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 service composition combines individual services to generate more powerful services to answer end users needs. Individual services are provided by internet of things components or any web service. Dealing with the service composition optimization process is crucial in large scale IoT context. To improve the service composition process, the main used non-functional parameter is quality of service (QoS). QoS is represented by a set of criteria. We take the assumption that QoS is calculated as a linear combination of these criteria. In this paper, we propose a multi objective Markov Decision Process (MDP) for optimizing the QoS process. The proposed multiple objectives MDP algorithm computes the optimal QoS coefficients and propose a data-driven decision for the best services work-flow on a real world dataset.

Index Terms—Internet of Things, Reinforcement Learning, Quality of Services

1 Introduction

Internet of things offers new possibilities to increase significantly the number of available services to individuals and businesses. It particularly improves the quality of web services and increases in the same time complexity and number of web services. Thus, selecting an appropriate and optimal services workflow becomes a big challenge.

Following the Service Oriented Architecture (SOA) paradigm [Alrifai et al., 2010], composite applications are specified as a process of abstract services. At a given run time and for a given user, each abstract service can be achieved using a selected concrete service from a set of a functionally equivalent ones. This activity is known as a service composition problem [Stelmach, 2013], or more generally as service orchestration problem [Papazoglou and van den Heuvel, 2007]. Several papers are interested in this problem while they study a huge part of the related work in their papers [Khanouche et al., 2017], [Zhang et al., 2016].

From the operational point of view, the main problem concerns the composition process of existing services to achieve more complex task that cannot be filled before. This composition process is close to applications composition in mathematics and concerns concretes services. This composition is always associative and commutative [Gabrel et al., 2015]. 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). The general constraint is, in the concrete service composition process, each concrete service appears one and only once. We will give, in this paper, some elements on the global complexity of this composition problem.

The main objective of this work is to select from all possible permutations the optimal one. It means that we need to define an objective function to 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 like throughput and services response time are combined in a multi-objective function to be optimized.

To deal with this problem several approaches are proposed in the literature including graph search based approaches [Deng et al., 2014], [Jiang et al., 2014], [Rodriguez-Mier et al., 2016], [Siebert et al., 2015], meta-heuristic population based approaches [Deng et al., 2017], [Deng et al., 2016c], [Chandra et al., 2016], [Wu et al., 2016], [Zhang et al., 2016], planning based approaches [Chen et al., 2017], [Zou et al., 2014], integer linear programming approaches [] and machine learning based approaches [Deng et al., 2016b], [Deng et al., 2016a], [Rao et al., 2011], [Ben Mabrouk et al., 2009]. Or service selection methods using reputation models [Wang and Vassileva, 2007], [Wang et al., 2011], recommender system [Manikrao and Prabhakar, 2005], [Liu, 2005]

In this paper, we propose a reinforcement learning approach to select an optimal composition services according to QoS award function without knowing the user preferred rank on the QoS parameters. A minimum number of queries are needed in required situations to select the best service arrangement. To the best of our knowledge, there exist a few number of works that solve this problem using recommender systems on large scale real data. We have performed a large number of experiments on a real dataset [Zheng et al., 2014] and promising results are given in this paper.

The next section gives the technical description of the problem, section **??** details the problem formulation using our formalism. section **??** describes proposed interactive reinforcement learning algorithms to deal with service composition problem et shows who to computes quality of service coefficient and who to propose an optimal service orchestration without a prior knowledge. Section **??** summarizes experimental results.

2 PRELIMINARIES

Internet Of Things services composition is an effective method that combines individual services to generate a more powerful service.

The problem arises when dealing with complex user tasks formed of multiple (abstract) activities, and each activity can be achieved using several services that are functionally equivalent, but providing different Quality Of Service (QoS) levels. If we consider the SOA paradigm, as is shown by the example 1, the question to be asked is then: *"what are the concrete services that should be selected for each activity (Abstract services) in the user's task in order to meet the user's QoS requirements and produce the highest QoS?"*

Example 1.

Fig. 1. Service composition example (must show CS,AS, users, the composition.see example in the paper Selecting Skyline Services for QoS-based Web Service Composition by Alrifai and all)

The term concrete service refers to an invocable service, whereas an abstract service, called also a class of services, defines, in an abstract manner, the functionality of a service. For each abstract service, there may exist several concrete services that have the same functionality but possibly with different quality levels. This definition is given according to the open standard group service oriented architecture (www.opengroup.org/standards/soa). A service is an autonomous and platform independent computational entity, which can be described, published, discovered and dynamically assembled for developing distributed systems. According to SOA, each service has several main properties [1]:

- it represents a business activity with specified outcome,
- it may consists of other underlying services,
- it must be black box for users and
- it is self-contained.

In SOA architecture, a service can be a service provider, service broker or service consumer. There exist several services that utilize web services standards including web services based on Web Services Description Language (WSDL) [Christensen et al., 2001], Restful HTTP [Pautasso, 2014], Microsoft WCF [Cibraro et al., 2010], Apache hrift [Facebook, 2007] and SORCER [Papazoglou et al., 2007][2].

The main SOA's goal concerns how to compose an application including problems related to distribution, deployment and separately maintained services. In SOA, the services utilize meta-data which describing services functional characteristics and non-functional quality of service characteristics. One important and challenging research area is how to propose an appropriate services composition in dynamic and unpredictable environments [Mostafa and Zhang, 2015], [Ben Mabrouk et al., 2009], [Song and Lee, 2013] considering quality of services parameters.

There are several business standards related to the exact composition of SOA [Balzer, 2004], [Valipour et al., 2009]. In many situations an individual service is not able to support complex requirements for the users. For this reason, one operational problem is still the same. For a given set of services, an evaluation function and execution context (users, time steps, service contract, ...), what is the best service composition pattern (service quality, time consume, resources consume, ...) to well address a predefined problem?

In this Web Services Architecture, the W3C working group defines any service as a resource that is characterized by a abstract set of provided functionality. These abstract notions must be implemented by a concrete agents [Booth et al., 2004]. In the rest of this paper, we will denote the possible implementations of W3C services by concrete services. Instances of concrete services are executed over time to response end users needs. We refer to user services to designate concrete service instances linked to a given user. We refer to user services in order to designate concrete service instance linked to a given user. Note that when several service instances are executed over time, temporal references (i.e time step) must be given too.

3 RELATED WORKS

The selection of the service providers in abstract services is influenced by many requirements and definitions. These requirements are different for various applications which are defined based on user preferences. Thus, service selection according to the user requirements such that ensuring the service balance of functionality and non functionality is a big challenge.

There are many works that propose the best service composition based on the known user preferences on QoS. Sometimes, users can not define their preferences on quality of services while they can answer these questions during the execution of services. Thus, there are several works in state of the art that study the service composition selection in interaction with users. Since that these methods are interactive with user of the system, some of them using different model to solve this problem including machine learning and reinforcement learning approaches. In this section we review these approaches with respect to our proposed method in this paper.

3.1 Recommender System based Approaches

[Manikrao and Prabhakar, 2005]
Since there are too many services with similar functionality, the optimal service composition should be selected according to services qualities and capabilities. [Manikrao and Prabhakar, 2005] propose a recommender system that helps the system user to select her optimal services among the list of services with similar functionality. In order to solve this problem, they learn the preferred ranking of services for a given user. When they learn her preferred ranking, they can recommend the best web services in order of her ratings.

[Liu et al., 2009]
[Liu et al., 2009] computes the best web service composition regarding the user's requirements by taking different QoS in different invoke time of web services. Their approach is adaptive in dynamic environments with different QoS values w.r.t their execution in various time steps. They receive the

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

user's requirements at the beginning and then compute the optimal service composition using the Dijistra algorithm. Their approach consists of two steps: first getting the whole possible execution path from a directed acyclic graph for different execution time and second is to acquire the best service composition according to the user's requirements. It should be noticed that the user's requirements in this work are some thresholds on various quality of service attributes that is known before starting the service composition process.

[Wang et al., 2017]

Another interesting concern is taking into account various types of non-functionality including qualitative and quantitative attributes such as provider, location, platform and response time, throughput, reliability availability respectively. [Wang et al., 2017] propose a service selection method w.r.t both parameters types. They solve this problem using two different approaches: In the first approach they combine the two various types (qualitative and quantitative) using a global optimization function. In order to define a ranking on the qualitative attributes, they use conditional preference networks (CP-N) model. In the second approach, they use a genetic algorithm.

[ARASI et al., 2017]

this paper is not a good paper in terms of technical method but it has a good explanation at the beginning in introduction and related works.

They propose a service selection for composite services on the association between QoS parameters and their preferred weight by the user. It means, they defines which service should be selected from a set of backup services respecting the user requirements given by the weights on QoS attributes. [ARASI et al., 2017] assume that the user preferences weight on QoS parameters is given as $(\bar{W} = (w_1, \cdots, w_d))$. Thus they normalize all the extracted information on different web services using the weight vector $\bar{W}$. It means instead of the given quality qos_i^j for abstract service i on attribute j, they use:

$$\text{update } qos_i^j = w_i \left[\frac{qos_i^j}{\max \sum_m qos_m^j} \right] \tag{1}$$

I did not understand what is maximum in the fraction?

in fact, they measure the difference of qos_i^j from the maximum normalized value w.r.t other existed qos on the j-th QoS attribute. In this study, normally the user weight preferences on QoS attributes are given and they are used to calculate the optimal workflow for the service composition problem.

3.2 Machine Learning based Approaches

3.3 Reinforcement Learning based Approaches

[Wang et al., 2010]

This paper propose a method to solve a composition service problem in a dynamic environment. Since web services information about their qualities changes during time and under various criteria, they produce a mechanism that adapt itself using reinforcement learning approach. For instance, some services may not be available when their providers leave the business or some may upgrade their quality of Services (QoS). As an example, Amazon EC2 has decreased its prices for several times (15% cut in November 2009) or the growth of costumers usually increases the response time of a service. Thus in the introduced method by [Wang et al., 2010], they learn the QoS information through executing the services while the optimal workflow is updated to the change of the environment.

They model a service composition problem as a Markov Decision Process and implement the Q-learning method on it. The important point is their approach in defining reward values for the MDP model. They assume if the preference weights of users on QoS attributes is given as $\bar{W} = (w_1, \cdots, w_d)$, the value of each executed web services should be defined as:

$$R(ws) = \sum w_i \times \frac{qos_i^j - qos_i^{\min}}{qos_i^{\max} - qos_i^{\min}} \tag{2}$$

where qos_i^j is the observed value of the i-th attribute for service ws. And $qos_i^{\max}$ an $qos_i^{\min}$ demonstrates the maximum and minimum values of the i-th attribute for all web services.

[Mostafa and Zhang, 2015]

[Mostafa and Zhang, 2015] propose two multi-objective methods to handle QoS-aware web services composition with conflicting objectives and various restrictions on the quality of services. In their reinforcement learning based approaches, they consider that the user requirements are not known. They model a web service dynamic environment as a partially observed MDP and use the Q-learning similar approach to solve the service composition problem. In the single policy multi-objective service composition technique, they assume each QoS-objective as a separate learning agent and they select the optimal web service in each class of services (with the same functionality) such that the service has the highest sum of QoSs for one of its attributes. It is obvious that this average method accumulate more quality of services rather that the linear combination defined among user's weights (preferences) and the QoS attributes such as [Wang et al., 2010].

The second method computes a set of Pareto optimal service compositions, which have the equivalent quality to satisfy multiple QoS-objectives with different user preferences. This method is completely based on the Q-learning method except that they don not select the optimal Q-value function in each iteration. Since Q-value function is a vector including several values for various QoS attributes, computing the most preferred vector in each iteration without knowing user's preferences is impossible. Thus, [Mostafa and Zhang, 2015] gets the convex hull vertices of all Q-value functions. It means they keep all Q-values with containing maximum value on one the QoS attributes.

4 Problem Formulation

According to problem simplifications given before, The problem of building a service that meets an end-to-end need can be defined as follows:

Given a set of n services $S = \{S_1, \ldots, S_n\}$. Each service S_i can be implemented by one of possible 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 by S_{ij}^k an instance S_{ij} executed by the actor a_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. For sake of simplicity, the service instance S_{ij} executed by the actor a_k at the time t is denote by $S_{ij}^k(t)$ in the rest of this paper[3].

Example 2. Table 1 is an extracted line from the real dataset [Zheng et al., 2014], [Zhang et al., 2016] used in our experiment. According to the upper notation, this line should be denoted as $S^{97}_{19994,3104}(5)$. The line represents the quality of abstract service 19994 (time response and throughput) implemented with the concrete service 31410 by user 97 at time step 5.

TABLE 1
Real dataset instance used in our experiments [Zheng et al., 2014], [Zhang et al., 2016].

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 each service, a set of parameters are added to quantify the services according to various criteria. Most considered criteria are: time response, throughput, reliability, availability, price, execution time and etc. [**?**]. These criteria are associated with each execution and can be also associated to various users, to abstract services or concrete services.

Let us consider $Q = \{q_1, \ldots, q_m\}$ as the set of all possible criteria. As mentioned above, 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 reduce the formulas 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. To evaluate the concrete service global quality, we add some weighted coefficients in order to rank service preferences.

In SOA architectures, the orchestration process composes existing services in order to create a new service having central control over the whole process [**?**] . In our context we limit the orchestration process to abstract services composition. This composition process is guided by one main constraint between abstract services: when current abstract service needs results from some others, those ones must be executed before. When all constraints are given, precedence graph is built (see example 3). Each complete path (including each abstract service ones) defines one possible orchestration.

Example 3. Let us consider four abstract services S_1, S_2, S_3, S_4, such that: S_2 and S_3 need S_1 results and S_4 is executed after S_2. Figure 3 gives the abstract services precedence graph. 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.

Each abstract service may be realized with several possible concret services. When abstract service orchestration

3. A concrete service S_{ij} executed by an actor a_k at time t is called *execution*.

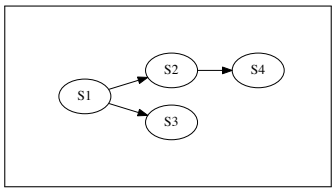

Fig. 2. Precedence graph between four abstract services.

is defined, we need to choose for each abstract service an appropriate concrete service that performs the corresponding service. in SOA architectures, workflow describe how activities or tasks should be carried out. We denote by concrete workflow the possible concrete services organisation in order to realize an end to end user service in a respect to a given abstract service orchestration. In the e

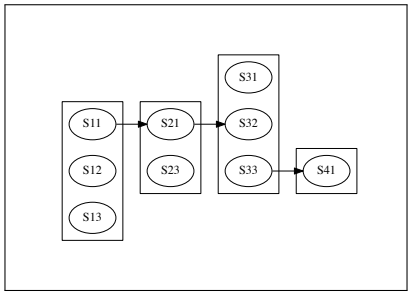

Fig. 3. The concrete workflow $(S_{11}, S_{21}, S_{32}, S_{41})$ is a possible organisation to realize the orchestration given in figure 3.

Let consider f as a global objective function for the end-to-end service. This function optimizes Q criteria list according to 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.

For each $k \in \{1..M\}$, to obtain F_{ψ_k} it is necessary to select the best concrete service concatenation according to the Q criteria between concrete services in the given ψ_k orchestration according to predefined optimisation function $\varphi_{ki,(k\in\{1,\ldots,M\},i\in\{1,\ldots,n_k\}}$ between criteria of Q.
Without loss of generality, the service selection problem can be defined as follow:

$$Find\ the\ evaluation\ function\ f\ as\ the\ best\ value\ of\ F_{\psi_k}$$

Example 4. Let us consider a problem with three abstract services $\{S_1, S_2, S_3\}$ with 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 ones and finish with S_3 concretes services). Let us consider a problem with two evaluation criteria $Q = \{q_1, q_2\}$ (for instance, throughput and service time response) such that the concrete services evaluation function to be optimized is defined as 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:

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

However, from operational point of view, the evaluation of the quality of concrete services is estimated over concrete service executions (instances) over the time. Additional sub-problems must be solved before optimizing the previous problem among which: how to solve the service composition problem without any knowledge about user preferences criteria. As is shown in related work section, several works are done using various approaches. However some additional constraints make this problem more difficult. The main constraint concerns the fact that no preferences are given on the components of the evaluation function. We propose in our work to learn this preferences from data by using an original approach based on discrete time vector valued MDP especially to estimate quality of service criteria weights for any abstract service orchestration without using user quality of service preferences.
In section **??** we present a dataset used for our experiments and our considered assumptions. Section **??** summarizes Vector-valueted decision process (VMDP) needs for our work and gives the VMDP service selection formulation according to the context presented in section **??**. Section **??** presented used algorithms to solve the problem and gives some complexity issues. Obtained results are given in section **??**.

4.1 On the global problem complexity

Without loss of generality, let us consider the problem with n abstract services $S_1, \ldots, S_n$, 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. We assume also than each end-to-endThe 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 select the best one according to given user or computed preferences.

As far as we know, there is no existing work dealing with this enumeration problem. 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.

4.1.1 Complexity when there exist a total order between abstract services

Let us consider $S_1, \ldots, S_n$ 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 $n! * n^m$.

4.1.2 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.

4.1.3 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[4]. 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 $2 * (n-1)^m * (2^{n-1} - 1) * (n-1)!$.

4.1.4 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 egaI 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: $\sum_{i=1}^{i=n}(m^i * S(n, i) * i!) \leq CSC \leq \sum_{i=1}^{i=n}(((n-i+1) * m)^i * S(n, i) * i!)$.

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

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

5 MDP Problem Formulation

Markov Decision Processes (MDPs) are the suitable models for sequential decision problems such as QoS decomposition problems. In this section, we demonstrate how to use the MDP model to model and solve the service composition problem. We are looking for an optimal QoS-selection corresponding the user's preferences on qualities of services. Since the user's preferences among service qualities are unknown, we consider the problem as a multi-objective service composition under uncertainties. For this reason, we utilize a partially known MDP model and more particularly a Vector-valued MDP (VMDP) model. Before getting into

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

details, it is required to describe some preliminary properties and definitions as follows.

Property 1. A **Concrete Service** S_{ij}, or a concrete service executed by an actor with or without time reference properties can be summarized on two parts: functional properties and non-functional ones.

- Functional : S_{ij} is under the form of transaction function Action(S_{ij}) that takes an input data vector InputData(S_{ij}) to produce an output data vector OutputData(S_{ij})
- Non-functional : is defined by a QoS attributes vector QoS(S_{ij}), the energy profile EProf(S_{ij}), etc.

Property 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 nonfunctional 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. For the sake of simplicity, we will demonstrate the modeling process step by step.

5.1 Vector-valued Markov Decision Process

Referring to Section 4 invoking each concert service in a time step t produces different quality of services. Reminding Example 2, invoking concrete service 3104 for abstract service 19994 at time 5 gives two different values for the response time and throughput: $Q(s_{ij}) = (\text{rt}(S_{ij}), \text{tp}(S_{ij}))$.

Definition 1. Formally, a *Discrete-time Markov Decision Process (Discrete-time MDP)* [Alagoz et al., 2009] 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[5].
- States: S is a finite set of States
- Actions: $A(s)$ is a finite set of actions that agent can select to interact with the environment.
- 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: $r_t : S \times A \longrightarrow \mathbb{R}$. $r_t(s,a)$ quantifies the utility of performing action a in state s at the t time step.

A *Decision rule* d_t is a function depends 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: $\forall\, t \in \{1, \cdots, T\}\ d_t = d$.

A solution for 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:

$$v_N^\pi(s) = r_N(s, \pi(s)) \ \ \forall s \in S_T \tag{3}$$

where S_T is the set of terminal states as a subset of all states S: $S_T \subset S$. Remind that there is no action decision in S_T as the set of terminal states. For the rest of time steps $t < T$, the value function is defined as:

$$v_t^\pi(s) = r_t(s, \pi(s)) + \gamma \sum_{s' \in S} P_t(s'|s, \pi(s)) v^\pi(s') \tag{4}$$

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

$$\pi \succeq \pi' \Leftrightarrow \forall s \in S\ v_0^\pi(s) \geq v_0^{\pi'}(s) \tag{5}$$

An MDP solution is an *optimal policy*, that is the highest policy with respect to the other policies and the preference relation $\succeq$, i.e. $\pi*$ is an optimal policy if:

$$\pi^* \text{s.t.}\ \forall\, \pi,\ \pi^* \succeq \pi \tag{6}$$

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

$$v_N^* = r_N(s)\ \forall s \in S_T \tag{7}$$

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

$$v_t^*(s) = \max_{a \in A(s)} \left\{ r_t(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_{t+1}^*(s') \right\} \tag{8}$$

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

$$Q_t(s,a) = r_t(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_{t+1}^*(s') \tag{9}$$

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

$$a_t^* \in \text{argmax}_{a \in A(s)} \{Q_t(s,a)\} \ \ \text{for } t = 1 \cdots N-1 \tag{10}$$

Sometimes, selecting an action in a given state may have several effects instead of only one value. It means, by extending the discrete-time MDP to a discrete-time vector-valued MDP (discrete-time VMDP), we have:

Definition 2. [Alizadeh et al., 2016] 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.

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

5.2 MDP for Service Compositions

By modeling the service composition as a discrete-time VMDP, we can compute the best selected work-flow respecting user preferences on quality of services attributes. Note that we are allowed to communicate with users in order to get information about their priorities and preferences.

We use discrete-time VMDP modeling to select the optimal concrete service in each abstract service w.r.t time stage t. For the sake of simplicity, this model is noted as discrete-time VMDP-Service Composition (discrete-time VMDP-SC) which is introduced in the rest of this section (see [Wang et al., 2010], [Mostafa and Zhang, 2015]).

Definition 3. 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.
- $\{S_{ij}\}_j$ is the 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. (Notice that d is the number of quality of service attributes and we have $\overline{Q}_t(S_i, S_{ik}) = (q_{1t}(S_i, S_{ik}), \cdots, q_{dt}(S_i, S_{ik}))$).
- AS_T is the set of terminal services. It means the execution of the service composition terminates by ending in one of these states.

The solution for QoS-aware service selection is defined as a policy in VMDP-SC model.

Definition 4. A **policy service composition** $\pi : AS \longrightarrow SC$ 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 attributes.

This policy is known as a workflow or plan in the IOT literature. Since reward values in MDP-SC are the Q vectors for each concrete services, each policy should be evaluated with a vector function (see Equation 4):

$$\bar{v}_t^{\pi}(S_i) = \overline{Q}_t(S_i, \pi(S_i)) + \gamma \sum_{S_j \in S} P(S_j|\pi(S_i), S_i)\bar{v}_{t+1}^{\pi}(S_j) \tag{11}$$

By assumption, this function is called Q *vector valued function*. Now, comparing two workflows/policies boils down to comparing two vectors. The optimal workflows satisfying various users with different preferences among the QoS attributes are not the same.

Example 5. Give an example from the data base to explain the reason of last written phrase.

Thus, we need a model that presents the user preferences over quality of services attributes.

Example 6. Another examples that shows the relation between user preferences on the qos attributes.

For this reason, we define user preferences over the QoS attributes as a linear combination of the quality of service attributes. 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:

$$Q_t(S_i, S_{ij}) = \sum_{k=1}^{d} \bar{w}_k q_{kt} = \bar{w} \cdot \overline{Q}(S_i, S_{ij}) \quad \forall t = 1, \cdots, N \tag{12}$$

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 the user preferences on the QoS attributes is given, the optimal workflow can be computed easily. Since defining a weight to each QoS attribute is not obvious by users (we need stronger motivation related to IOT), we assume that $\bar{w}$ is unknown and try to find the best workflow/policy/plan by querying users when it is necessary. (this phrase should be rephrased and very strong motivation should be added to this part.).

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:

$$W = \{(w_1, w_2, \cdots, w_d) \mid \sum_{i=2}^{d} w_i \leq 1 \text{ and } w_1 = 1 - \sum_{i=2}^{d} w_i\} \tag{13}$$

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:

$$\bar{v}^a \succeq_P \bar{v}^b \Leftrightarrow \forall\ i\ a_i \geq b_i \tag{14}$$

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

$$\bar{v}^a \succeq_K \bar{v}^b \Leftrightarrow \forall\ \bar{w} \in W\ \bar{w} \cdot \bar{v}^a \geq \bar{w} \cdot \bar{v}^b \tag{15}$$

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

Remind that, the Kdominance comparison is a linear programming problem. it means, $\bar{v}^a \succeq_K \bar{v}^b$ satisfies, if there is a non-negative solution for the following LP:

$$\begin{cases} \min \bar{w} \cdot (\bar{v}^a - \bar{v}^b) \\ \text{subject to } \ \bar{w} \in W \end{cases} \tag{16}$$

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 policy/workflow that gives the best trade-off among multiple QoS criteria, satisfying the user preferences on QoS attributes by querying users very few times.

In this section, we propose an Algorithm namely *Interactive Value Iteration for Service Composition (IVI-SC)*. Previously,

we explained how model web services as a discrete-time MDP in order to solve the service composition problem. In this section, we demonstrate, how to find the solution using the existed solutions for MDPs. Some researchers use interactive value iteration methods to find the optimal service composition respecting the user of system preferences [Weng and Zanuttini, 2013], [Alizadeh et al., 2016]. In this paper, we modified the interactive value iteration on a finite-horizontal MDP to find the best service composition satisfying users' priorities on quality of services.

6 Interactive Reinforcement Learning Algorithms for the Service Compositions

In this section 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 to find 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 8 and finds the optimal actions given in equation 10 to obtain the optimal policy/work-flow. Our solution is introduced in Algorithm 6.

Data: VMDP-SC$(T, AS, CS(), P_t, \bar{r}_t)$, a W polytope of user weights on objectives, precision ϵ
Result: The optimal service selection policy for the given user.
$t \longleftarrow T$
$\pi_{\text{best}} \longleftarrow$ choose random policy
$\bar{v}_T(s_T) \longleftarrow (0, \cdots, 0)^6 \ \forall s_T$ at time T
$\mathcal{K} \longleftarrow$ set of constraints on W
while $t \geq 0$ **do**
  $t \longleftarrow t-1$
  for ***each*** $h_t = (h_{t-1}, cs_{t-1}, as_t) \in H_t$ **do**
    best $\longleftarrow (0, \cdots, 0)$
    for ***each*** $cs \in CS(as_t)$ **do**
      $\bar{v}_t(as_t) \longleftarrow \overline{QoS}_t(as_t, cs) + \sum_{as'} P_t(as'|as_t, cs)\bar{v}_{t+1}(as')$
      (best , $\mathcal{K}$) $\longleftarrow$ getBest(best, $\bar{v}_t, \mathcal{K}$)
      $\bar{v}_t(as_t) \longleftarrow$ best
      if *best* = $\bar{v}_t(as_t)$ **then**
        $\pi_{\text{best}}(as_t) \longleftarrow cs$
      end
    end
  end
end
return π_{best}

Algorithm 1: How to select the best composite for each abstract service respecting user preferences on QoS attributes

In iterative algorithm 6, first we assign a zero vector to the set of states (abstract services) at time step T. At each iteration on the abstract services, the algorithm should solve equation 8, using the value from the previous iteration. Notice that $h_t = (h_{t-1}, cs_{t-1}, as_t)$ indicates that by invoking a concrete service as_{t-1} at time $t-1$, the system will be at concrete service as_t. In the finite horizon time (our case), the iteration continues until either this difference becomes small enough or the horizon time steps finish.

Data: finds the more preferred vector between two vectors $\bar{v}$ and $\bar{v}'$ w.r.t $\mathcal{K}$
Result:
if *paretodominates*$(\bar{v}, \bar{v}')$ **then**
  return $(\bar{v}, \mathcal{K})$
end
if *paretodominates*$(\bar{v}', \bar{v})$ **then**
  return $(\bar{v}', \mathcal{K})$
end
if *Kdominates*$(\bar{v}, \bar{v}', \mathcal{K})$ **then**
  return $(\bar{v}, \mathcal{K})$
end
if *Kdominates*$(\bar{v}', \bar{v}, \mathcal{K})$ **then**
  return $(\bar{v}', \mathcal{K})$
end
$(\bar{v}_{\text{best}}, \mathcal{K}) \longleftarrow$ query$(\bar{v}, \bar{v}', \mathcal{K})$
return $(\bar{v}_{\text{best}}, \mathcal{K})$

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.

Since the quality of services are the d dimensional vectors, solving equation 8 and finding the maximum among the vectors is not obvious. For this reason, we remind three comparison methods (presented in equations 14, 13 and 14) and utilize function **Best** (Algorithm 6). This function receives two d dimensional vectors with the W polytope confining the user weight preferences on the quality of services. If pareto comparison does not have a solution, the Kdominance comparison method will try this comparison out. Otherwise the query function should be called (given in Algorithm 6) where the user response to the comparison between the two given vectors, adds a new constraint to the W.

Algorithm 6 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 $best = \bar{v}_t(as_t)$ checks if the best selected concrete service for as_t has been changed regarding the previous iteration. If it was the case, the optimal concrete service should be replaced by the concrete service cs which generates a better vector value for as_t.

add a part on complexity of algorithms and so on.

7 Performance Evaluation

We evaluate our methods on a public available dataset containing two parameters for quality of services: response

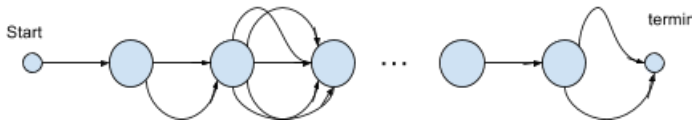

Fig. 4. A sequential form of abstract service connection

time and throughput. These are the records between 339 users and 5825 web services distributed worldwide [Zheng et al., 2014], [Zhang et al., 2016]. The dataset also includes some information about user features, service features such as countries, autonomous systems, IP dresses, latitude and longitude. In the studied data base, users execute various web services in different time slices. Practically, the information are given only for 64 time steps.

In this section, we explain first how to model the studied dataset as a VMDP-SC and then we will examine our algorithm on the dataset in two different approaches including the divided dataset based on the users' existed information and a filtered version of database.

7.1 Model DataSet as a VMDP

The main issue in implementing Algorithm 6 on any database is that how to model the given set as a vector-valued MDP. In the supported dataset [Zheng et al., 2014], [Zhang et al., 2016], there are several text files including wslist.txt, userlist.txt, rtdata.txt and tpdata.txt. Using the wslist.txt, we extract a list of web services and their related abstract services, if a related abstract service exist for the selected web service. In total, there are 5825 web services and 137 abstract services. The userlist.txt includes the information about 338 users of different web services. The two other files rt.txt and tp.txt are consist of the data about user execution of web services in 64 different time steps on response time and throughput respectively. The point is that, they present the related results of 142 system users. Each invoked web service by a special user has two parameters for measuring the service quality: response time and throughout.

According to the extracted information from the database and VMDP-SC definition (see 3) we have:

- number of episodes: $N = 64$
- 137 number of abstract services
- 5825 number of concrete services (in our case web services)
- The transition function and terminal states depends on the proposed model or relation among the abstract services.

  1 For the sequential model (see Fig 7.1) the start sate is an empty state and connected to the first selected abstract service in the model. While the terminal state is an empty state to indicate the MDP is a finite horizon one. The transition probability $P_t(S_j|S_i, S_{ik})$ is 1, if web service S_{ik} is invocable for abstract service S_i(according to our database) and abstract service S_j is the next demanded service in our selected sequential MDP model for all time steps $t = 0 \cdots 63$.

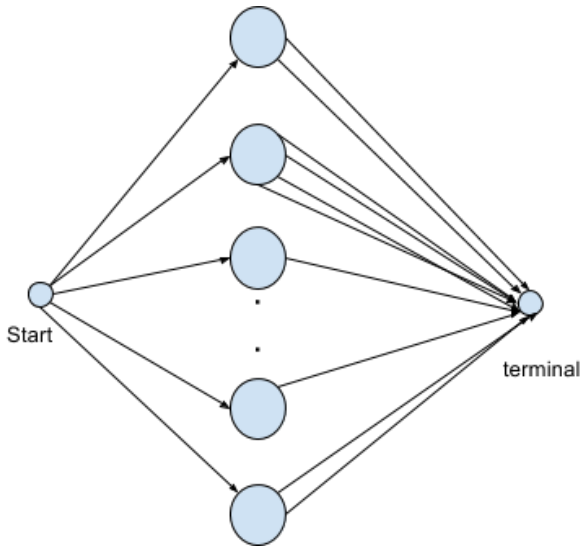

Fig. 5. A parallel form of abstract service connection

user	weight vector
$\bar{W}_1$	[0.07075913789991828, 0.9292408621000817]
$\bar{W}_2$	[0.8573741847324399, 0.14262581526756013]
$\bar{W}_3$	[0.1696287781131175, 0.8303712218868825]
$\bar{W}_4$	[0.6451844883834318, 0.3548155116165682]
$\bar{W}_5$	[0.18190820427369447, 0.8180917957263055]

TABLE 2
The weight vectors for 5 random system users.

  2 For the parallel model (see Fig 7.1), the start and terminal states are the empty sates such that the start state has access to all abstract services and the abstract services are connected to the terminal state based on the possible web services for each one. In this case, $P_t(S_j|S_i, S_{ik})$ is 1 if S_j is the terminal state, otherwise it is 0 for all time step $t = 0, \cdots, 63$.

- and the $\overline{Q}_t$ function is built based on the extracted data on web services and their two qualities (response time and throughout).

7.2 Classified Dataset

7.3 Filtered Dataset

lambdas should be replaced by $\bar{W}$ weight vectors in the tables.

Our dataset is a real database generated by observing various users using 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 quality of web services of two files tp.txt and rt.txt, we recognize that there is no information on some web services related to some abstract services. For this reason, we remove the abstract services without any information on their related web services.

In order to evaluate our algorithms performance, we have observed the results for 5 different system users with various preferences on service qualities: response time and throughput. The user weights vector (W) on service qualities

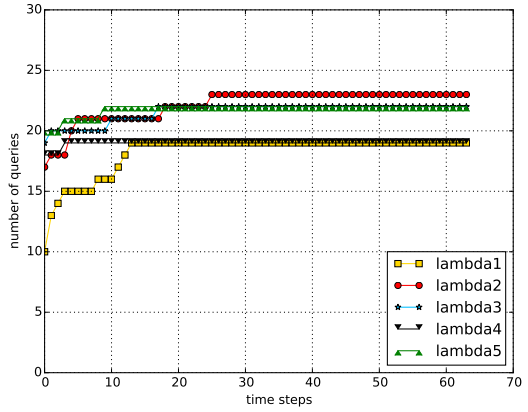

Fig. 6. This figure shows the number of queries proposed to the user during each time step. The weight preferences are based on table 2.

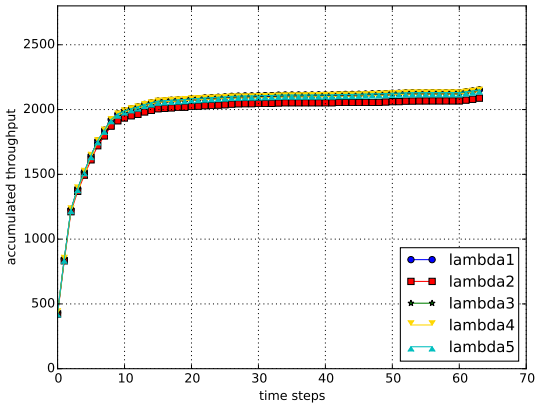

Fig. 8. this figures demonstrates how the accumulated throughout increases during each time step. The weight preferences are based on table 2.

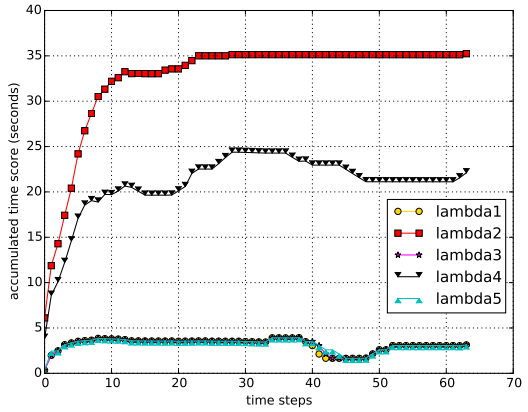

Fig. 7. This figures demonstrates how the accumulated response time increases during each time step. The weight preferences are based on table 2.

is given in table 2. Notice that the weight preferences on quality of services are unknown to our algorithms. We keep mention them here, because they are required for the rest of experiments.

Figure 6 demonstrates how we communicate with users during 64 time steps. As it can be seen in this figure, we don't ask more than 23 queries to the users with various preferences on quality services. The number of communications with users are important because we can not interrupt users with too many questions.

On the other hand two Figures 7 and 8 show how service qualities response time and throughput respectively change with respect to time step for the 5 given user types. Figure 7 indicates that three users with $\bar{W}_1$, $\bar{W}_3$ and $\bar{W}_5$ weight preferences have the similar changing in their response time evolution. While W_2 causes the highest computed response time. In the other hand, Figure 8 shows that all tested preferred weights accumulates with almost the same steep.

We are interested in the optimal workflow i.e. which concrete services for each abstract service should be selected in order to maximize the service qualities satisfying user weight preferences. Therefore Table3 observes the differences between the proposed optimal workflow by algorithm 6 and an exact solution. Remind that the exact workflow is calculable by taking the weight into account and using a classical approach on MDPs such as Value iteration method **??**. In VMDP formulation (see 3) the reward values are vectors, but knowing the user weights $\bar{W}$, transfers the quality vectors to the qulaity values: $\bar{W} \cdot \bar{Q}$. For this reason, the value iteration is applicable on MDPs service compositions. Table3 compares the workflow proposed by algorithm 6 and the exact workflow computed by value iteration method [ARASI et al., 2017]. It can be seen that the computed optimal workflow by our approach is almost the same as the exact computed workflow for the 5 user preferences types except very few number of services. This guaranties the accuracy of our approach.

8 Conclusion and Future Works

Acknowledgments

The authors would like to thank...

References

[Alagoz et al., 2009] Alagoz, O., Hsu, H., Schaefer, A. J., and Roberts, M. S. (2009). Markov decision processes: A tool for sequential decision making under uncertainty. *Medical Decision Making*, 30(4):474–483.

[Alizadeh et al., 2016] Alizadeh, P., Chevaleyre, Y., and Lévy, F. (2016). Advantage Based Value Iteration for Markov Decision Processes with Unknown Rewards. In *International Joint Conference on Neural Networks (IJCNN 2016)*, Vancouver, Canada.

[Alrifai et al., 2010] Alrifai, M., Skoutas, D., and Risse, T. (2010). Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 11–20. ACM.

[ARASI et al., 2017] ARASI, F. E. M., GOVINDARAJAN, S., and SUBBARAYAN, A. (2017). Weighted quality of service based ranking of web services. In *Indian Journal of Science and Technology*.

[Balzer, 2004] Balzer, Y. (July 16, 2004). Improve your soa project plans.

[Ben Mabrouk et al., 2009] Ben Mabrouk, N., Beauche, S., Kuznetsova, E., Georgantas, N., and Issarny, V. (2009). *QoS-Aware Service Composition in Dynamic Service Oriented Environments*, pages 123–142. Springer Berlin Heidelberg.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture.

TABLE 3
This table demonstrates how each predicted policy from algorithm 6 is close to the optimal workflow when the λ is known.

		lambda 1		lambda 2		lambda 3		lambda 4		lambda 5	
Services	number of WS	IVI	Exact	IVI	Exact	IVI	Exact	IVI	Exact	IVI	Exact
AS5786	3	2212	2214	2213	2214	2212	2214	2213	2212	2212	2214
AS1659	1	2585	2585	2585	2585	2585	2585	2585	2585	2585	2585
AS680	44	1398	1401	1398	1401	1398	1401	1398	1401	1398	1401
AS73	5	3914	3914	3912	3912	3914	3914	3912	3912	3914	3914
AS156	2	4180	4180	4179	4179	4180	4180	4179	4179	4180	4180
AS559	2	None	None	None	None	None	None	None	None	None	None
AS19262		3900	3900	3900	3900	3900	3900	3900	3900	3900	3900
AS553	2	1466	1466	1448	1448	1466	1466	1448	1466	1466	1466
AS2852	2	920	920	920	920	920	920	920	920	920	920
AS3112		None	None	None	None	None	None	None	None	None	None
AS760	4	91	91	91	91	91	91	91	91	91	91
AS766	11	2366	2366	2366	2366	2366	2366	2366	2366	2366	2366
AS1930	2	2204	2208	2204	2204	2204	2208	2204	2204	2204	2208
AS137	2	None	None	None	None	None	None	None	None	None	None
AS239	3	None	None	None	None	None	None	None	None	None	None
AS131	4	4060	4060	4060	4060	4060	4060	4060	4060	4060	4060
AS20130	2	3695	3695	3695	3694	3695	3695	3695	3695	3695	3695
AS237	6	816	816	816	816	816	816	816	816	816	816
AS17	4	4136	4136	4134	4133	4136	4136	4134	4133	4136	4136
AS52	3	4464	4464	4464	4464	4464	4464	4464	4464	4464	4464
AS2497	1	1885	1885	1885	1885	1885	1885	1885	1885	1885	1885
AS13041	24	2375	2375	2375	2375	2375	2375	2375	2375	2375	2375
AS7377	5	3782	3778	3782	3778	3782	3778	3782	3778	3782	3778
AS32	9	4388	4388	4388	4388	4388	4388	4388	4388	4388	4388
AS3	2	None	None	None	None	None	None	None	None	None	None
AS9	1	None	None	None	None	None	None	None	None	None	None
AS8	3	3668	3668	3668	3667	3668	3668	3668	3668	3668	3668
AS111	1	4239	4239	4239	4239	4239	4239	4239	4239	4239	4239
AS2107	1	2347	2347	2347	2347	2347	2347	2347	2347	2347	2347
AS1741	2	1044	1044	1043	1043	1044	1044	1043	1043	1044	1044
AS2900	1	None	None	None	None	None	None	None	None	None	None
AS209	34	4031	4031	4031	4031	4031	4031	4031	4031	4031	4031
AS5723		3832	3832	3851	3851	3832	3832	3832	3832	3832	3832
AS7018		4393	4393	4393	4393	4393	4393	4393	4393	4393	4393
AS7132	6	4209	4211	4209	4209	4209	4211	4209	4211	4209	4211
AS87	5	4112	4112	4112	4112	4112	4112	4112	4112	4112	4112
AS224	9	2075	2075	2075	2075	2075	2075	2075	2075	2075	2075
AS2200	29	1108	1108	1108	1108	1108	1108	1108	1108	1108	1108
AS786	254	3013	3013	3013	3013	3013	3013	3013	3013	3013	3013
AS4538	3	817	817	817	817	817	817	817	817	817	817
AS25	1	None	None	None	None	None	None	None	None	None	None
AS18	1	None	None	None	None	None	None	None	None	None	None

[Chandra et al., 2016] Chandra, M., Agrawal, A., Kishor, A., and Niyogi, R. (2016). Web service selection with global constraints using modified gray wolf optimizer. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1989–1994.

[Chen et al., 2017] Chen, N., Cardozo, N., and Clarke, S. (2017). Goal-driven service composition in mobile and pervasive computing. *IEEE Transactions on Services Computing*, PP(99):1–1.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1.

[Cibraro et al., 2010] Cibraro, P., Claeys, K., Cozzolino, F., and Grabner, J. (June 15, 2010). *Professional WCF 4: Windows Communication Foundation with .NET 4*. Wrox.

[Deng et al., 2016a] Deng, S., Huang, L., Hu, D., Zhao, J. L., and Wu, Z. (2016a). Mobility-enabled service selection for composite services. *IEEE Trans. Services Computing*, 9:394–407.

[Deng et al., 2017] Deng, S., Huang, L., Taheri, J., Yin, J., Zhou, M., and Zomaya, A. Y. (2017). Mobility-aware service composition in mobile communities. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(3):555–568.

[Deng et al., 2014] Deng, S., Huang, L., Tan, W., and Wu, Z. (2014). Top-k automatic service composition: A parallel method for large-scale service sets. *IEEE Transactions on Automation Science and Engineering*, 11:891–905.

[Deng et al., 2016b] Deng, S., Wu, H., Hu, D., Zhao, J. L., undefined, undefined, undefined, and undefined (2016b). Service selection for composition with qos correlations. *IEEE Transactions on Services Computing*, 9:291–303.

[Deng et al., 2016c] Deng, S., Wu, H., Taheri, J., Zomaya, A. Y., and Wu, Z. (2016c). Cost performance driven service mashup: A developer perspective. *IEEE Transactions on Parallel and Distributed Systems*, 27:2234–2247.

[Facebook, 2007] Facebook (2007). Apache trift tm.

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

[Jiang et al., 2014] Jiang, W., Hu, S., and Liu, Z. (2014). Top k query for qos-aware automatic service composition. *IEEE Transactions on Services Computing*, 7(4):681–695.

[Khanouche et al., 2017] Khanouche, M. E., Attal, F., Amirat, Y., Chibani, A., and Kerkar, M. (2017). Qos-aware services composition based on clustering technique for ambient intelligence. *IEEE Transactions on Automation Science ans Engineering*, 16.

[Liu, 2005] Liu, W. (2005). 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.

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

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

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

[Papazoglou et al., 2007] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, pages 38–45.

[Papazoglou and van den Heuvel, 2007] Papazoglou, M. P. and van den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415.

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

[Rao et al., 2011] Rao, R., Savsani, V., and Vakharia, D. (2011). TeachingâĂŞlearning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43:303 – 315.

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

[Siebert et al., 2015] Siebert, J., Cao, J., Lai, Y., Guo, P., and Zhu, W. (2015). Lasec: A localized approach to service composition in pervasive computing environments. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):1948–1957.

[Song and Lee, 2013] Song, S. and Lee, S.-W. (2013). A goal-driven approach for adaptive service composition using planning. *Mathematical and Computer Modelling*, 58:261 – 273.

[Stelmach, 2013] Stelmach, P. (2013). Service composition scenarios in the internet of things paradigm. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pages 53–60. Springer.

[Valipour et al., 2009] Valipour, M. H., AmirZafari, B., Maleki, K. N., and Daneshpour, N. (2009). A brief survey of software architecture concepts and service oriented architecture. *2nd IEEE International Conference on Computer Science and Information Technology*, page 34âĂŞ38.

[Wang et al., 2017] Wang, H., Ma, P., Yu, Q., Yang, D., Li, J., and Fei, H. (2017). Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition. *Journal of Parallel and Distributed Computing*, 100:71–84.

[Wang et al., 2010] Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., and Bouguettaya, A. (2010). *Adaptive Service Composition Based on Reinforcement Learning*, pages 92–107. Springer Berlin Heidelberg.

[Wang et al., 2011] Wang, P., Chao, K.-M., Lo, C.-C., and Farmer, R. (2011). An evidence-based scheme for web service selection. *Information Technology and Management*, 12(2):161–172.

[Wang and Vassileva, 2007] Wang, Y. and Vassileva, J. (2007). A review on trust and reputation for web service selection. In *Distributed Computing Systems Workshops*, pages 25–25.

[Weng and Zanuttini, 2013] Weng, P. and Zanuttini, B. (2013). 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. AAAI Press.

[Wu et al., 2016] Wu, Q., Ishikawa, F., Zhu, Q., and Shin, D. (2016). Qos-aware multigranularity service composition: Modeling and optimization. *IEEE Trans. Systems, Man, and Cybernetics: Systems*, 46(11):1565–1577.

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

[Zheng et al., 2014] Zheng, Z., Zhang, Y., and Lyu, M. R. (2014). Investigating qos of real-world web services. *IEEE Transactions on Services Computing*, 7(1):32–39.

[Zou et al., 2014] Zou, G., Lu, Q., Chen, Y., Huang, R., Xu, Y., and Xiang, Y. (2014). Qos-aware dynamic composition of web services using numerical temporal planning. *IEEE Trans. Serv. Comput.*, 7(1):18–31.

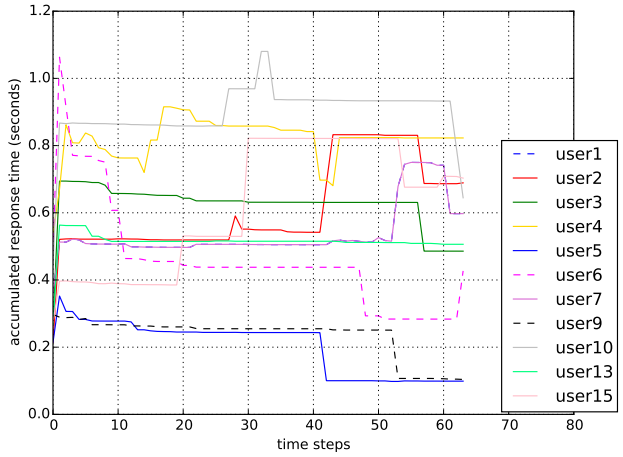

Fig. 9. accumulated response time vs time step for several users extracted from the main data base for $\bar{\lambda}_1 = [0.07075913789991828, 0.9292408621000817]$

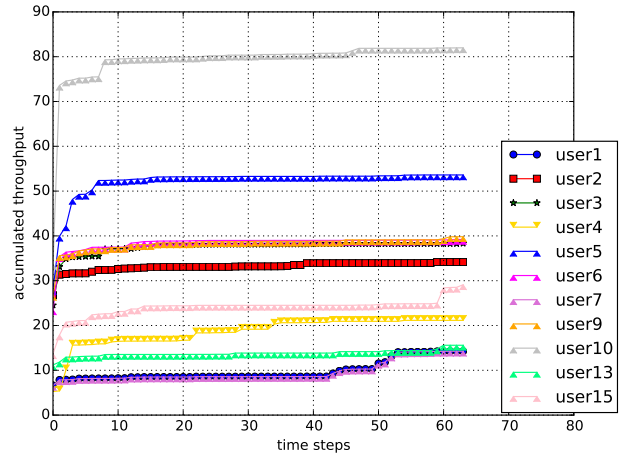

Fig. 10. accumulated throughout vs time step for several users extracted from the main data base for $\bar{\lambda}_1 = [0.07075913789991828, 0.9292408621000817]$

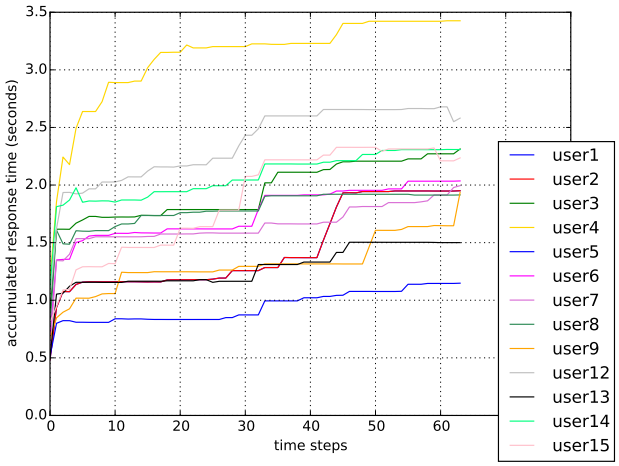

Fig. 11. accumulated response time vs time step for several users extracted from the main data base for $\bar{\lambda}_2 = [0.8573741847324399, 0.14262581526756013]$

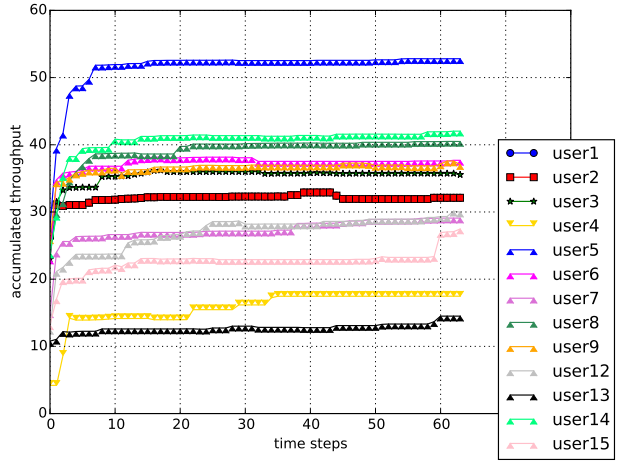

Fig. 12. accumulated throughout vs time step for several users extracted from the main data base for $\bar{\lambda}_2 = [0.8573741847324399, 0.14262581526756013]$

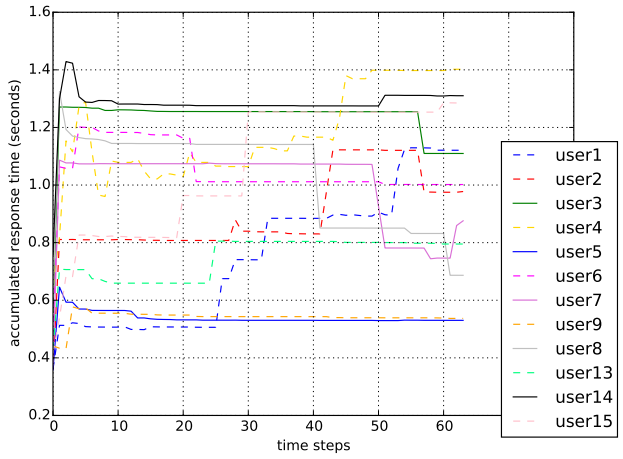

Fig. 13. accumulated response time vs time step for several users extracted from the main data base for $\bar{\lambda}_3 = [0.1696287781131175, 0.8303712218868825]$

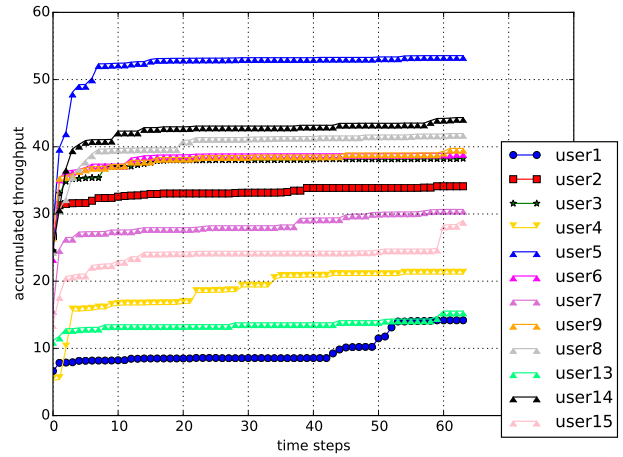

Fig. 14. accumulated throughout vs time step for several users extracted from the main data base for $\bar{\lambda}_3 = [0.1696287781131175, 0.8303712218868825]$