Interactive QoS-aware Services Selection for the Internet of Things

Pegah ALIZADEH[1], Aomar OSMANI[1], Abdelghani CHIBANI[2], Yacine AMIRAT[2], Mohamed Essaid KHANOUCHE[2]
[1] Laboratoire LIPN-UMR CNRS 7030. PRES Sorbonne Paris-cité, FRANCE
[2] Laboratoire LISII Université UPEC, FRANCE

Abstract—Internet of Things (IoT) services composition combines individual services to create more powerful services for answering end users needs. Individual services are provided by internet of things components or any web service. Dealing with the services composition optimization process is crucial in the context of IoT. To improve the service composition process, the main used non-functional parameter is Quality of Service (QoS) which is represented by a set of criteria. In this paper, We assume that QoS is presented as a linear combination of these criteria. Then, we propose a multi-objective Markov Decision Process (MDP) for optimizing the QoS-aware services composition process. The proposed multiple objectives MDP approach computes the optimal QoS coefficients and propose a data-driven decision for the best services workflow on a real world dataset.

Index Terms—Internet of Things, Services Composition, Reinforcement Learning, Quality of Services, Concrete Services, Abstract Services.

1 Introduction

THE open standard group Service-Oriented Architecture [?] defines a service as an autonomous and platform independent computational entity that can be described, published, discovered and dynamically composed for creating an added value services. A service can be a service provider, a service broker or a service consumer. According to the SOA-based vision, a service has the following main properties [1] : i) it represents a business activity with specified outcome, ii) it may consist of other services types, iii) it must be black box for users and iv) it is self-contained. Among different services types, there exist several services that utilize web services standards including Web Services Description Language (WSDL) [CCM+01]HTTP [Pau14], Microsoft WCF [CCCG10], Apache hrift [Fac07] and SORCER [PTDL07].

Internet Of Things services composition is an efficient computing paradigm that aggregates individual services to create more powerful services with new functionalities, called composite service, that none of the services could provide individually [PvdH07]. In a IoT services composition process, two types of services can be distinguished: concrete service and abstract service. The term concrete service refers to an executed service, whereas an abstract service (called also a class of services) describes the functionality of a service in an abstract manner. At a given run time and for a given user, each abstract service can be achieved using several concrete services with the same functionality but possibly with different Quality of Service (QoS) levels. The IoT services composition problem arises when dealing with complex user requirements. This defines as kind of abstract services composition plan that says: *"what are the concrete services that should be selected for each abstract service and what is the best abstract services permutation in the user's task in order to satisfy the user's requirements in terms of the QoS ?"*. Therefore, the main SOA's goal concerns how to compose an application including problems related to distribution, deployment and separately maintained services. The services use meta-data which describe services functional and non-functional properties. An important and challenging research area is then how to propose an appropriate composite service in dynamic and unpredictable environments considering QoS criteria [MZ15], [BMBK+09], [SL13].

From the operational point of view, the main problem concerns the composition process of existing services to achieve more functionalities that none of the services could provide individually. This composition process is close to applications composition in mathematics and concerns concrete services. This composition is always associative and commutative [GMM15]. Otherwise, specific constraints must be given to limit the impact of these properties. In the general case, the composition complexity is the number of permutations with equality (subsets of services are used as a single element of the arrangement). In the concrete service composition process, the general constraint is that each concrete service appears one and only once.

In this paper, we are presenting some elements on the global complexity of this composition problem. This work main objective is to select the optimal permutation from all possible ones. It means, we need to define an objective function that can be optimized with any optimization method. The quality of service parameters are decisive in the success or failure of the service composition process. These parameters such as throughput and services response time are combined in a multi-objective function to be optimized later.

There are various services composition approaches proposed in the literature, including Integer Linear Programming-based approaches [ZBN+04], [ZBD+03], [CCGP07], graph-based approaches [YZL07], [LN15], [dSMZ14], [RMPLM16], constraints decomposition-based approaches [ARN12], Pareto optimality-based approaches

1. https://en.wikipedia.org/wiki/Service-oriented_architecture

[KAC+16], [YB13], [WZY+17], machine learning based approaches [MZ15], [DWH+16], [DHH+16], [RSV11], [MGI15] and recommender system-based approaches [MP05], [Liu05], [CZYL14]. There are some papers that study a huge part of the related works for solving this problem [KAC+16], [ZCDH16].

In this paper, we propose a reinforcement learning approach to select an optimal composition services according to QoS reward function without knowing the user preferred weights on the QoS attributes. A minimum number of queries are needed in required situations to select the best service arrangement. The main contribution of this work concerns minimising the number of questions that system proposes to the user in order to optimise the QoS value using Markov Decision Process (MDP). Even if there exists some works that handle this problem using MDP [MZ15], to the best of our knowledge, there exists no works that solve this problem without knowing the users preferences on the QoS attributes at the beginning. We have performed a large number of experiments on a real dataset [ZZL14] and the interesting results are given in this paper.

The next section summarises the related work on QoS-aware services composition. Section 3 details technical description of the services composition. Section 4 formalizes the services composition problem as an MDP. The proposed interactive reinforcement learning algorithms to deal with services composition problem is detailed in Section 5 and section 6 summarizes experimental results. Section 7 gives the conclusion and some perspectives.

2 Related Works

Most of services composition studies in the literature are based on knowledge of user preferences with respect to QoS criteria [?], [?], [?], [?]. Sometimes acquiring user preferences is not easy in practice because users can not answer some questions properly during the composition execution [SSSS10]. Thus, several proposed approaches solve the services composition problem in interaction with users and without requiring the knowledge of their preferences in advance, such as machine learning-based approaches. Some approaches belonging to this class are summarised hereafter.

In [MP05], a recommender system is proposed in order to help the user to select optimal services among a list of services with similar functional properties. In order to solve this problem, they learn the user's ranking on QoS attributes and they recommend the best web services w.r.t the user's ratings to the user.

An approach using Dijkstra algorithm is proposed to find the best service composition by taking into account the user's QoS requirements [LLCY09]. This approach is adaptive in dynamic environments with different QoS values w.r.t their execution in various time steps. The proposed approach consists of two steps: 1) get the whole possible execution path from a directed acyclic graph for different execution time and 2) find the best service composition according to the user's requirements. The Latter presents some thresholds on QoS values attributes that are known before starting the service composition process.

A services selection algorithm taking into account qualitative and quantitative QoS attributes is proposed in [WMY+17]. The qualitative attributes considered in this approach are provider, location and platform, whereas the quantitative attributes are response time, throughput, reliability and availability. The services selection problem is solved using two approaches. The first one combines qualitative and quantitative QoS attributes in a global optimization function. The second approach uses a genetic algorithm.

A Location-aware Web service Recommendation (LoRec) system is proposed to help users select services having the optimal QoS [CZYL14]. This system collects first users' QoS observation that are related to the past usage experience of different Web services. The users are then partitioned, according to their locations and QoS observations, in order to make personalized service recommendations for a given user. This system significantly improves the recommendation accuracy and reduces the time complexity.

Several services composition approaches use the *Reinforcement Learning* technique. The services composition problem in a dynamic environment is modelled as a Markov Decision Process and solved using Q-learning method [WZZ+10]. In this approach, the QoS attributes values are learned through executing the services while the optimal workflow of elementary service invocation actions is updated according to the change occurred in the environment. The authors assume that if the user's preferences with respect to QoS attributes are given as $\bar{W} = (w_1, \cdots, w_d)$, the reward value of each executed web service ws_i is defined as follows:

$$R(ws_i) = \sum w_i \times \frac{qos_i^j - qos_i^{\min}}{qos_i^{\max} - qos_i^{\min}} \tag{1}$$

where qos_i^j is the current value of the i-th QoS attribute of the service ws_j. Here, $qos_i^{\max}$ and $qos_i^{\min}$ represent, respectively, the maximum and minimum values of the i-th QoS attribute for all web services.

Two reinforcement learning-based QoS-aware services composition approaches are proposed in [MZ15]. The first one consists of a single policy multi-objective composition approach where each QoS-objective is considered as a separate learning agent to find the compositions having the highest QoS with unknown user preferences. The second approach is a multiple policy multi-objective composition that uses Q-learning method to determine a set of Pareto optimal compositions having the same QoS, to satisfy multiple QoS-objectives by taking into account different user preferences.

To deal with low efficiency of large-scale services composition based on traditional reinforcement learning, a service composition approach based on automatic Hierarchical Reinforcement Learning (HRL) is proposed in [WHY16]. In this approach, the services composition is modeled as a Semi-Markov Decision Process where the time steps of each web services is taken into consideration. An automatic task decomposition and MAXQ HRL are then used to find a composite service with good efficiency.

In [LJF+15], the services composition problem in highly-dynamic environments is modeled as an uncertainly planning problem using a Partially Observable Markov Decision Process. A Time-based Reinforcement Learning approach is then proposed to solve the planing problem and find the composite service satisfying the user's requirements. The proposed approach does not require knowledge of complete information about services. It uses historical information and estimates the success probability of a services composition using results and QoS of services.

A QoS-based services selection approach coupled with a learning mechanism is proposed to ensure a flexible and failure-tolerant dynamic services composition [YTA+09]. The proposed approach uses a Markov Decision Processes and a Bayesian Learning mechanism to deal with the uncertain nature of concrete service invocation due to the changes that may occur in the ubiquitous environment.

To deal with the uncertainty of QoS values and non deterministic services behavior, the Constraint-Satisfied Service Composition problem is formulated as a Markov Decision Process, called CSSC-MDP, and solved using Q-learning algorithm [RWX17]. The CSSC-MDP approach selects, for each service in the composition, the optimal candidate service based on the constraints which need to be satisfied by the following services. The selection strategy used in CSSC-MDP approach aims at maximizing the expected cumulative reward which is on behalf of the satisfaction degree of the user's QoS constraints.

Some of the services composition approaches are based on other machine Learning technique such as the *clustering method*. For instance, a services selection approach is proposed in the context of QoS-aware services composition in ubiquitous environments [MGI15]. In this approach, the services selection with global QoS constraints is formulated as a set-based optimization problem [ZTB10]. The k-Means clustering method is then used to find the composite service maximizing the QoS value and satisfying the global QoS constraints. The proposed approach is devised in both centralized and distributed fashions, which makes it suitable for both infrastructure-enabled and infrastructure-less ubiquitous environments. A Parallel Clustered Particle Swarm Optimization (PCPSO) algorithm for QoS-aware service composition is proposed in [HMM+16]. To handle a big-data and high number of services in a mobile environment, the proposed algorithm runs in parallel using MapReduce [DG08] to find the optimum composition in a reduced time computation. The PCPSO algorithm includes two phases. In the first phase, using k-means method, several groups of candidate services are selected from the available huge number of services. In the second phase, for each services group resulting from the first phase, a single service is selected using PSO method to obtain the optimum composite service in terms of QoS.

3 Problem formulation

For the sake of simplicity, the problem of service composition that meets an end-to-end user requirements can be defined as follows. Given a set of n services $S = \{S_1, \ldots, S_n\}$ where each service S_i can be implemented by n_i concrete services $\{S_{i1}, \ldots, S_{in_i}\}$. Each concrete service S_{ij} may be executed by a set of actors $\{a_1, \ldots, a_k\}$. An actor can be an end user or any other entity for which the service is rendered. We denote an instance S_{ij} executed by the actor a_k, by S_{ij}^k. Furthermore, in several situations, the same service instance can be executed more than ones over the time by the same actor. The well used time reference to deal with this fact is time step. In the rest of this paper, the concrete service S_{ij} executed by the actor a_k at time t is denoted by $S_{ij}^k(t)$ and is called *execution*. The following examples shows a real case data base for the service composition problem.

Example 1. Table 1 is an extracted line from the real dataset [ZZL14], [ZCDH16] used in our experiments. According to the aforementioned notations, this line should be denoted as $S_{19994,3104}^{97}(5)$. It represents for the abstract service 19994, the response time and throughput values of the concrete service 3104 executed by user 97 at time step 5.

TABLE 1
Extracted line from the real dataset [ZZL14], [ZCDH16] used in our experiments.

User	time	service	con. service	time resp.	throughput
97	5	19994	3104	0.238	0.773

Each service performs functions that serve the actors. To evaluate the quality of a service at the application level, a set of criteria are used such as response time, throughput, reliability, availability, price [XFZ09], [ZBN+04].

Let $Q = \{q_1, \ldots, q_m\}$ be the set of all possible QoS criteria. The criteria can be applied at all levels including the abstract services, concrete services or for different users in various time steps. Without loss of generality, we limit the criteria to the case of concrete services level. In this case, $q_l(S_{ij})$ denotes the quality value of the q_l criteria for the S_{ij} service. We suppose that all criteria value are normalized.

In SOA architectures, the orchestration process composes the existing services in order to create a new service having central control over the whole process [2]. In our context, the main part of the orchestration process is the abstract service composition. This composition process is guided by one main constraint among abstract services: when an abstract service requires some input results coming from other services, those abstract services must be executed before. When all constraints are given, a precedence graph is built (see Figure 2). A path including each abstract service, defines a possible orchestration.

Example 2. Let S_1, S_2, S_3 and S_4 be four abstract services such that services S_2 and S_3 need outputs of the service S_1 and the service S_4 is executed after S_2. The abstract services precedence graph is given in figure 2. Then, the possible orchestrations with respect to this graph are: $(S_1, \{S_2, S_3\}, S_4)$, (S_1, S_2, S_3, S_4), (S_1, S_2, S_4, S_3), etc.

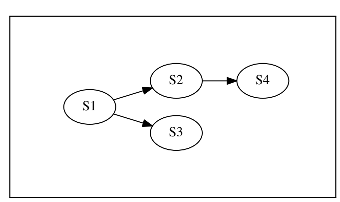

Fig. 1. An Example of abstract services precedence graph.

2. www.soa-in-practice.com/soa-glossary.html

Each abstract service may be realized by several concrete services. When abstract service orchestration is defined, we need to choose an appropriate concrete service that performs each corresponding service. Services composition can be seen as a workflow where activities and tasks can be carried out by users and machines in an IoT environment. In this paper, the possible concrete services organization in order to realise an end-to-end user service according to a given abstract service orchestration is denoted by concrete workflow. Figure 3 gives an example of concrete workflow for orchestration given in figure 2.

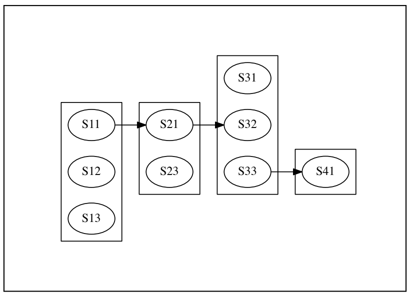

Fig. 2. The concrete workflow $(S_{11}, S_{21}, S_{32}, S_{41})$ is a possible organization to realize the orchestration given in figure 2.

Let f be a global objective function that optimizes the end-to-end Q criteria of a service by taking into account some fixed constraints. Let us consider $\psi(S)$ be the M possible abstract service orchestrations in which we denote the k-th possible orchestration as ψ_k (for $k \in \{1..M\}$). Thus, F_{ψ_k} indicates the evaluation function of all possible concrete service concatenation of ψ_k.

To obtain F_{ψ_k} (For each $k \in \{1..M\}$), it is necessary to select the best concrete service concatenation according to the Q criteria among the concrete services in the given ψ_k orchestration. ψ_k is defined according to the predefined optimisation function $\varphi_{ki,(k\in\{1,\dots,M\},i\in\{1,\dots,n_k\}}$ on the Q criteria. In our context the ψ_k function will be assumed as linear combination of Q attributes (see example **??**)

Without loss of generality, the service selection problem can be defined as follow: find the evaluation function f as the best value of F_{ψ_k}

Example 3. Let us consider a problem with three abstract services $\{S_1, S_2, S_3\}$. This problem has a single possible orchestration $\psi(S_i)_{I\in\{1..3\}} = \psi_1$, such that ψ_1 represents a serial (S_1, S_2, S_3) orchestration (we execute first concrete services of S_1, then S_2 and end with S_3). Let us consider a problem with two evaluation criteria $Q = \{q_1, q_2\}$, for instance, throughput and service time response. An the concrete services evaluation function (to be optimized later) is defined as a linear combination between these criteria: $\varphi(q_1, q_2) = w_{i1}.q_1 + w_{i2}.q_2$ for each abstract service S_i.
In this case, the problem can be summarized as finding an appropriate weighted parameters w_{ij} by solving the following maximization problem:

$$\max\ (\sum_{i=1}^{i=3} w_{i1}.q_1 + w_{i2}.q_2)$$

As explained in the related work section, several studies are done using various approaches. However, some additional constraints make this problem more difficult. The main constraint concerns the fact that no system user preferences are given on the evaluation function components. In this paper, we propose an approach based on discrete time vector valued MDP to estimate the QoS criteria weights respecting the users' preferences for any abstract service orchestration.

3.1 The global problem complexity

Without loss of generality, let us consider a service composition problem with n abstract services $S_1, \dots, S_n$, such that each abstract service S_i can be concretized by exactly m possible concrete services and there is no explicit constraint between concrete services. The purpose of this section is to enumerate the number of end-to-end concrete service combination in order to evaluate the quality of service of each combination and then to select the best one according to given user or computed preferences.

To the best of our knowledge, there is no existing work dealing with this enumeration problem to evaluate service composition complexity. The main idea of this enumeration is first to evaluate the worst complexity case and then to find a best way to reduce the quality of service evaluation time cost by performing evaluations by equivalence groups for example. We will present some typical cases for which we have an exact enumeration and we will give a lower bound for the general case.

3.1.1 Complexity when there exist a total order between abstract services

Let us consider $S_1, \dots, S_n$ this order. Since that each S_i can be realized by m possible concrete services, the possible number of realizations for each order is n^m. If there is no additional constraint between abstract services, the number of abstract services permutations will be $n!$. Therefor the number of concrete services realizations will be $n! * n^m$.

3.1.2 Complexity when all abstract services are realized in parallel

In this case there exists only one possible combination between abstract services. For this alone combination, there exists n^m possibles realizations.

3.1.3 Only two abstract services are realized in parallel

Let us consider i the number of abstract services realized in parallel and $St(n, i)$ the Stirling number of the second kind[3]. If we consider the case where there is two and only two abstract services realized in parallel for a given one total order configuration of abstract services, the number of equivalent classes of length $(n-1)$ is given by $S(n, 2) = 2^{n-1} - 1$. Therefor the number of abstract service configurations will be $(2^{n-1} - 1) * (n-1)!$. In this particular case, for each abstract service configuration there is at most $2 * (n-1)^m$ possible realizations. We conclude that the possible number of configurations is $2 * (n-1)^m * (2^{n-1} - 1) * (n-1)!$.

3. $St(n, i)$ count the number of ways to partition an n-element set into i equivalence classes. It satisfy the recursion formula: $S(n, i) = S(n-1, i-1) + i * S(n-1, i)$.

3.1.4 The complexity in the case of k equivalent classes

The number of abstract services equivalent classes if given by $S(n,k)$. The number of abstract service configurations will be $S(n,k)*k!$.

Let us denote by CSC, the number of concrete services configurations and CSC_k the number of concrete service configuration with k equivalent classes. In this case it is possible to compute upper and lower bound of possible configurations as follow: In all cases, the cardinal of all set of equivalent classes of abstract services is greater or equal to m and for the case of k equivalent classes the set greatest cardinal is $(n-k+1)$. Thereby, it's possible to give the follow bounded result: $m^k * S(n,k) * k! \leq CSC_k \leq ((n-k+1)*m)^k * S(n,k) * k!$

From all that, it is possible to bound the realization number in general case as follow: $\sum_{i=1}^{i=n}(m^i * S(n,i) * i!) \leq CSC \leq \sum_{i=1}^{i=n}(((n-i+1)*m)^i * S(n,i) * i!)$.

$$CSC \geq \sum_{i=1}^{i=n}(m^i * \sum_{j=0}^{j=i}(-1)^j \binom{i}{j}(i-j)^n)$$

$$CSC \leq \sum_{i=1}^{i=n}(((n-i+1)*m)^i * \sum_{j=0}^{j=i}(-1)^j \binom{i}{j}(i-j)^n)$$

4 SERVICES COMPOSITION AS AN MDP PROBLEM

Markov Decision Processes (MDPs) are the suitable models for sequential decision problems such as QoS decomposition problems. In this section, we describe how to use MDPs to formalize and solve the services composition problem. We are looking for an optimal QoS-selection strategy satisfying the user's requirements in terms of QoS. Since the user's preferences with respect to QoS criteria are unknown, we use a partially known MDP model and more particularly a Vector-valued MDP (VMDP) model. Before getting into details, it is required to describe some preliminary properties and definitions.

Definition 1. A **concrete service** S_{ij}, or a concrete service executed by an actor with or without time reference properties can be described by several *functional* and *non-functional* properties.

- Functional properties are generally described under the form of transaction function namely Action(S_{ij}) that takes an input data vector InputData(S_{ij}) to produce an output data vector OutputData(S_{ij})
- Non-functional properties are including a vector of QoS attributes $Q(S_{ij})$, a set of quality of experience criteria (QoE), and other aspects about the service such as energy consumption and the context of use.

Definition 2. An **abstract service** $S_i = \{S_{i1}, \cdots, S_{in_i}\}$ is a class of n_i concrete services with similar functional properties. That means they have the same input data vector and output data vector, but their non-functional properties are different.

In the rest of this section, we will explain how various classes of abstract services, each one including many concrete services can be modeled as a Vector-valued MDP.

4.1 Vector-valued Markov Decision Process

Referring to Section 3 invoking each concert service in a time step t produces different quality of services. In Example 1, invoking concrete service 3104 for abstract service 19994 at time 5 gives two different values 0.238 and 0.733 for the response time and throughput respectively: $Q(S_{ij}) = (\mathrm{rt}(S_{ij}), \mathrm{tp}(S_{ij}))$.

Definition 3. Formally, a *Discrete-time Markov Decision Process (Discrete-time MDP)* [AHSR09] is defined by a tuple $(T, S, A, P_t(.|s,a), r_t)$ where:

- $T = 0, \cdots, N$ are the decision time steps at which the decisions are made[4].
- States: S is a finite set of states
- Actions: $A(s)$ is a finite set of actions that agent can select in state s.
- State Transition Probability Distribution: $P_t(s'|s,a)$ encodes the probability of going to state s' when the agent is in state s and chooses action a.
- Reward Function: $r_t : S \times A \longrightarrow \mathbb{R}$ where $r_t(s,a)$ quantifies the utility of performing action a in state s at t time step.

A *Decision rule* d_t is a function depending on time t that defines what action $d_t(s) \in A(s)$ at time t the agent should select. By assuming N number of time steps, we define *policy* $\pi = (d_1, \cdots, d_{N-1})$ as a sequence of $N-1$ decision rules. The policy is stationary if the decision rule for all time steps are the same i.e. : $\forall\ t \in \{1, \cdots, T\}\ d_t = d$.

A solution for an MDP is a policy $\pi : S \longrightarrow A$ that associates an action to each state. Normally, policies are evaluated by a value function $v^\pi : S \longrightarrow \mathbb{R}$. The value function is computed recursively using several recursive functions:

$$v_N^\pi(s) = r_N(s, \pi(s)) \ \ \forall s \in S_T \tag{2}$$

where S_T is the set of terminal states as a subset of all states. For the rest of time steps $t < T$, the value function is defined as:

$$v_t^\pi(s) = r_t(s,\pi(s)) + \gamma \sum_{s' \in S} P_t(s'|s,\pi(s)) v_{t+1}^\pi(s') \tag{3}$$

where γ is a discount factor and $0 < \gamma \leq 1$. Therefore, the preference relation among policies is defined as below:

$$\pi \succeq \pi' \Leftrightarrow \forall s \in S\ v_0^\pi(s) \geq v_0^{\pi'}(s) \tag{4}$$

The **MDP solution** is an *optimal policy* which is the highest policy with respect to the other policies and w.r.t $\succeq$, i.e. $\pi*$ is an optimal policy if $\forall\ \pi,\ \pi^* \succeq \pi$.

To find such a policy/workflow, we can use a dynamic programming, namely *Bellman Equation*.

$$v_N^* = r_N(s)\ \forall s \in S_T \tag{5}$$

and for all $t = 1, \cdots, N-1$ and $s \in S$, the value of the optimal policy is computed as:

$$v_t^*(s) = \max_{a \in A(s)} \left\{ r_t(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_{t+1}^*(s') \right\} \tag{6}$$

4. time steps can be days, hours, minutes or any time interval

For the sake of simplicity, we define Q-value function on state s and action a at time step t as:

$$Q_t(s,a) = r_t(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v^*_{t+1}(s') \tag{7}$$

In the other hand, the optimal policy is the policy that selects action a^* at stage t from the following:

$$a^*_t \in \text{argmax}_{a \in A(s)} \{Q_t(s,a)\} \quad \text{for } t = 1 \cdots N-1 \tag{8}$$

Sometimes, selecting an action in a given state and a given time step may have several effects instead of only one value. Therefore, by extending the discrete-time MDP to a discrete-time vector-valued MDP, we have:

Definition 4. [ACL16] A **discrete-time Vector-valued MDP (discrete-time VMDP)** is defined by a tuple $(T, S, A, P_t(.|s,a), \bar{r}_t)$ where the vector-valued reward function $\bar{r}$ is defined on $S \times A$ and $\bar{r}(s,a) = (r_{1t}(s,a), \cdots, r_{dt}(s,a)) \in \mathbb{R}^d$ is the vector valued reward defined by $\bar{r}$ in state s and action a.

Notice that the VMDP is another form of Multi objective MDP. That means, d is the number of objectives in the environment while each element i in reward vector $\bar{r}(s,a)$ indicates cost of the i-th objective in the model by selecting action a in state s.

4.2 Service composition as a discrete-time VMDP

By modeling the service composition as a discrete-time VMDP, we can compute the best selected concrete service composition satisfying user requirements in terms of QoS. We assume that, we are allowed in this work to communicate with users in order to get information about their preferences with respect to QoS criteria. We use discrete-time VMDP modeling to select the optimal concrete service for each abstract service w.r.t time stage t. For the sake of simplicity, this model is noted as the discrete-time VMDP-Services Composition (discrete-time VMDP-SC) which is introduced hereafter (see [WZZ+10], [MZ15]).

Definition 5. A **VMDP-Service Composition (VMDP-SC** is a tuple $(T, AS, CS, P_t(.|as, cs), \bar{r}_t, AS_T)$, where

- $T = 1 \cdots N$ is a total number of time stages.
- AS is a finite set of abstract services.
- CS is a set of all concrete services, where $CS(S_i)$ indicates a set of available concrete services for the abstract service $S_i \in AS$.
- $P_t(S_j|S_i, S_{ik})$ is the probability of invoking the concrete service S_{ik} for abstract activity S_i and resulting in the abstract activity S_j.
- $\overline{Q}_t : AS \times CS \longrightarrow \mathbb{R}^d$ is a reward function. The $\overline{Q}(S_i, S_{ik})$ reward is the generated Q vector value after invoking S_{ik} in S_i at time step t. Given that d represents the number of QoS criteria, we obtain $\overline{Q}_t(S_i, S_{ik}) = (q_{1t}(S_i, S_{ik}), \cdots, q_{dt}(S_i, S_{ik}))$.
- AS_T is the set of terminal services. The execution of the service composition terminates in one of these states.

In fact, the solution for QoS-aware service composition is the optimal policy for VMDP-SC model.

Definition 6. A **policy service composition** $\pi : AS \longrightarrow CS$ is a function that defines which concrete service should be invoked for each abstract service in order to give the best trade-offs among multiple QoS criteria.

This policy is known as a workflow or composition in the IoT services composition literature. Since reward values in MDP-SC are the Q vectors for each concrete service, each policy should be evaluated with the following vector function (see Equation 3):

$$\bar{v}^\pi_t(S_i) = \overline{Q}_t(S_i, \pi(S_i)) + \gamma \sum_{S_j \in S} P(S_j|\pi(S_i), S_i) \bar{v}^\pi_{t+1}(S_j) \tag{9}$$

Thus, comparing two composition/policies boils down to comparing two vectors. The optimal compositions satisfying various users with different preferences on the QoS attributes can be different. Thus, we need a model that presents the user preferences over quality of services attributes.

For this reason, the Simple Additive Weighting (SAW) technique [QTDC10] is used to aggregate the QoS attributes values of services into a single utility value by considering user's preferences expressed as weights. The services selection is then transformed into a single objective optimization problem to find the candidate services providing the best utility value. In fact, if any user gives a weight to each attribute, the dependency between the users' weights and quality of service attributes is defined as below:

$$Q_t(S_i, S_{ij}) = \sum_{k=1}^{d} \bar{w}_k q_{kt} = \bar{w} \cdot \overline{Q}(S_i, S_{ij}) \qquad \forall t = 1, \cdots, N \tag{10}$$

where $\bar{w} = (w_0, \cdots, w_d)$ is a weight vector, indicating the user preferences on the QoS attributes such that

$$\sum_{i=1}^{d} w_i = 1$$

If user's preferences on QoS attributes are given, the optimal composition can be therefore computed easily using SAW technique. However, determining appropriate weights for QoS attributes needs knowledge of user preferences, which is often not obvious to obtain in practice. Even if user preferences have been obtained, setting accurately these weights remains a problem. For instance, it is hard to decide the weight of response time as 0.2 or 0.21, which appears no big difference yet it can affect the result of the QoS optimal composition [CHLH15]. Accordingly, we assume that $\bar{w}$ is unknown and try to find the best composition/policy by querying users when it is necessary.

To compare workflow vector values with each other, we consider first, the unknown weight vectors are confined in a $d-1$ dimensional polytope W such that:

$$W = \{(w_1, w_2, \cdots, w_d) \mid \sum_{i=2}^{d} w_i \leq 1 \text{ and } w_1 = 1 - \sum_{i=2}^{d} w_i\} \tag{11}$$

To compare Q vector values with each other, we can use three different comparison methods. Assume $\bar{v}^a = (a_1, \cdots, a_d)$ and $\bar{v}^b = (b_1, \cdots, b_d)$ are two d-dimensional vectors representing expectation of sum of QoS values for two workflows a and b.

- the most natural comparison method is *pareto comparison* that defines:

$$\bar{v}^a \succeq_P \bar{v}^b \Leftrightarrow \forall\ i\ a_i \geq b_i \tag{12}$$

- *Kdominance comparison* defines $\bar{v}^a$ is more preferred than $\bar{v}^b$ if, it is better for any $\bar{w}$ in polytope W:

$$\bar{v}^a \succeq_K \bar{v}^b \Leftrightarrow \forall\ \bar{W} \in W\ \bar{W} \cdot \bar{v}^a \geq \bar{W} \cdot \bar{v}^b \tag{13}$$

- query this comparison to the user, i.e. $\bar{v}^a \succeq_q \bar{v}^b$.

We remind that, the Kdominance comparison is a linear programming problem. In other words, $\bar{v}^a \succeq_K \bar{v}^b$ is satisfied if there is a non-negative solution to the following LP:

$$\begin{cases} \min \bar{W} \cdot (\bar{v}^a - \bar{v}^b) \\ \text{subject to } \ \bar{W} \in W \end{cases} \tag{14}$$

If there is no non-negative solution for two comparisons $\bar{v}^a \succeq_K \bar{v}^b$ and $\bar{v}^b \succeq_K \bar{v}^a$, these two vectors are not comparable using the Kdominance.

In the rest of this paper, we will explain how to find the optimal composition/policy that gives the best trade-off among multiple QoS criteria, satisfying the user requirements in terms of QoS by querying him very few times.

5 Interactive Reinforcement Learning Algorithms for the Service Compositions

We propose an algorithm namely *Interactive Value Iteration for Service Composition (IVI-SC)*. Previously, we explained how model the services composition problem as a discrete-time MDP. In this section, we describe how to find the solution using the existed solutions for MDPs. Some researchers use interactive value iteration methods to find the optimal policy respecting the user of system preferences [WZ13], [ACL16]. In this paper, we modified the interactive value iteration on a finite-horizontal MDP to find the best service composition satisfying users' requirements in terms of QoS. We assume that an MDP model of services (VMDP-SC) with finite discrete-time is given. The services can be invoked in $T+1$ number of discrete time steps: $\{0, \cdots, T-1\} \cup \{T\}$ where T is a final empty time stage. Since the MDP-SC objective is finding the policy that maximizes a measure of long-run expected Q vectors, we propose a backward induction method to solve the Bellman equation given in equation 6 and finds the optimal actions given in equation 8 to obtain the optimal policy/work-flow. Our solution is introduced in Algorithm 1.

In the iterative algorithm 1, first we assign a zero vector to the set of states (abstract services) at time step T. For each abstract service $S_i(t)$ (in time t) given in the MDP-CS, the algorithm selects the best concert service among the all available ones. These actions (concrete services) are dependent on various time steps, for instance the possible actions of the $S_i(t)$ service in times step t can be different from the possible actions for time step $t+1$. In the finite horizon time (our case), the iteration continues until either this difference becomes small enough or the horizon time steps finish.

Since the quality of services are the d dimensional vectors, solving equation 6 and finding the maximum among the vectors is not obvious. For this reason, we remind three comparison methods (presented in equations 12, 13 and 14) and utilize the **Best** function (given in Algorithm 5). This function receives two d dimensional vectors with the W polytope confining the user weight preferences on the

```
Data: VMDP-SC(T, AS, CS(), P_t, r̄_t), a W polytope of
      user weights on objectives
Result: The optimal service selection policy for the
        given user.
t ⟵ T
π_best ⟵ choose random policy
v̄_T(s_T) ⟵ (0, ··· , 0)^5 ∀s_T at time T
𝒦 ⟵ set of constraints on W
while t ≥ 0 do
    t ⟵ t − 1
    for S_i(t) do
        best ⟵ (0, ··· , 0)
        for each S_ij(t) ∈ CS(S_i(t)) do
            v̄_t(S_i(t)) ⟵
            QoS(S_i(t), S_ij(t)) + Σ_{S_m(t+1)} P_t(S_m(t +
            1)|S_i(t), S_ij(t)) v̄_{t+1}(S_m(t + 1))
            ( best , 𝒦) ⟵ getBest(best, v̄_t, 𝒦)
            v̄_t(S_i(t)) ⟵ best
            if best = v̄_t(S_i(t)) then
                π_best(S_i(t))⟵S_ij(t)
            end
        end
    end
end
return π_best
```

Algorithm 1: Interactive Value Iteration for Service Composition: How to select the best composite for each abstract service respecting user preferences on QoS attributes

```
Data: finds the more preferred vector between two
      vectors v̄ and v̄' w.r.t 𝒦
Result:
if paretodominates(v̄, v̄') then
    return (v̄, 𝒦)
end
if paretodominates(v̄', v̄) then
    return (v̄', 𝒦)
end
if Kdominates(v̄, v̄', 𝒦) then
    return (v̄, 𝒦)
end
if Kdominates(v̄', v̄, 𝒦) then
    return (v̄', 𝒦)
end
(v̄_best, 𝒦) ⟵ query(v̄, v̄', 𝒦)
return (v̄_best, 𝒦)
```

Algorithm 2: Best: this algorithm finds the most preferred vector between two given vectors.

Data: $\bar{v}, \bar{v}', \mathcal{K}$
Result: it queries the comparison between $\bar{v}$ and $\bar{v}'$, to the user and modifies $\mathcal{K}$ according to her response.
Build query q for the comparison between $\bar{v}$ and $\bar{v}'$
if *if the user prefers* $\bar{v}$ *to* $\bar{v}'$ **then**
| **return** $(\bar{v}, \{(\bar{v} - \bar{v}') \cdot \bar{W} \geq 0\})$
end
else
| **return** $(\bar{v}', \{(\bar{v}' - \bar{v}) \cdot \bar{W} \geq 0\})$
end

Algorithm 3: query: queries the user about her preferences on existed quality of services.

quality of services. If the pareto comparison can not find the greater vector, we will test the Kdominance comparison for finding the most preferred vectors. Otherwise the query function should be called (given in Algorithm 5). The user's response to the comparison between the two given vectors, adds a new constraint to the W polytope.

Algorithm 1 finally finds the optimal policy/work-flow or service composition for the given system MDP-SC and returns back the optimal policy π_{best}. Notice that the condition $best = \bar{v}_t(S_i(t))$ in Algorithm 1 checks if the best selected concrete service for $S_i(t)$ has been changed regarding the previous iteration. If it is, the optimal concrete service should be replaced by the concrete service $S_{ij}(t)$ which generates a better vector value for $S_i(t)$.

The complexity of the proposed algorithm as an exact algorithm is polynomial w.r.t three parameters : 1) the number of abstract services forming the composition, the number of candidate services per each abstract services, and the number of QoS criteria. Assume $|AS|$ is the set of all abstract services and $M = \max_{i,t} CS(S_i(t))$ is the maximum number of abstract services in each time step t and each abstract service S_i. For computing the best QoS vector in each inner iteration, the *Best* algorithm tests the pareto dominance and k-dominance comparison twice in the worst case which are polynomial w.r.t d. d is the number of attributes for quality of services and any k-dominance (LP) can be solve in polynomial time. Then, the algorithm has $O(T.|AS|.M.d)$ where T is the number of time stages.

6 PERFORMANCE EVALUATION

[?] introduces a method for finding the list of all non-dominated service composition regardless of user weight preferences $\bar{W}$ on QoS attributes. That means, each exact computed service composition is included in this set. For this reason, IVI-SC proposes an algorithm for computing the exact service composition for each system user. In the following we show how the experimental results work.

We evaluate our methods on a public available data-set containing two parameters for quality of services: throughput and response time. These are the records between 339 users and 5825 web services distributed worldwide [ZZL14], [ZCDH16]. The data-set also includes some information about user features and service features such as countries, autonomous systems, IP dresses, latitude and longitude. In the studied data base, 142 users execute various web services

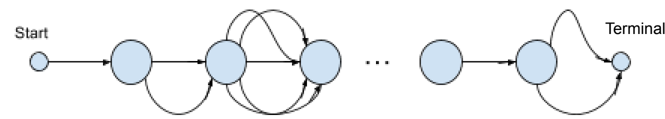

Fig. 3. A sequential form of abstract service connection

in different time slices. Practically, the information are given only for 64 time steps. In this section, we first explain how to model the dataset as a VMDP-SC and then we will examine our algorithm on the data-set.

6.1 Data-Set as a VMDP

The main issue in implementing IVI-SC (Algorithm 1) on any database is that how to model the given data-set as a vector-valued MDP. In the supported dataset [ZZL14], [ZCDH16], there are several text files including wslist.txt, userlist.txt, rtdata.txt and tpdata.txt. The wslist.txt represents some information on various web services and their related abstract services. It is our source file for extracting a list of web services and their related abstract services; if a related abstract service exists for the selected web service. The userlist.txt includes some information about users of different web services. The two other files tpdata.txt and rtdata.txt include the throughout and response time values respectively on various web-services executed by 142 users. That means any invoked web service by a user has two parameters for measuring the service quality: throughout and response time.

The studied database [ZZL14], [ZCDH16] is generated in real by observing various users utilizing enormous number of web services. We notice that all provided data inside database are not useful. After extracting all web services and their related abstract services from wslist.tx file, and getting the web services qualities from two files tp.txt and rt.txt, we have a VMDP-SC with the following parameters (refer to 5) :

- number of episodes: $N = 64$
- 744 number of abstract services
- 3551 total number of concrete services (in our case web services)
- The transition function and terminal states depend on the proposed model or relation types among the abstract services.
- and the $\overline{Q}_t$ function is built based on the extracted data on web services and their two qualities (response time and throughout).

To demonstrate the efficiency of our approach in calculating the optimal workflow, we study our method on a common model in state of the art: the sequential model.

For the sequential model (see Fig 6.1) the start sate is an empty state and connected to the first selected abstract service in the model. On the other hand, the terminal state is an empty state that indicates the MDP is a finite horizon one. In this model, the sequential order on abstract services can be defined in any order. In our model, the order has been selected randomly once to fix the MDP model. For any time step $t \in \{0 \cdots 63\}$, the transition probability $P_t(S_j(t+1)|S_i(t), S_{ik}(t))$ is 1, if web service S_{ik} is invocable for a given abstract service $S_i(t)$ according to our database and

user	weight vector
$\bar{W}_0$	[0.319797998295 , 0.680202001705]
$\bar{W}_1$	[0.8573741847324399, 0.14262581526756013]
$\bar{W}_2$	[0.1696287781131175 , 0.8303712218868825]
$\bar{W}_3$	[0.6451844883834318 , 0.3548155116165682]
$\bar{W}_4$	[0.47245438345 , 0.52754561655]

TABLE 2
The weight vectors for 5 system users with various preferences on the attributes (throughput, response time).

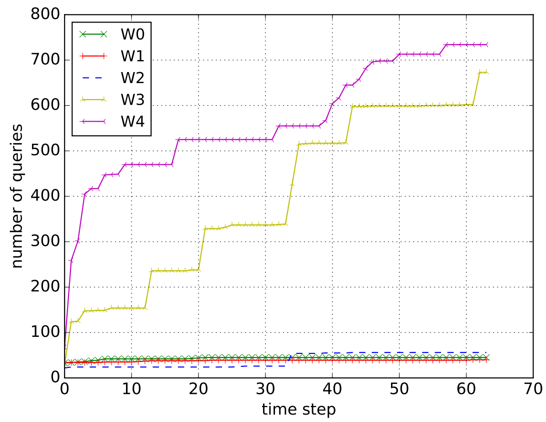

Fig. 4. This figure shows the number of queries proposed to the user during each time step. The weight preferences are based on table 2.

abstract service $S_j(t+1)$ is the next demanded service in our selected sequential MDP model, otherwise $P_t(S_j(t+1)|S_i(t), S_{ik}(t)) = 0$.

6.2 Model Whole Dataset as VMDP

To evaluate algorithm 1 on our supported dataset, we consider the complete dataset. That means, we keep all quality services executed by all 142 users. Modelling the huge size data base as a VMDP-SC and implementing the IVI-SC (1) on the model is challenging.

In order to evaluate our algorithms performance, we analysed the results for 5 different users systems with various preferences on the service qualities: response time and throughput. Our tested user weights vector ($\bar{W}$) on service qualities is given in table 2. Notice that the weight preferences on quality of services are "unknown" to our algorithms. Our proposed algorithm 1 calculates the optimal workflow while learning the users weight preferences on the attributes. We mention the weight vectors for the simplicity of demonstrating the experimental results and to compare them with the learned weights from Algorithm 1.

Figure 4 shows how the interactive value iteration algorithm communicates with users during 64 time steps. Since the user weight preferences are unknown to the algorithm, it is required to query them in the required situations. According to the figure, the algorithm 1 does not ask more than 56 queries to the users with various preferences weights $\bar{W}_0$, $\bar{W}_1$ and $\bar{W}_2$ on the service qualities. On the other hand,for two weight preferences $\bar{W}_3$ and $\bar{W}_4$, the algorithm queries too many questions to the user. Regarding to the QoS values and these weights, comparing vectors using pareto dominant and K-dominant methods are not informative.

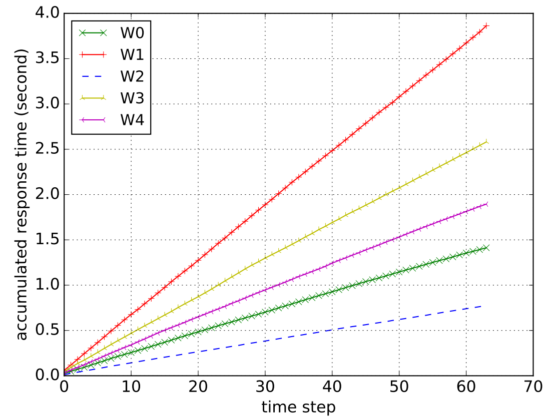

Fig. 5. This figures demonstrate how the accumulated response time increases during each time step. The weight preferences are based on table 2.

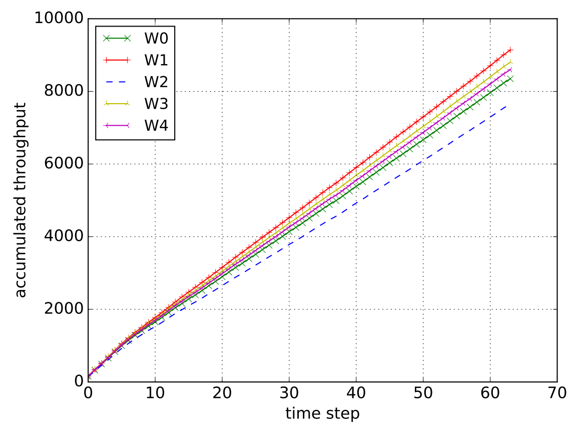

Fig. 6. this demonstrates how the accumulated throughout increases during each time step. The weight preferences are based on table 2.

Thus, finding the optimal workflow is difficult for the similar weights. Since, IVI-SC is an exact algorithm, the initial sequence of data-set presentation have an important affect on the number of required queries for finding the optimal policy. In our next research, we are interested in observing the number of required queries for finding the optimal service composition w.r.t the dataset presentation form.

On the other hand two Figures 5 and 6 show how the service qualities including response time and throughput respectively, change with respect to time step for the 5 given weight preferences on the attributes. The optimal service composition maximizes total sum of throughputs while minimizing the total sum of response times. The two figures demonstrate how throughput and response time increase linearly w.r.t time step. Figure 5 shows that the minimum accumulated response time for all weight preferences does not pass 4 seconds. In the other hand, Figure 6 maximises the accumulated throughput until around 9000.

In this paper, we are interested in the optimal workflow i.e. selecting the best concrete services for each abstract service in order to maximize the service qualities satisfying user weight preferences. If the users weight vectors are available, the exact workflow is calculable by taking the weight into account and using a classical approach on MDPs such as Value iteration method [**?**], [AGS17]. In VMDP formulation (see 5) the reward values are vectors, but knowing the user weights $\bar{W}$, transfers the quality vectors to the quality values: $QoS(S_i, S_{ij}) = \bar{W} \cdot \overline{Qos}(S_i, S_{ij})$. For this reason, the value iteration is applicable on MDPs service compositions.

Our experiments indicate that the computed optimal workflow by IVI-SC (algorithm 1) is the same as the exact computed workflow for the 3 user preferences , given in our experiments (W_0, W_1 and W_3). For the two other weight preferences W_2 and W_4, the IVI-SC and the exact approach are different in a few number of abstract services, 178 abstract services for W_2 and 38 abstract services for W_4. In general, we have 744 abstract services in our experiments. Table **??** presents the list of abstract services where our approach (IVI-SC) and the exact approach propose different concrete services for the service composition problem with W_2. And Table **??** shows the differences between these two approaches for W_4. In fact, selecting a concrete service for an abstract service with too many number of concrete services is more complex than an abstract service with a few number of ones. For instance, respecting to Table **??**, service $AS6983$ has 50 available concrete services, while $AS14280$ has only two concrete services to choose.

7 CONCLUSION AND FUTURE WORKS

In this paper, a reinforcement learning-based approach is proposed to solve the services composition problem in the context of IoT-based environments without requiring user's preferences on the QoS attributes. The services composition problem is formulated as discrete-time Vector-valued Markov Decision Process and solved using interactive reinforcement method. The experiments show how the implemented IVI-SC algorithm on a Dataset [ZZL14] of web services maximises the throughput and minimises the response time for various system users with different preferences on the attributes. And how our approach finds the optimal service composition by learning the user's preferences weights with a high accuracy.

The registered qualities of the web services in our studied dataset [ZCDH16], [ZZL14] are executed by 142 users. In this paper, we modelled the whole dataset as a sequential MDP-CS and computed the optimal service composition. In our future work, we will study our algorithm on different models according to their given orchestration, such as parallel MDP-CS and etc. We are also interested in classifying and observing the users (in our case 142 users) types w.r.t their service qualities and their execution information on the web services.

REFERENCES

[ACL16] Pegah Alizadeh, Yann Chevaleyre, and François Lévy. Advantage Based Value Iteration for Markov Decision Processes with Unknown Rewards. In *International Joint Conference on Neural Networks (IJCNN 2016)*, Vancouver, Canada, July 2016.

[AGS17] F. Ezhil Mary ARASI, S. GOVINDARAJAN, and A. SUBBARAYAN. Weighted quality of service based ranking of web services. In *Indian Journal of Science and Technology*, Jul 2017.

[AHSR09] Oguzhan Alagoz, Heather Hsu, Andrew J. Schaefer, and Mark S. Roberts. Markov decision processes: A tool for sequential decision making under uncertainty. *Medical Decision Making*, 30(4):474–483, 2009.

[ARN12] Mohammad Alrifai, Thomas Risse, and Wolfgang Nejdl. A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2):7, 2012.

[BMBK+09] Nebil Ben Mabrouk, Sandrine Beauche, Elena Kuznetsova, Nikolaos Georgantas, and Valérie Issarny. *QoS-Aware Service Composition in Dynamic Service Oriented Environments*, pages 123–142. Springer Berlin Heidelberg, 2009.

[CCCG10] Pablo Cibraro, Kurt Claeys, Fabio Cozzolino, and Johann Grabner. *Professional WCF 4: Windows Communication Foundation with .NET 4*. Wrox, June 15, 2010.

[CCGP07] Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, and Francesco Lo Presti. Flow-based service selection forweb service composition supporting multiple qos classes. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 743–750. IEEE, 2007.

[CCM+01] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, et al. Web services description language (wsdl) 1.1, 2001.

[CHLH15] Ying Chen, Jiwei Huang, Chuang Lin, and Jie Hu. A partial selection methodology for efficient qos-aware service composition. *IEEE Transactions on Services Computing*, 8(3):384–397, 2015.

[CZYL14] Xi Chen, Zibin Zheng, Qi Yu, and Michael R Lyu. Web service recommendation via exploiting location and qos information. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1913–1924, 2014.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DHH+16] ShuiGuang Deng, Longtao Huang, Daning Hu, J. Leon Zhao, and Zhaohui Wu. Mobility-enabled service selection for composite services. *IEEE Trans. Services Computing*, 9:394–407, 2016.

[dSMZ14] Alexandre Sawczuk da Silva, Hui Ma, and Mengjie Zhang. A graph-based particle swarm optimisation approach to qos-aware web service composition and selection. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 3127–3134. IEEE, 2014.

[DWH+16] Shuiguang Deng, Hongyue Wu, Daning Hu, J. Leon Zhao, undefined, undefined, undefined, and undefined. Service selection for composition with qos correlations. *IEEE Transactions on Services Computing*, 9:291–303, 2016.

[Fac07] Facebook. Apache trift tm, 2007.

[GMM15] V. Gabrel, M. Manouvrier, and C. Murat. Web services composition: Complexity and models. *Discrete Applied Mathematics*, 196(Supplement C):100 – 114, 2015.

[HMM+16] M Shamim Hossain, Mohd Moniruzzaman, Ghulam Muhammad, Ahmed Ghoneim, and Atif Alamri. Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment. *IEEE Transactions on Services Computing*, 9(5):806–817, 2016.

[KAC+16] Mohamed Essaid Khanouche, Yacine Amirat, Abdelghani Chibani, Moussa Kerkar, and Ali Yachir. Energy-centered and qos-aware services selection for internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3):1256–1269, 2016.

[Liu05] W. Liu. Trustworthy service selection and composition - reducing the entropy of service-oriented web. In *INDIN 2005. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.*, pages 104–109, Aug 2005.

[LJF+15] Yu Lei, Zhou Jiantao, Wei Fengqi, Gao Yongqiang, and Yang Bo. Web service composition based on reinforcement learning. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 731–734. IEEE, 2015.

[LLCY09] Z. Liu, T. Liu, L. Cai, and G. Yang. A novel web service selection based on dynamic quality evaluation. In *2009 International Conference on Computational Intelligence and Software Engineering*, pages 1–6, Dec 2009.

[LN15] Guisselle A García Llinás and Rakesh Nagi. Network and qos-based selection of complementary services. *IEEE Transactions on Services Computing*, 8(1):79–91, 2015.

[MGI15] Nebil Ben Mabrouk, Nikolaos Georgantas, and Valerie Issarny. Set-based bi-level optimisation for qos-aware service composition in ubiquitous environments. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 25–32. IEEE, 2015.

[MP05] U. S. Manikrao and T. V. Prabhakar. Dynamic selection of web services with recommendation system. In *International Conference on Next Generation Web Services Practices (NWeSP'05)*, pages 5 pp.–, Aug 2005.

[MZ15] A. Mostafa and M. Zhang. Multi-objective service composition in uncertain environments. *IEEE Transactions on Services Computing*, (99), 2015.

[Pau14] Cesare Pautasso. Restful web services: Principles, patterns, emerging technologies. In *Web Services Foundations*, pages 31–51. 2014.

[PTDL07] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, pages 38–45, 2007.

[PvdH07] Mike P. Papazoglou and Willem-Jan van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, Jul 2007.

[QTDC10] Lianyong Qi, Ying Tang, Wanchun Dou, and Jinjun Chen. Combining local optimization and enumeration for qos-aware web service composition. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 34–41. IEEE, 2010.

[RMPLM16] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes. An integrated semantic web service discovery and composition framework. *IEEE Transactions on Services Computing*, 9(4):537–550, July 2016.

[RSV11] R.V. Rao, V.J. Savsani, and D.P. Vakharia. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43:303 – 315, 2011.

[RWX17] Lifang Ren, Wenjian Wang, and Hang Xu. A reinforcement learning method for constraint-satisfied services composition. *IEEE Transactions on Services Computing*, 2017.

[SL13] Seheon Song and Seok-Won Lee. A goal-driven approach for adaptive service composition using planning. *Mathematical and Computer Modelling*, 58:261 – 273, 2013.

[SSSS10] Dimitrios Skoutas, Dimitris Sacharidis, Alkis Simitsis, and Timos Sellis. Ranking and clustering web services using multicriteria dominance relationships. *IEEE Transactions on Services Computing*, 3(3):163–177, 2010.

[WHY16] Hongbing Wang, Guicheng Huang, and Qi Yu. Automatic hierarchical reinforcement learning for efficient large-scale service composition. In *Web Services (ICWS), 2016 IEEE International Conference on*, pages 57–64. IEEE, 2016.

[WMY+17] Hongbing Wang, Peisheng Ma, Qi Yu, Danrong Yang, Jiajie Li, and Huanhuan Fei. Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition. *Journal of Parallel and Distributed Computing*, 100:71–84, 2017.

[WZ13] Paul Weng and Bruno Zanuttini. Interactive Value Iteration for Markov Decision Processes with Unknown Rewards. In *IJCAI '13 - Twenty-Third international joint conference on Artificial Intelligence*, pages 2415–2421, Beijing, China, August 2013. AAAI Press.

[WZY+17] Shangguang Wang, Ao Zhou, Mingzhe Yang, Lei Sun, Ching-Hsien Hsu, et al. Service composition in cyber-physical-social systems. *IEEE Transactions on Emerging Topics in Computing*, 2017.

[WZZ+10] Hongbing Wang, Xuan Zhou, Xiang Zhou, Weihong Liu, Wenya Li, and Athman Bouguettaya. *Adaptive Service Composition Based on Reinforcement Learning*, pages 92–107. Springer Berlin Heidelberg, 2010.

[XFZ09] PengCheng Xiong, YuShun Fan, and MengChu Zhou. Web service configuration under multiple quality-of-service attributes. *IEEE Transactions on Automation Science and Engineering*, 6(2):311–321, 2009.

[YB13] Qi Yu and Athman Bouguettaya. Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):776–789, 2013.

[YTA+09] Ali Yachir, Karim Tari, Yacine Amirat, Abdelghani Chibani, and Nadjib Badache. Qos based framework for ubiquitous robotic services composition. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2019–2026. IEEE, 2009.

[YZL07] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6, 2007.

[ZBD+03] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z Sheng. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421. ACM, 2003.

[ZBN+04] Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5):311–327, 2004.

[ZCDH16] Y. Zhang, G. Cui, S. Deng, and Q. He. Alliance-aware service composition based on quotient space. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 340–347, June 2016.

[ZTB10] Eckart Zitzler, Lothar Thiele, and Johannes Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79, 2010.

[ZZL14] Zibin Zheng, Yilei Zhang, and Michael R. Lyu. Investigating qos of real-world web services. *IEEE Transactions on Services Computing*, 7(1):32–39, 2014.

TABLE 3
This table demonstrates how each predicted service composition from algorithm 1 is different from the exact service composition for W_2.

Services	number of WS	IVI	Exact	Services	number of WS	IVI	Exact
AS14280	2	566	567	AS9829	2	1559	1565
AS6360	3	3723	3725	AS29650	2	1583	1586
AS4766	9	2226	2237	AS4768	2	2053	2054
AS3288	2	1622	1623	AS42695	2	2503	2504
AS20284	8	1547	1554	AS42915	3	1588	1590
AS852	7	533	4126	AS22047	2	612	613
AS14415	13	4120	4119	AS50517	3	2304	2306
AS16265	2	2018	2665	AS5050	4	3584	3586
AS13301	6	1382	1487	AS13041	24	2375	2368
AS15806	7	1566	1576	AS47720	3	1569	1571
AS11955	12	3882	3888	AS8075	11	12	4183
AS33821	2	2256	2258	AS23148	5	583	4233
AS29944	3	4223	4224	AS35041	2	2514	2533
AS20773	6	1348	2548	AS6830	2	1579	1584
AS17819	4	2325	2329	AS7050	4	4198	4201
AS27437	3	4047	4050	AS48347	3	2271	2273
AS19855	3	3652	3653	AS760	3	90	92
AS23650	22	625	853	AS8737	13	1932	1940
AS87	5	4111	4112	AS9116	2	1615	1630
AS8551	4	1620	1634	AS19875	5	547	549
AS15366	4	1244	1246	AS55481	4	44	47
AS5603	2	2345	2349	AS39418	10	959	970
AS31727	4	1582	2922	AS5786	3	2212	2214
AS9848	4	2232	2233	AS29097	3	2538	2540
AS29076	5	2275	2293	AS29951	8	3856	3860
AS4808	10	670	770	AS32577	10	3640	3638
AS17431	2	752	753	AS32475	3	3728	3729
AS2819	5	918	946	AS3786	13	2215	2225
AS32613	3	93	94	AS27030	3	4467	4469
AS11305	3	3688	4046	AS24969	2	999	1004
AS15670	3	1982	1984	AS1251	10	175	194
AS156	2	4179	4180	AS12714	2	2279	2280
AS16095	3	998	1016	AS16245	2	1000	1018
AS15290	6	536	574	AS8542	6	2081	2090
AS31815	11	4236	4414	AS34235	4	1141	1208
AS8151	3	1918	1919	AS9308	5	748	801
AS4812	11	760	862	AS5409	2	1440	1441
AS3389	4	4109	4106	AS8928	2	1202	3045
AS81	5	509	510	AS3561	25	3647	3579
AS6785	2	955	956	AS8659	4	2543	2545
AS45061	12	648	649	AS44249	10	2171	2173
AS702	16	156	1147	AS701	19	3493	3491
AS22070	2	3661	3662	AS11426	50	3207	3163
AS1128	5	1964	1965	AS2611	10	132	140
AS25518	2	1774	1775	AS14584	4	3894	3896
AS224	7	2115	2075	AS16339	2	2911	2912
AS15348	3	3731	3732	AS16237	4	1959	1960
AS31827	3	4341	4342	AS32	9	4387	4388
AS30190	2	4408	4409	AS9143	8	1974	1991
AS8220	15	160	1182	AS9318	6	2229	2243
AS4230	8	167	199	AS22489	3	210	211
AS43200	3	2087	2089	AS3307	3	2091	2120
AS3301	6	2506	2522	AS9811	3	695	697
AS34779	9	2332	2335	AS73	4	3911	3914
AS553	2	1448	1466	AS12859	7	125	2024
AS15756	3	2283	2298	AS15879	4	1961	2029
AS10929	4	4065	4066	AS15085	2	554	555
AS21844	13	4163	4161	AS12731	4	1361	1378
AS237	6	3523	816	AS195	13	4188	4027
AS1226	3	4010	4012	AS1221	2	43	79
AS8358	2	1521	1522	AS680	44	1398	1401
AS8	3	3667	3668	AS3316	4	2260	2261
AS4837	29	680	791	AS27617	3	4445	4446
AS8190	3	4067	4068	AS8904	5	2264	2266
AS43541	3	913	915	AS2519	4	1857	1859
AS42949	2	1994	1995	AS18125	5	1861	1862
AS2200	28	1120	1108	AS26228	6	2077	2080
AS19024	8	3924	3929	AS11343	3	4444	4442
AS40142	7	3740	3767	AS1103	3	1989	1992
AS33491	3	3681	3682	AS33494	9	3700	3707
AS12695	3	2277	2291	AS9120	2	1011	1028
AS41635	3	2250	2252	AS21309	3	1792	1794
AS8342	5	950	2269	AS12306	9	736	4317
AS12301	3	1512	1514	AS9338	9	2040	2046
AS10694	9	4123	4354	AS3778	7	3839	3838
AS6983	50	3214	3234	AS43220	2	980	981
AS39111	3	964	1319	AS3614	4	3539	3541
AS17477	9	18	2062	AS29134	2	939	940
AS43892	2	1532	1533	AS14335	2	3596	3597
AS8972	3	1343	1344	AS3292	36	2068	2478
AS8803	3	2554	2555	AS12874	23	1711	1821
AS1930	2	2204	2208	AS210	13	3709	3712
AS24958	3	947	948	AS15083	28	863	864
AS6400	4	1030	1031	AS1205	4	108	111
AS8508	3	2168	2169	AS11798	18	3822	3824
AS36017	3	4473	4474	AS55454	3	2031	2032
AS36850	4	3655	3657	AS40619	4	3526	3525
AS16276	8	1191	1201	AS5537	2	2288	2295
AS13367	10	3750	3738	AS12312	4	1365	1367

TABLE 4
This table demonstrates how each predicted service composition from algorithm 1 is different from the exact service composition for W_4.

Services	number of WS	IVI	Exact	Services	number of WS	IVI	Exact
AS42695	2	2503	2504	AS50517	3	2304	2306
AS16265	2	2018	2665	AS15806	7	1566	1576
AS8075	11	3830	4183	AS23148	5	583	4233
AS35041	2	2514	2533	AS48347	3	2271	2273
AS760	3	90	92	AS39418	10	963	970
AS4808	10	670	770	AS32475	3	3728	3729
AS32613	13	99	94	AS156	2	4179	4180
AS25260	5	1268	1270	AS16245	2	1000	1018
AS8542	6	2081	2090	AS5409	2	1440	1441
AS3561	25	3647	3579	AS2611	10	132	135
AS14584	4	3894	3896	AS16237	4	1959	1960
AS10929	4	4065	4066	AS12731	4	1361	1378
AS4837	29	739	791	AS2519	4	1857	1859
AS18125	5	1861	1862	AS2200	28	1120	1052
AS26228	6	2077	2080	AS41635	3	2250	2252
AS21309	3	1792	1794	AS10694	9	4123	4124
AS3778	7	3836	3838	AS39111	3	964	1319
AS3614	4	3539	3541	AS1930	2	2204	2208
AS8508	3	2168	2169	AS36850	4	3656	3657