and U a set of utility nodes. Ω_x represents all the possible values of node x and moreover A is the set of directed arcs between the nodes. Arrows pointing from chance variables to decision variables indicate the information available to the decision-maker when choosing a value for that decision variable. Arrows pointing from one decision variable to another indicate the temporal order of the decisions in the problem. In the case of both types of arrows, the optimal value for the decision variable at the tails of these arrows must be determined as a function of the observed value of the variables at the heads of the arrows. The no-forgetting principle [Shachter 1988] is assumed to hold, meaning that the strategy for each decision variable is a function of observed chance variables and previously made decisions.

Regarding the cloud service composition problem stated before, we model the cloud service composition problem as an ID problem as follows. We represent the weights from end-users and the advertising QoS values from cloud providers as chance nodes in an ID. We add conditional probabilistic relationships among these advertising QoS values from cloud service providers and all the selection decisions on abstract SaaS and IaaS are represented as decision nodes in the ID. Utility nodes represent the score of a composition plan.

4.4. Hybrid ID

Unlike traditional ID problems, cloud service composition problems are *hybrid IDs* [Li and Shenoy 2010]. Hybrid IDs are IDs containing a mix of discrete and continuous chance and decision nodes. We focus on a finite set $V = D \cup C$ of variables. Variables in D are called decision variables whereas those in C are called chance variables. Each variable $X \in V$ is associated with a set Ω_X of possible states. X is discrete if Ω_x is finite or countable, otherwise X is continuous. Continuous variables with nondeterministic conditionals are denoted as continuous [Li and Shenoy 2010], while those with deterministic conditionals are denoted as *deterministic* [Li and Shenoy 2010]. Let C_d denote the set of all discrete chance variables and let C_c the set of all continuous chance variables. Then, $C = C_d \cup C_c$.

Figure 4 shows the influence diagram representation for the trip planning example. In this ID, D_0 represents the decision to select the concrete IaaS provider for the composite service and D_i $(i = 1, 2, 3)$ denotes the decision to select concrete SaaS providers for each abstract SaaS τ_i. The values of these decision nodes are discrete, that is, the identification of the candidate SaaS providers or IaaS providers. $w_a(t)$ represents the defined weight for QoS attribute a $(a = 1, 2, 3)$ during period $period_t$, whereas $q_a^{\tau_i}(t)$ denotes the realized value for QoS attribute a of the abstract SaaS τ_i during period $period_t$. Finally, $q_a(t)$ denotes the aggregated value of attribute a during

period $period_t$. These chance nodes are continuous in our example and normally a real number within a range, for instance, weights are real numbers between 0 and 1. The utility node represents the score for the composition plan and is computed using the utility function as shown in Eq. (9).

In the diagram representation, we depict decision variables as rectangular nodes, discrete chance variables by single-bordered elliptical nodes, continuous chance variables by double-bordered elliptical nodes, deterministic chance variables by triple-bordered elliptical chance nodes, and the utility function by diamond-shaped nodes. Regarding the tenure example, all the chance nodes are continuous nodes. Nodes $q_a(t)$, where $a = 1, 2, 3$, are deterministic (shown in Figure 4).

The conditional probability functions associated with variables in an ID are represented by functions called *potentials*. The discrete potential for a discrete chance node r is a function

$$\alpha(r) : \Omega_r \rightarrow [0, 1], \tag{10}$$

The values of discrete potentials are probabilities. Continuous chance variables are typically associated with conditional Probability Density Functions (PDFs). The density potential ζ for a continuous chance node r is a function

$$\zeta(r) : \Omega_r \rightarrow \mathbb{R}^+, \tag{11}$$

where $\mathbb{R}+$ is the set of nonnegative real numbers. The values of density potentials are probability densities. Deterministic variables have conditional distributions described by *Dirac potentials* that use Dirac delta functions δ [Dirac 1927]. Readers can refer to Dirac [1927] and Li and Shenoy [2010] for more information about Dirac delta functions and Dirac potentials. The utility potential ν for a utility node $t \subseteq N$ is a function

$$\nu(t) : \Omega_t \rightarrow \mathbb{R}. \tag{12}$$

The values of utility potentials are in units of utiles.

A mixed potential has three parts, namely a discrete potential, a continuous potential (that can be a density potential or Dirac potential), and a utility potential. Suppose α is a discrete potential for r. Then, a mixed potential representation of α is $\mu = (\alpha, \emptyset_c, \emptyset_u)$, where $\emptyset_c$ denotes the identity continuous potential for $\emptyset$ and where $\emptyset_u$ denotes the identity utility potential for $\emptyset$. Suppose ζ is a continuous potential for s. Then, a mixed potential representation of ζ is $\mu = (\emptyset_d, \zeta, \emptyset_u)$, where $\emptyset_d$ denotes the discrete identity potential for $\emptyset$. Finally, if ν is a utility potential for t, then a mixed potential representation of ν is $\mu = (\emptyset_d, \emptyset_c, \nu)$. Moreover, $\emptyset_d$ is a discrete potential for $\emptyset$ whose value is in units of probability, $\emptyset_c$ is a continuous potential for $\emptyset$ whose value is in units of density, and $\emptyset_u$ is a utility potential for $\emptyset$ whose value is in units of utiles.

4.5. Solution to Hybrid ID

This research adopts an extended Shenoy-Shafer architecture [Li and Shenoy 2010] to solve hybrid IDs that include deterministic chance nodes.

The main idea of the solution is to use *Mixture-Of-Polynomials* (MOP) approximations for PDFs and utility functions. All variables in the hybrid ID need to be marginalized in a sequence that respects the information constraints in the sense that, if X precedes Y in the information sequence, then Y must be marginalized before X. Each time when marginalizing a decision variable, we keep track of where the maximum is attained (as a function of the remaining variables in the potential being marginalized). This yields a decision function for the decision variable. The collection of all decision functions constitutes an optimal strategy for the influence diagram.

Algorithm 1 shows the pseudocode for the proposed solution. The algorithm first generates the marginalization sequence of the ID (lines 1–11). The sequence is formed

ALGORITHM 1: Solution of hybrid ID

```
   Input: ID^{-U} ← The influence diagram without the utility node
   Order ← Empty stack that represents the marginalization sequence
   S ← Set of all nodes with no outgoing arcs in ID^{-U}
   Plan ← Empty set that represents the composition plan
 1 for each node n in S do
 2 |  visit(n);
 3 end
 4 function visit(node n)
 5 if n has not been visited yet then
 6 |  mark n as visited;
 7 |  for each node m with an arc from m to n do
 8 |  |  visit(m);
 9 |  end
10 |  push n to Order;
11 end
12 node ← pop from Order
13 while node ≠ NULL do
14 |  if node = decision node then
15 |  |  add marginal(node, ID^{-U}) to Plan;
16 |  else
17 |  |  marginal(node, ID^{-U})
18 |  end
19 |  ID^{-U} ← ID^{-U} delete node node
20 |  node ← pop from Order
21 end
```

by pushing the topological sorting order into a stack, where the node pushed last will be the first to be marginalized. Each time when a node is marginalized (lines 12–21), the algorithm will invoke the *marginal function* (detailed in the following section). If the node is a decision node, the algorithm will store the decision strategy in the composition plan set.

Marginalization of Potentials. The definition of marginalization of potentials depends on the nature of the variable being marginalized. We marginalize discrete chance variables by addition over their state space, continuous chance variables by integration over their state space, and decision variables by maximization over their state space. Suppose α is a potential (discrete, continuous, utility, or some combination thereof) for a and suppose $X \in a$ is a discrete variable. Then, the marginal of α by deleting X, denoted by α^{-X}, is a potential for $a\backslash\{X\}$ given as follows.

$$\alpha^{-X}(y) = \sum\{\alpha(x, y)|x \in \Omega_X\} \text{ for all } y \in \Omega_{a\backslash\{X\}} \tag{13}$$

If $X \in a$ is a continuous variable, then α^{-X} is defined as

$$\alpha^{-X}(y) = \int_{-\infty}^{\infty} \alpha(x, y)dx \text{ for all } y \in \Omega_{a\backslash\{X\}}, \tag{14}$$

whereas if $X \in a$ is a decision variable, then α^{-X} is defined as

$$\alpha^{-X}(y) = max\{\alpha(x, y)|x \in \Omega_X\} \text{ for all } y \in \Omega_{a\backslash\{X\}}. \tag{15}$$

Marginalization of Mixed Potentials. Similar to the marginalization of potentials, marginalization of mixed potentials depends on the nature of the variable being marginalized. We distinguish between marginalizing a decision variable and

ALGORITHM 2: Function: marginal(X, ID)

```
1  if X ∈ D then
2      switch(X)
3      case(1): X ∉ r, X ∉ s, X ∈ t
4              return (α, ζ, ν^{-X})
5      case(2): X ∈ r, X ∉ s, X ∈ t
6              return (∅_d, ζ, (α ⊗ ν)^{-X})
7      case(3): X ∉ r, X ∈ s, X ∈ t
8              return (α, ∅_c, (ζ ⊗ ν)^{-X})
9      case(4): X ∈ r, X ∈ s, X ∈ t
10             return (∅_d, ∅_c, (α ⊗ ζ ⊗ ν)^{-X})
11 else
12     switch(X)
13     case(1): X ∉ r, X ∉ s, X ∈ t
14             return (α, ζ, ν^{-X})
15     case(2): X ∈ r, X ∈ s, X ∉ t, (r ∪ s)\X ⊆ C_d
16             return ((α ⊗ ζ)^{-X}, ∅_c, ν)
17     case(3): X ∈ r, X ∈ s, X ∉ t, (r ∪ s)\X ⊈ C_d
18             return (∅_d, (α ⊗ ζ)^{-X}, ν)
19     case(4): X ∉ r, X ∈ s, X ∉ t, s\X ⊈ C_d
20             return (α, ζ^{-X}, ν)
21     case(5): X ∉ r, X ∈ s, X ∉ t, s\X ⊆ C_d
22             return (α ⊗ ζ^{-X}, ∅_c, ν)
23     case(6): X ∈ r, X ∉ s, X ∉ t, r\X ⊈ C_c
24             return (α^{-X}, ζ, ν)
25     case(7): X ∈ r, X ∉ s, X ∉ t, r\X ⊆ C_c
26             return (∅_d, α^{-X} ⊗ ζ, ν)
27 end
```

marginalizing a chance variable from a mixed potential. Suppose $\mu = (\alpha, \zeta, \nu)$ is a mixed potential for $r \cup s \cup t$, where α is a discrete potential for r, ζ is a continuous potential for s, and ν is a utility potential for t. Algorithm 2 shows the pseudocode for the marginal function.

Suppose $X \in D$ and $X \in r \cup s \cup t$ (lines 1–10). We assume each decision variable is the domain of at least one utility potential. Hence, at the time when a decision variable is to be marginalized from a mixed potential, it will always be in the domain of the utility part. Therefore, we identify four cases (lines 3–10) where a decision node is marginalized.

Suppose $X \in C$ and $X \in r \cup s \cup t$. We divide the marginal of μ by deleting X, denoted as μ^{-X}, into seven cases. Since we assume there is only one single utility potential in the ID representation, we do not need to consider the conditions corresponding to arc reversal in influence diagrams [Shachter 1986]. The last six cases in which the chance variable being marginalized does not belong to the domain of the utility potential are exactly as discussed in Shenoy and West [2011]. The combinations (denoted as $\otimes$) of potentials are detailed in the next section.

Combination of Potentials. The definition of combination of potentials depends on the nature (discrete, continuous, or utility) of potentials. Although there are nine permutations, we have only two distinct definitions. Utility functions are additive factors of the joint utility function, thus the combination of two utility potentials involves pointwise addition. In the other eight cases, a combination of potentials involves pointwise multiplication.

Suppose ν_1 is a utility potential for t_1 and ν_2 a utility potential for t_2. Then, the combination of ν_1 and ν_2, denoted by $\nu_1 \otimes \nu_2$, is a utility potential for $t_1 \cup t_2$ given by

$$(\nu_1 \otimes \nu_2)(x) = \nu_1(x^{\downarrow t_1}) + \nu_2(x^{\downarrow t_2}) \text{ for all } x \in \Omega_{t_1 \cup t_2}. \quad (16)$$

Suppose α_1 is a potential (discrete, continuous, or utility) for t_1 and α_2 a potential (discrete, continuous, or utility) for t_2. Suppose that both α_1 and α_2 are not both utility. Then, the combination of α_1 and α_2, denoted as $\alpha_1 \otimes \alpha_2$, is a potential for $t_1 \cup t_2$ given by

$$(\alpha_1 \otimes \alpha_2)(x) = \alpha_1(x^{\downarrow t_1})\alpha_2(x^{\downarrow t_2}) \text{ for all } x \in \Omega_{t_1 \cup t_2}. \quad (17)$$

If α_1 and α_2 are both discrete, then $\alpha_1 \otimes \alpha_1$ is a discrete potential, whereas if α_1 and α_1 are both continuous, then $\alpha_1 \otimes \alpha_1$ is a continuous potential. Finally, if α_1 is discrete or continuous, and α_1 is utility, or vice versa, then $\alpha_1 \otimes \alpha_1$ is a utility potential. In all other cases, we will define the nature of the combined potential when we define marginalization of mixed potentials.

Combination of Mixed Potentials. Suppose $\mu_1 = (\alpha_1, \zeta_1, \nu_1)$ and $\mu_2 = (\alpha_2, \zeta_2, \nu_2)$ are two mixed potentials for $r_1 \cup s1 \cup t1$ and $r2 \cup s2 \cup t2$, respectively, with discrete parts α_1 for r_1 and α_2 for r_2, continuous parts ζ_1 for s_1 and ζ_2 for s_2, and utility parts ν_1 for t_1 and ν_2 for t_2, respectively. Then, the combination of μ_1 and μ_2, denoted by $\mu_1 \otimes \mu_2$, is a mixed potential for $(r_1 \cup s_1 \cup t_1) \cup (r_2 \cup s_2 \cup t_2)$ given by

$$\mu_1 \otimes \mu_2 = (\alpha_1 \otimes \alpha_2, \zeta_1 \otimes \zeta_2, \nu_1 \otimes \nu_2). \quad (18)$$

Mixture-Of-Polynomials Approximations. At last, we describe MOP functions used in the algorithm. A one-dimensional function $f : \mathbb{R} \to \mathbb{R}$ is said to be a Mixture-Of-Polynomials (MOP) function if it is a piecewise function of the form

$$f(x) = \begin{cases} a_{0i} + a_{1i}x + \ldots + a_{ni}x^n \text{ for} & x \in A_i, i = 1, \ldots, k, \\ 0 & \text{otherwise} \end{cases}$$

where $A_1, \ldots, A_k$ are disjoint intervals in $\mathbb{R}$ that do not depend on x and where $a_{0i}, \ldots, a_{ni}$ are constants for all i. Therefore $f(x)$ is denoted as a k-piece and n-degree (assuming $a_{ni} \neq 0$ for some i) MOP function. The main motivation for defining MOP functions is that such functions are easy to integrate in closed form and are closed under multiplication and integration [Shenoy and West 2009b] as well as under differentiation and addition.

An m-dimensional function $f : \mathbb{R}^m \to \mathbb{R}$ is said to be a MOP function if

$$f(x_1, \ldots, x_m) = f_1(x_1) \cdot f_2(x_2) \ldots f_m(x_m),$$

where each $f_i(x_i)$ is a one-dimensional MOP function as defined before. If $f_i(x_i)$ is a k_i-piece, n_i-degree MOP function, then f is a $(k_1 \cdot k_2 \cdots k_m)$-piece, $(n_1 + \cdots + n_m)$-degree MOP function. Hence it is important to keep the number of pieces and degrees to a minimum. Shenoy and West [2009b] discuss the process of finding MOP approximations of univariate and bivariate conditional distributions.

5. EXPERIMENTS AND RESULTS

We conducted a set of experiments to assess the performance of the proposed composition framework. We use the trip planning scenario as our testing environment to set up the experiment parameters. The purpose is to demonstrate how the proposed composition framework can help the company EasyPlanner to select the best composition plan. We run our experiments on a Macbook Pro with 2.2 GHz Intel Core i7 processor and 4G RAM under Mac OS X 10.7.3. Our experiments mainly focus on evaluating the proposed approach using synthetic cloud services.

5.1. Experiment Setup

We set up a set of experiment parameters to evaluate the performance of the CloudCom framework. In the ID representation shown in Figure 4, D_i, denoting the candidate SaaS providers for abstract SaaS τ_i in the execution path $Path$, can be any integer in the interval $[1, 2, \ldots, M]$. Values for continuous chance nodes $q_a^{\tau_i}(t)$ and $w_a(t)$ can be any real numbers.

The chance nodes $q_a^{\tau_i}(1)$ that denote the advertising QoS values at period $period_1$ for the abstract SaaS T_i in the execution path $Path$, have PDF represented as a Gaussian function $N(B_{a,i}, (C_{a,i})^2)$, where B and C are predefined constants. The chance nodes $q_a^{T_i}(t), t \geq 2$ have the conditional PDF represented as

$$q_a^{T_i}(t)|q_a^{T_i}(t-1) \Leftarrow N\big(q_a^{T_i}(t-1), (C_{a,i})^2\big). \tag{19}$$

The deterministic nodes $q_a(t)$, which denote the value for a QoS attribute a of the execution path $Path$ at period $period_t$, are computed using the aggregation functions shown in Table I.

5.2. Accuracy of the Framework

To study the accuracy of the CloudCom framework, we set the constant values as follows: a, indicating the QoS attribute considered, is set to 5, while N, representing the number of abstract SaaS, is set to 6. Finally, t, denoting the period, is set to 1. Each abstract SaaS and IaaS can be implemented by $M = 4$ cloud providers. W_a, $a = 1, 2, \ldots, 5$ is set to $\frac{1}{5}$. Without loss of generality, we assume that all the $B_{a,i}$ have the same value B and that all the $C_{a,i}$ have same value C. We conduct five experiments by changing the values for B and C. Figure 5 shows the comparison results.

We compare the results generated by the proposed CloudCom framework with those of a brute-force approach. The brute-force method is to compute the weighted sum score for each possible composition plan and return the optimal composition plan with the highest score. Infinite possible scenarios exist for a composition plan. To compute the weighted sum score (which is the product of the possibility and the QoS values), we restrict the values for the QoS attribute to be integers only in the range of $(-3, 3)$. The four test cases are denoted as (B, C): $(0, 1)$, $(0, 2)$, $(1, 1)$, and $(1, 2)$. From the experiment results (Figure 5), we can see that the proposed approach can always find a better composition solution compared to the brute-force method.

5.3. Scalability of the Framework

To study the scalability of the framework, we focus on how the response time varies as the problem becomes larger. This is done by correspondingly changing the parameters T and M. While the values for B and C are kept constant. By increasing the values for T, the composition problem becomes long-term based, while by increasing the values for M, the candidate composition plans scale to a larger set. Figure 7(a) shows the response time when the considered period increases, whereas Figure 7(b) shows the response time when the number of candidate cloud providers increases. When the period increases, the problem becomes the integration of polynomials with a large number of variables. The time complexity of this integration problem has an exponential (e^T) lower bound [Fu 2012]. When the candidate cloud providers scale, the proposed approach has polynomial time complexity (M^6). On the other hand, the brute-force approach has exponential time complexity ($M^{N \cdot T}$).

In addition, we compare the proposed approach with a traditional Web service composition approach, such as that shown in Zeng et al. [2004]. The traditional technique selects and composes services based on short-term QoS values (here, *short term* means that the selection and composition of services are based on the QoS performance at

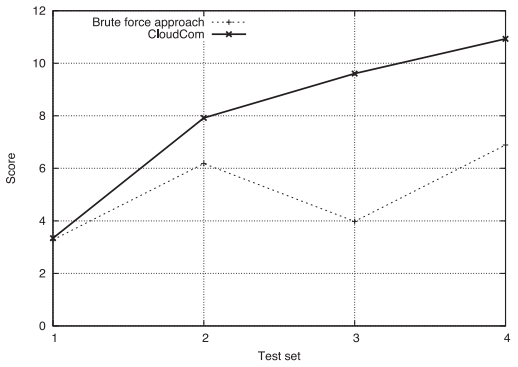


Fig. 5. CloudCom vs. brute-force approach.

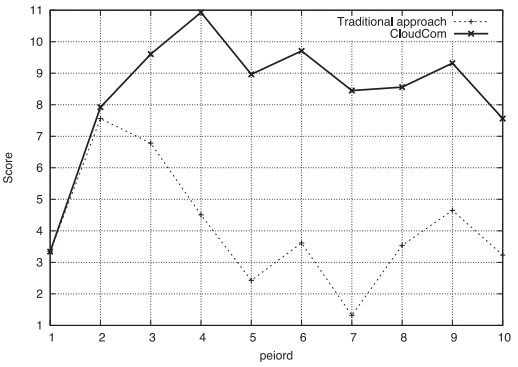
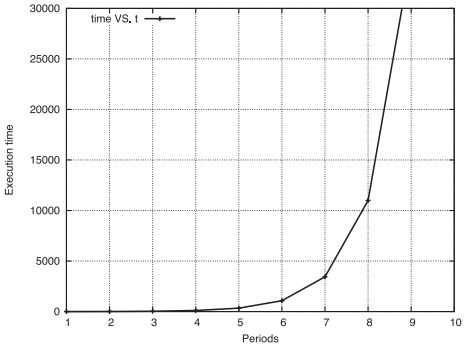
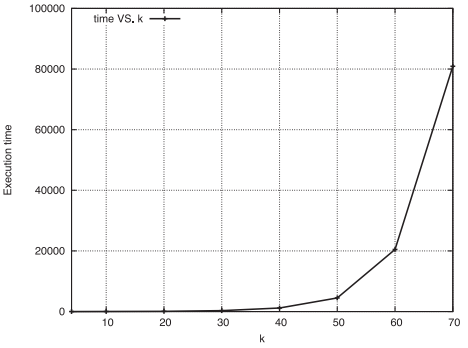


Fig. 6. CloudCom vs. traditional composition approach.



(a) parameter t



(b) parameter k_i^j

Fig. 7. Scalability of the framework.

query time). As in the motivating example, the traditional approach would find the best composition based solely on its QoS performance during the first period. In this experiment, we change the value of T and compare the score computed using both the proposed and traditional approaches. We run five tests for each value of T and the ultimate value of the score is the average of these five tests. Figure 6 shows the results of our experiments where T varies from 1 to 10 while other parameters are kept constant. When the value of T is small, for instances, 1 or 2, the scores are more or less similar. However, when the value of T increases, the proposed approach will always find better composition plans than the traditional approach. These results are consistent with the observation that the proposed approach can always find a better composition plan, whether with a short-term or a long-term composition problem.

6. RELATED WORK

Cloud service composition is different from traditional composition approaches. In this section, we present the related work about traditional service composition solutions in SOC followed by the related work about ID valuation solutions in other areas.

6.1. Service Composition Approaches

Service composition is an active research area in service-oriented computing [Medjahed et al. 2003]. Service composition problems can be categorized into two groups: (1) those

focusing on the functional composability among component services and (2) those that aim to make optimal decisions on selecting the best component services based on nonfunctional properties (QoS).

Functionally-driven service composition approaches typically adopt semantic descriptions of services. Examples of automatic approaches include the policy-based approach proposed by Chun et al. [2005] and the composability-model-driven approach sugested by Medjahed et al. [2003]. Other functionally-driven composition approaches use AI planning methods. Most of them assume [McIlraith and Son 2002; Ponnekanti and Fox 2002; Wu et al. 2003] that each service is an action which alters the state of the world as a result of its execution. The input and output parameters of a service act as preconditions and effects in the planning context. Users only need to specify the inputs and outputs of the desired composite service and a plan (or a composite service) would automatically be generated by the AI planners.

Functionally-driven service composition methods mostly do not attempt to find an optimal solution but only to find a solution. However, the nonfunctional properties (QoS) of the resulting composite service are determinant factors to ensure customer satisfaction. Different users may have different requirements and preferences regarding QoS. Therefore, QoS-aware composition approaches are needed. The QoS-aware service composition problem is usually modeled as a multiple-criteria decision-making [Zeng et al. 2003] problem. The most popular approaches include integer linear programming and genetic algorithms. An Integer Linear Program (ILP) consists of a set of variables, a set of linear constraints, and a linear objective function. After having translated the composition problem into this formalism, specific solver software such as LPSolve [Berkelaar et al. 2004] can be used [Zeng et al. 2003; Ardagna and Pernici 2007]. Canfora et al. [2005] and Ye et al. [2011] use Genetic Algorithms (GA) for service composition. Individuals of the population correspond to different composition solutions, their genes to the abstract component services, and the possible gene values to the available real services. While GAs do not guarantee to find the optimal solution, they can be more efficient than ILP-based methods (that have exponential worst-case time complexity).

Most of the existing composition approaches are not well suited for a cloud environment [Ye et al. 2011]. They usually consider the qualities at the time of the composition [Medjahed et al. 2003], however, service composition in cloud computing needs to be considered from a long-term perspective.

6.2. Influence Diagram Approaches

A traditional method for solving hybrid IDs is to approximate a hybrid ID with a discrete ID by discretizing the continuous chance and decision variables [Keefer 1994]. If we discretize a continuous variable using a few bins, we may have an unacceptable approximation of the problem. On the other hand, if we use many bins, we increase the computational effort of solving the resulting discrete ID. In the BN literature, Kozlov and Koller [1997] describe a dynamic nonuniform discretization technique for chance variables depending on where the posterior density lies. This technique needs to be adapted for solving hybrid IDs.

One of the earliest works on using continuous variables in IDs is by Shachter and Kenley [1989]. In their representation, called Gaussian IDs (GIDs), all chance variables are continuous, having the so-called Conditional Linear Gaussian (CLG) distribution. This is a Gaussian PDF whose mean is a linear function of its parents and whose variance is a constant. GIDs also assume that all decision nodes are continuous and that the utility function is a quadratic function of its parent chance and decision variables. Poland and Shachter [1993] extend GIDs to Mixture-of-Gaussian IDs (MoGIDs). MoGIDs have discrete and continuous chance variables. As in GIDs, continuous

variables have a CLG distribution whose mean is a linear function of its continuous parents, as well as a constant variance. However, discrete nodes cannot have continuous parents.

Another strategy [Moral et al. 2001] for easing the problem of integration is to approximate conditional PDFs by mixtures of exponential functions whose exponent is a linear function of the state of the variable and its continuous parents. Each mixture component (or piece) is restricted to a hypercube. One advantage of this approximation is that such Mixtures of Truncated Exponentials (MTEs) are easy to integrate in closed form. In the same spirit as MTEs, Shenoy and West [2011] suggest the use of Mixture Of Polynomials (MOPs) to approximate conditional PDFs. Like MTE functions, MOP functions are easy to integrate and the family of MOP functions is closed under multiplication and integration. Unlike MTEs, however, finding MOP approximations is easier for differentiable functions as one can use the Taylor series expansion to find a MOP approximation. Finding an MTE approximation for a multidimensional distribution (such as the conditional for a variable given different continuous variables as parents) can be difficult, whereas finding a MOP approximation is easier because it can be found using the multidimensional version of the Taylor series.

In an ID, a continuous chance variable is said to be deterministic if the variances of its conditional distributions (for each state of its parents) are all zeroes. Deterministic variables pose a special problem since the joint density for all the chance variables does not exist. Cobb and Shenoy [2005a] suggest approximating a nonlinear deterministic function by a piecewise linear function and then using the technique proposed in Cobb and Shenoy [2005b]. Cinicioglu and Shenoy [2009] describe an arc reversal theory for hybrid Bayesian networks with deterministic variables with a differentiable deterministic function. They use Dirac delta functions [Dirac 1927] to represent deterministic functions where the deterministic function does not have to be linear or even invertible. The approach adopted in this research is a further extension of Shenoy and West [2009a] for solving influence diagrams containing discrete, continuous, and deterministic variables.

7. CONCLUSION

This research proposes a cloud service composition framework named CloudCom to aid end-users in selecting and composing SaaS providers and IaaS providers in the cloud environment. Compared to the traditional service composition framework in SOC, CloudCom considers service composition from a long-term perspective. An economic model represented using a Bayesian network is used to select and compose cloud services. In addition, a hybrid influence diagram approach is adopted to solve cloud service composition problems. Specifically, the main contributions of this article included the following.

—*Economic model.* We use Bayesian network to represent the cloud economic model. The preferences and behaviors of end-users are modeled using a conditional probabilistic relationship.

—*Decision analysis approach.* The cloud service composition problem is modeled from a decision analysis perspective. For an end-user, the composition framework has several decisions to make during the composition procedure. Influence diagrams are proposed to represent and solve the composition problem.

From the experiments, we can see that the proposed composition framework can find long-term-based and economic-model-driven cloud service composition. It performs better than the brute-force and traditional composition approaches. To the best of our knowledge, there are few existing works to compose cloud services. This article is our first step towards a coherent cloud service composition framework. Whereas there are

a number of different ways to represent an economic model (e.g., trend model, market-based model), the problem to find an optimal composite cloud service that has the best trend regarding the QoS performance is still an open question. Future work includes optimizing the composition framework to reduce the time complexity and adapting the BN-based economic model to make is suitable for other general cloud service composition scenarios.

REFERENCES

D. Ardagna and B. Pernici. 2007. Adaptive service composition in flexible processes. *IEEE Trans. Softw. Engin.* 33, 6, 369–384.

M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica et al. 2009. Above the clouds: A berkeley view of cloud computing. Tech. rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

W. J. Baumol and A. S. Blinder. 2011. *Economics: Principles and Policy*. Cengage Learning.

M. Berkelaar, K. Eikland, and P. Notebaert. 2004. Lpsolve: Open source (mixed-integer) linear programming system. Tech. rep., Eindhoven University of Technology.

G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. 2005. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05).* 1069–1075.

S. A. Chun, V. Atluri, and N. R. Adam. 2005. Using semantics for policy-based web service composition. *Distrib. Parallel Databases* 18, 1, 37–64.

E. N. Cinicioglu and P. P. Shenoy. 2009. Arc reversals in hybrid Bayesian networks with deterministic variables. *Int. J. Approx. Reason.* 50, 5, 763–777.

B. Cobb and P. Shenoy. 2005a. Nonlinear deterministic relationships in bayesian networks. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'05).* 27–38.

B. R. Cobb and P. P. Shenoy. 2005b. Hybrid bayesian networks with linear deterministic variables. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

D. Dash, V. Kantere, and A. Ailamaki. 2009. An economic model for self-tuned cloud caching. In *Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE'09).* 1687–1693.

P. A. M. Dirac. 1927. The physical interpretation of the quantum dynamics. *Proc. Royal Soc. London A1927*, 113.

S. Dustdar and W. Schreiner. 2005. A survey on web services composition. *Int. J. Web Grid Serv.* 1, 1, 1–30.

B. Fu. 2012. Multivariate polynomial integration and differentiation are polynomial time inapproximable unless p= np. In *Proceedings of the 6th International Frontiers in Algorithmics, and Proceedings of the 8th International Conference on Algorithmic Aspects in Information and Management (FAW-AAIM'12).* 182–191.

A. Haque, S. M. Alhashmia, and R. Parthibanb. 2011. A survey of economic models in grid computing. *Future Gener. Comput. Syst.* 27, 8, 1056–1069.

F. V. Jensen. 1996. *An Introduction to Bayesian Networks*. UCL Press, London.

V. Kantere, D. Dash, G. Francois, S. Kyriakopoulou, and A. Ailamaki. 2011. Optimal service pricing for a cloud cache. *IEEE Trans. Knowl. Data Engin.* 23, 9, 1345–1358.

D. L. Keefer. 1994. Certainty equivalents for three-point discrete-distribution approximations. *Manag. Sci.* 40, 6, 760–773.

A. V. Kozlov and D. Koller. 1997. Nonuniform dynamic discretization in hybrid networks. In *Proceedings of the 13th International Conference on Uncertainty in Artificial Intelligence (UAI'97).*

Y. Li and P. P. Shenoy. 2010. Solving hybrid influence diagrams with deterministic variables. In *Proceedings of the 26th International Conference on Uncertainty in Artificial Intelligence (UAI'10).*

F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. 2011. NIST cloud computing reference architecture. http://www.cs.cmu.edu/~garth/15719/papers/nist_cloud_computing_reference.pdf.

X. Liu, C. Liu, M. Rege, and A. Bouguettaya. 2010. Semantic support for adaptive long term composed services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'10).* 267–274.

S. Mcilraith and T. C. Son. 2002. Adapting golog for composition of semantic web services. In *Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (KR'02).* 482–493.

B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. 2003. Composing web services on the semantic web. *The VLDB J.* 12, 4, 333–351.

N. Milanovic and M. Malek. 2004. Current solutions for web service composition. *IEEE Internet Comput.* 8, 6, 51–59.

S. Moral, R. Rumi, and A. Salmeron. 2001. Mixtures of truncated exponentials in hybrid bayesian networks. In *Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'01).* 156–167.

H. R. Motahari-Nezhad, B. Stephenson, and S. Singhal. 2009. Outsourcing business to cloud computing services: Opportunities and challenges. *IEEE Internet Comput. 10.*

W. B. Poland and R. D. Shachter. 1993. Mixtures of gaussians and minimum relative entropy techniques for modeling continuous uncertainties. In *Proceedings of the 9th International Conference on Uncertainty in Artificial Intelligence (UAI'93).* Morgan Kaufmann, San Fransisco, 183–190.

S. R. Ponnekanti and A. Fox. 2002. Sword: A developer toolkit for web service composition. In *Proceedings of the International Conference on World Wide Web (WWW'02).*

R. D. Shachter. 1986. Evaluating influence diagrams. *Oper. Res.* 34, 6.

R. D. Shachter. 1988. Probabilistic inference and influence diagrams. *Oper. Res.* 36, 4.

R. D. Shachter and C. R. Kenley. 1989. Gaussian influence diagrams. *Manag. Sci.* 35, 527–550.

P. P. Shenoy and J. C. West. 2009a. Inference in hybrid bayesian networks with deterministic variables. In *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09).* 46–58.

P. P. Shenoy and J. C. West. 2009b. Mixtures of polynomials in hybrid bayesian networks with deterministic variables. In *Proceedings of the International Workshop on Uncertainty Processing (WUPES'09).*

P. P. Shenoy and J. C. West. 2011. Inference in hybrid bayesian networks using mixtures of polynomials. *Int. J. Approx. Reason.* 52, 5, 641–657.

M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. 1994. An economic paradigm for query processing and data migration in mariposa. In *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems (PDIS'94).* 58–67.

D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. 2003. Automating daml-s web services composition using shop2. In *Proceedings of the 2nd International Semantic Web Conference (ISWC'03).* 195–210.

Z. Ye, X. Zhou, and A. Bouguettaya. 2011. Genetic algorithm based qos-aware service compositions in cloud computing. In *Proceedings of the 16th International Conference on Database Systems for Advanced Applications (DASFAA'11).* 321–334.

Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. 2008. Deploying and managing web services: Issues, solutions, and directions. *The VLDB J.* 17, 3, 537–572.

L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. 2003. Quality driven web services composition. In *Proceedings of the International Conference on World Wide Web (WWW'03).* 411–421.

L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. 2004. QoS-aware middleware for web services composition. *IEEE Trans. Softw. Engin.* 30, 5, 311–327.

Received July 2013; revised April 2014; accepted July 2014