Interactive QoS-aware Services Selection for the Internet of Things

Pegah ALIZADEH[1], Aomar OSMANI[1], Mohamed Essaid Khanouche[2], Abdelghani Chibani[2], Yacine Amirat[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. We propose in this paper an MDP multi objective QoS optimisation process. The proposed multiple objectives MDP algorithm computes the optimal QoS coefficients and propose a data-driven decision for the best services workflow in real world QoS services datasets.

Index Terms—Computer Society, IEEE, IEEEtran, journal, LaTeX, paper, template.

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 paradigm [?], 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 [], or more generally as service orchestration problem [?]. Several papers are interested in this problem while they study a huge part of the related work in their papers [?], [?].

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. 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 concert 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 objective function to optimize. 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 [?], [?], [?], [?], meta-heuristic population based approches [], planning based approaches [?], [?], integer linear programming approaches [] and machine learning based approaches []. Or service selection methods using reputation models [?], [?], recommender system [?], [?]

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 [?] 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 priori knowledge. Section **??** summarizes experimental results.

2 RELATED WORKS

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 approach in this paper.

2.1 Recommender System based Approaches

[?] [?] [?]

2.2 Machine Learning based Approaches

2.3 Reinforcement Learning based Approaches

[?] [?]

The service composition approaches [?]

3 PROBLEM DESCRIPTION

Service oriented architecture is an open group standard. It provides services by application components trough a communication protocol and it is products and technologies independent [?]. A service is a an autonomous and platform independent computational entity, which can be described, published, descovered and dynamically assembled for developing distributed systems [?]. Main SOA properties are: each service must represents a business activity with specified outcome, may consists on underlying services and must be black box for users and self contain [?]. In SOA architecture, service can be a service provider, service broker or service consumer. Several implementation are proposed including web services based on WSDL [], Restfull HTTP [], Microsoft WCF [], Apache hrift [] and sorcer [].

The main SOA goal concern how to compose an applications including problems related to distribution, deployment and separately maintained services. In SOA, services uses metadata 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 [?] using quality of service parameters. There are several business standards related to the exact composition of a SOA [?], [?], [?], [?], [?], [?] however even in many situations an individual service is not able to solve complex requirement, one operational problem still the same: given a 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, ressources consume, ...) to well address a predefined problem ?

In this architecture, the W3C working group defines a service as a resource characterized by the abstract set of functionality that is provided. It is an abstract notion that must be implemented by a concrete agents [?]. In the rest of the paper, we will denote by concrete services the possible implementations of a W3C service. Instances of concrete services are executed over time to response end users needs. We will refer to user service to designate concrete service instance linked to a given user and when several service instances are executed over time, temporal references (i.e time step) must be given.

4 MOTIVATION FOR THE PRESENT STUDY A-T-ON DE QUOI REMPLIR CETTE SECTION?)

Expliquer l'objectif principal de cette famille de travaux
Donner un exemple significatif (non simulé montrant l'intérêt de ces travaux)
Expliquer ce que nous voulons apporter de plus ou de différent

5 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 $S_1, \ldots, S_n$. 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 execute several time over the time by the same actor. The well used time reference is time step. We denote by $S_{ij}^k(t)$ the service instance S_{ij} executed by the actor a_k at the time t[1].

Example 1. The execution line extracted from the real dataset used in our experiment will be denoted $S_{19994,3104}^{97}(5)$.

TABLE 1
Real dataset instance 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 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. These criteria are associated with each execution and can be also associated to users, to services or concrete services.

Let us consider $Q = \{q_1, \ldots, q_m\}$ the set of all possible criteria. As stated above it can be applied at all levels. Without loss of generality, in what follows we will reduce the formulas to the case of concrete services level. In this case, $q_l(S_{ij})$ will denote the quality value of the criteria q_l for the service S_{ij}. We suppose that criteria normalization step is done. To evaluate the concrete service global quality, weighted coefficients must be added in order to rank criteria preferences.

Find the optimal concrete service permutation according to a given evaluation function on temporal concrete services instances. This problem requires the resolution of a set of sub-problems among which:

-
-
-

As is shown in related work section, several works are done using various approaches. However some additional

1. A concret service S_{ij} executed by an actor a_k at time t is called *exectution*.

constraints make this problem more difficult. The main constraint concern 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 multi objective Markov decision precess approach.

6 MATERIAL AN METHODS

7 BACKGROUND : MULTI OBJECTIVE MARKOV DECISION PROCESS

8 PROPOSED SOLUTION

9 EXPERIMENTAL RESULTS

10 CONCLUSION

l

11 RELATED WORK

Essaid peux-tu reprendre ici le related work de ton papier QoS-aware

12 PROBLEMATIC

Internet Of Things (IOT) 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. 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?"*

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.

[?] finds a composition plan of abstract services by specifying the order of concrete services and rules for data transfer between these services. It means they have a sequential set of Abstract services that indicates an order on the set of activities and they are looking for the best concrete service in each abstract activity.

Invoking any abstract service produces several values for different QoS attributes such as response time, availability, cost, reliability and etc. They assume that the order of QoS attributes is given according to the user 's expectations. They propose an algorithm namely, *Energy-centered and QoS-aware Services Selection (EQSA)* that compute the optimal plan of service composition offering the QoS level required for user's satisfaction while minimizing the total energy consumption.

In this paper we are going to solve QoS-aware service composition without knowing the user preferred rank on the QoS attributes. Thus, **our proposed algorithm learns the user given weight on various QoS attributes while computing an optimal plan for the composite services selection. In this approach, we are allowed to query users in required situations. The final goal is to find the optimal plan by asking a few number of queries to the users on their preferences among QoS attributes.**.

In order to examine our algorithms experimentally, we propose several scenarios:

- **simulation scenarios [?]** :
  Without loss of generality, composite services considered in the simulation scenarios have a sequential structure. Other structures can be transformed into sequential structures using existing techniques [?]. The scenarios are generated by varying the number of Abstract services m, and the number of concrete services per class n. For each concrete service, three QoS attributes are evaluated: cost, availability, and reliability [?]. In this paper they generate data simultaneously:

- The availability and reliability are generated assuming a uniform distribution over the interval $[0.95, 0.99999]$.
- The cost of services is generated according to a uniform distribution over the interval $[10, 20]$.
- Fluctuations of the QoS values are considered as follows: at iteration $t+1$ of the selection process, the QoS value $qos^i_{q,j}(t+1)$ of each attribute is randomly chosen in the interval $[0.9qos^i_{q,j}(t), 1.1qos^i_{q,j}(t)]$ where $qos^i_{q,j}(t)$ represents the value of this attribute at time t.
- For the energy model they used the model described in [?]: the battery of each device has an initial charge value C_{initial}, chosen randomly in the interval $[0.7C_{\max}, 1.0C_{\max}]$, where $C_{\max}$ represents the maximum battery charge.
  This model has the advantage to consider services with different autonomy. Each invocation of a concrete service induces an average energy consumption. When a service is requested, a charge chosen randomly in the interval $[100\ mA.s, 10000\ mA.s]$ is subtracted from the actual battery charge of the device hosting the service. A device stops providing a service when a critical battery level $C_{\text{threshold}}$ is reached. The maximum battery capacity of a device is $1500mA$, whereas the critical battery level $C_{\text{threshold}}$ is set to 30% of the maximum battery charge.

- **More general scenario** :

  - Our proposed algorithm can be tested on the same model given in paper [?] while it can be implemented on the more general scenario. For instance, we can assume that abstract services do not have an ordinal structure and they can be connected to each other based on a given graph model.
  - Another assumption which is that, there is a probability distribution that indicates which abstract activity can be selected as a start activity.
  - The most interesting experimental parts are the test on the real data bases (we have some proposed data bases for this part.)

13 Motivation

I am going to give an example of the qos composite service which is modeled as a MDP. the goal is to give a better imagination of our approach to the users.

14 Problem Formulation

In this section, we describe the problem of QoS-aware service selection and the basic definitions relation to our approach. We utilize Markov Decision Process (MDP) concept to describe the service composition problem. MDPs are the suitable models for sequential decision problems such as QoS decomposition problem. We are looking for the optimal QoS-selection for various users while the user preferences on quality of services are unknown. Thus, our model is a partially known MDP model. Therefor, to tackle this problem, we use Vector-valued MDP (VMDP) to model multi-objective service composition under uncertainties. Before getting into details, it is required to describe some preliminary definitions as follows:

Definition 1. A **Concrete Service** cs_j is described by two parts: functional properties and non-functional properties.

- functional : cs_j is under the form of transaction function Action(cs_j) that takes an input data vector InputData(cs_j) to produce an output data vector OutputData(cs_j)
- non-functional : is defined by a QoS attributes vector QoS(cs_j) and the energy profile EProf(cs_j) Is it possible to add other g characteristics here? such as security .

Definition 2. An **Abstract Service** $AS_i = \{cs^i_1, \cdots, cs^i_n\}$ is a class of n 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.

14.1 Vector-valued Markov Decision Process

Regarding the provided technical database, invoking each concert service in a given time step t produces different quality of services. As an example, invoking concrete service cs^i at time t gives two different values for the response time and throughput: $QoS(cs^i) = (\text{rt}_t, \text{tp}_t)$.

Definition 3. Formally, a *Discrete-time Markov Decision Process (Discrete-time MDP)* [?] 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[2].
- 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: $\forall\, t \in \{1, \cdots, T\}\ d_t = d$.

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

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:

$$v_N^{\pi}(s) = r_N(s, \pi(s)) \;\; \forall s \in S_T \tag{1}$$

where S_T is the set of terminal states as a subset of all states S: $S_T \subset S$. Since S_T is a set of terminal states, there model should not make any action decision in these states. For the rest of time steps $t < T$, the value function is defined as:

$$v_t^{\pi}(s) = r_t(s, \pi(s)) + \gamma \sum_{s' \in S} P_t(s'|s, \pi(s)) v^{\pi}(s') \tag{2}$$

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

$$\pi \succeq \pi' \Leftrightarrow \forall s \in S \; v_0^{\pi}(s) \geq v_0^{\pi'}(s) \tag{3}$$

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

$$\pi^* \text{s.t.} \; \forall \; \pi, \; \pi^* \succeq \pi \tag{4}$$

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:

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

For the sake of simplicity, we define use a new notation for the Q-value function on state s and action a at time step t as, i.e. :

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

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

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

In this paper we are interested in discrete-time MDP with vector rewards. Sometimes, selecting an action in a given states returns back a vector value instead of a value. It means, by extending the discrete-time MDP to a discrete-time vector-valued MDP (discrete-time VMDP), we will have the modified following definition:

Definition 4. [?] A **discrete-time Vector-valued MDP (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.

14.2 MDP for Service Compositions

By modeling the service composition as a discrete-time VMDP, we will be able to find the best selected concrete services for any abstract activity by communicating with the agent and asking about her preferences on QoS attributes.

To solve the service composition problem without knowing anything about the user's preferences on the QoS attributes, we use discrete-time VMDP modeling to select the optimal concrete service in each abstract services w.r.t time stage t satisfying user's priorities. This service composition model can be called as discrete-time VMDP-Service Composition (discrete-time VMDP-SC) as the following. In the rest of this section, we assume any different type of MDP is a discrete-time MDP and will not rewrite it every time. The idea of this definition comes from Web Service Composition MDP (WSC-MDP) [?], [?], [?]

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 of the world.
- $SC(sa)$ is the set of available concrete services for the abstract service $sa \in SA$.
- $P_t(as'|as, sc)$ is the probability of invoking the concrete service sc in abstract activity as and resulting in the abstract activity as'.
- $\overline{QoS}_t : AS \times CS \longrightarrow \mathbb{R}^d$ is a reward function. The $\overline{QoS}(sa, sc)$ reward is the generated QoS vector value after invoking cs in as at time step t. Notice that d is the number of QoS attributes and we have $\overline{QoS}_t(sa, sc) = (qos_{1t}(sa, sc), \cdots, qos_{dt}(sa, sc))$.
- AS_T is the set of terminal services. It means the execution of the service composition terminates by arriving in one of these states.

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

Definition 6. A **policy service composition** $\pi : AS \longrightarrow SC$ is a function that defines which concrete service should be invoked in any 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 (is it correct?). Since reward values in MDP-SC are the QoS vectors for each concrete services, each policy should be evaluated with a vector function (see Equation 2):

$$\bar{v}_t^{\pi}(as) = \overline{QoS}_t(as, \pi(cs)) + \gamma \sum_{s' \in S} P(as'|\pi(as), as) \bar{v}_{t+1}^{\pi}(as') \tag{9}$$

By assumption, this function is called *QoS vector value 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 2. 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 3. 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:

$$QoS_t(as, cs) = \sum_{i=1}^{d} \bar{w}_i qos_{i_t} = \bar{w} \cdot \overline{QoS}(as, cs) \quad \forall t = 1, \cdots, N \quad (10)$$

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 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:

$$W = \{(w_1, w2, \cdots, w_d) \mid \sum_{i=2}^{d} w_i \leq 1 \text{ and } w_1 = 1 - \sum_{i=2}^{d} w_i\} \quad (11)$$

To compare QoS 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:

$$\bar{v}^a \succeq_P \bar{v}^b \Leftrightarrow \forall\, i\ a_i \geq b_i \quad (12)$$

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

$$\bar{v}^a \succeq_K \bar{v}^b \Leftrightarrow \forall\, \bar{w} \in W\ \bar{w} \cdot \bar{v}^a \geq \bar{w} \cdot \bar{v}^b \quad (13)$$

- 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:

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

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 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 [?], [?]. 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.

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. That means, the quality of all the invoked concrete services in time step T are zero. Since the MDP-SC objective is to find the policy that maximizes a measure of long-run expected QoSs, 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.

15 Interactive Reinforcement Learning Algorithms for the Service Compositions

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 ⟵ T
π_best ⟵ choose random policy
v̄_T(s_T) ⟵ (0,···,0)^3 ∀s_T at time T
𝒦 ⟵ set of constraints on Λ
while t ≥ 0 do
    t ⟵ t − 1
    for each h_t = (h_{t−1}, cs_{t−1}, as_t) ∈ H_t do
        best ⟵ (0,···,0)
        for each cs ∈ CS(as_t) do
            v̄_t(as_t) ⟵ QoS‾_t(as_t, cs) +
              Σ_{as'} P_t(as'|as_t, cs) v̄_{t+1}(as')
            ( best , 𝒦) ⟵ getBest(best, v̄_t, 𝒦)
            v̄_t(as_t) ⟵ best
            if best = v̄_t(as_t) then
                π_best(as_t) ⟵ 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

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
$(\bar{v}_{\text{best}}, \mathcal{K}) \longleftarrow \text{query}(\bar{v}, \bar{v}', \mathcal{K})$
return $(\bar{v}_{\text{best}}, \mathcal{K})$

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

Since the reward values depend on d qualities, the iterative algorithm first assign a zero vector to the set of states of the system at time T. At each iteration on the abstract services, the algorithm should solve equation 6, using the value from the previous iteration. Notice that $h_t = (h_{t-1}, cs_{t-1}, as_t)$ indicates by invoking a concrete service as_{t-1} at time $t-1$, the system will be at concrete service as_t. In the infinite horizon time, the iteration continues until the difference between successive values becomes extremely small, but in the finite horizon time the algorithm continues either this difference becomes small or the horizon time steps finish.

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 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 function **Best** (Algorithm 15). 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 15) where the user response to the comparison between the two given vectors, adds a new constraint to the W.

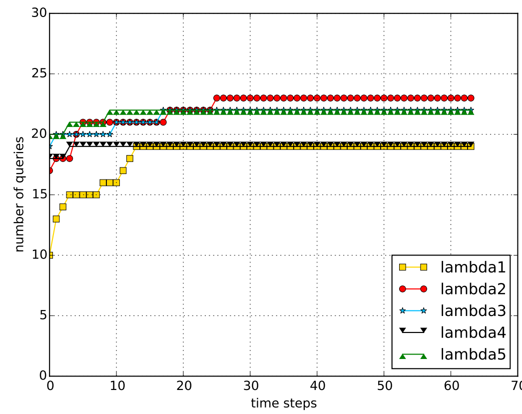

Fig. 1. this figure shows the number of queries proposed to the user during each time step.

Algorithm 15 finally finds the optimal policy/workflow 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.

TABLE 2
this tables indicates how each predicted policy from algorithm 15 is close to the optimal workflow when the $\bar{\lambda}$ 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

16 Performance Evaluation

We evaluate our methods on a public available dataset containing two parameters for quality of services: response time and throughput. These are the records between 339

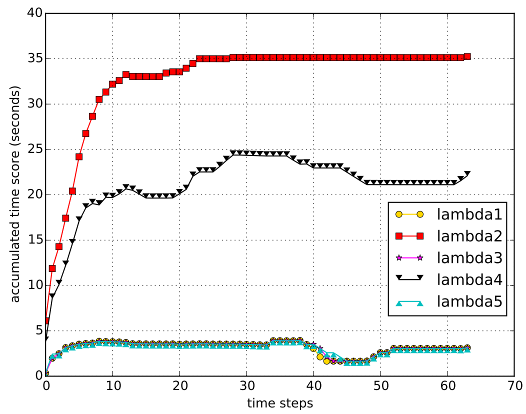

Fig. 2. this figures demonstrates how the accumulated response time increases during each time step. The lambda preferences are as follow: $\bar{\lambda}_1 = [0.07075913789991828, 0.9292408621000817]$, $\bar{\lambda}_2 = [0.8573741847324399, 0.14262581526756013]$, $\bar{\lambda}_3 = [0.1696287781131175, 0.8303712218868825]$, $\bar{\lambda}_4 = [0.6451844883834318, 0.3548155116165682]$ and $\bar{\lambda}_5 = [0.18190820427369447, 0.8180917957263055]$

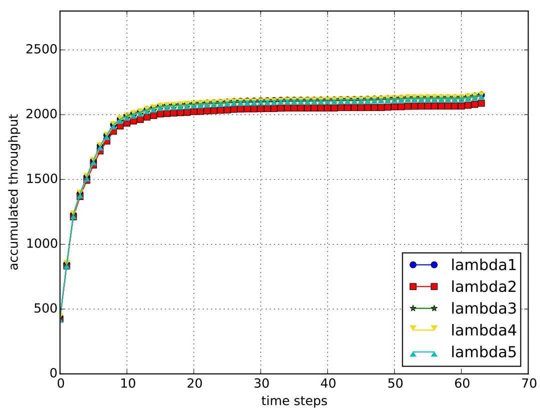

Fig. 3. this figures demonstrates how the accumulated throughout increases during each time step. The lambda preferences are as follow: $\bar{\lambda}_1 = [0.07075913789991828, 0.9292408621000817]$, $\bar{\lambda}_2 = [0.8573741847324399, 0.14262581526756013]$, $\bar{\lambda}_3 = [0.1696287781131175, 0.8303712218868825]$, $\bar{\lambda}_4 = [0.6451844883834318, 0.3548155116165682]$ and $\bar{\lambda}_5 = [0.18190820427369447, 0.8180917957263055]$

users and 5825 web services distributed worldwide [?]. 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.

16.1 Model DataSet as MOMDP

The main issue in implementing Algorithm 15 on any database is that how to model the given set as a multi-

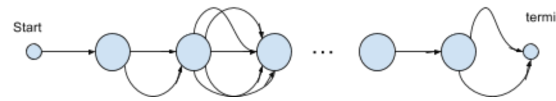

Fig. 4. A sequential form of abstract service connection

objective MDP. In the supported dataset [?] 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. That means at the end, each tested web services with 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 **??**) 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 16.1) the start sate ia an empty state which us connected to the first selected abstract services in the model. While the terminal state is an empty state for that indicate the the MDP is finite horizon. The probability transitions $P_t(as'|as, sc)$ is 1 if web service sc for abstract service as is available (according to our database) and abstract service as' is the next state in our selected sequential MDP model for all time steps $t = 0 \cdots 63$.
  2. For the parallel model (see Fig 16.1), the start and terminal states are the empty sates such that the start state has access to the 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, $Pt(as'|as, sc)$ is 1 if as' is the terminal state otherwise it is 0 for all time step $t = 0, \cdots, 63$.
- and the $\overline{QoS}_t$ function is build based on the extracted data on web services and their two qualities (response time and throughout).

16.2 Classified Dataset

16.3 Filtered Dataset

Our dataset is a real database generated by observing various users using nonvenomous number of web services. Then, all data inside database are not useful. After extracting all web

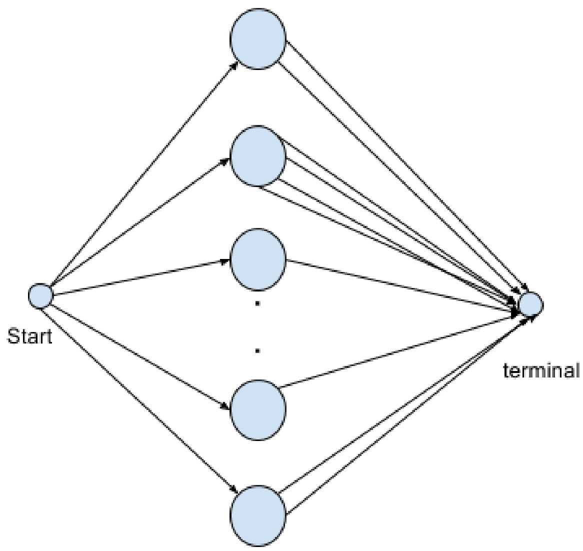

Fig. 5. A parallel form of abstract service connection

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 re 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.

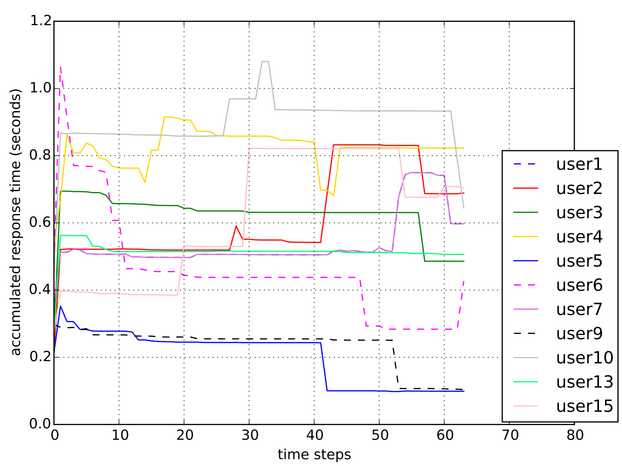

Fig. 6. accumulated response time vs time step for several users extracted from the main data base for $\bar{\lambda}_1 = [0.07075913789991828, 0.9292408621000817]$

Acknowledgments

The authors would like to thank...

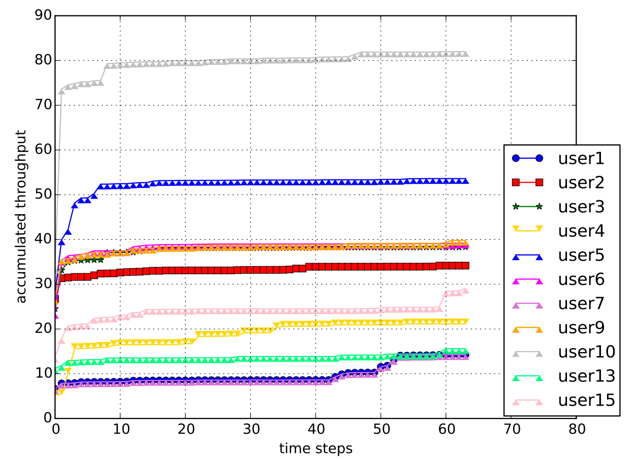

Fig. 7. accumulated throughout vs time step for several users extracted from the main data base for $\bar{\lambda}_1 = [0.07075913789991828, 0.9292408621000817]$

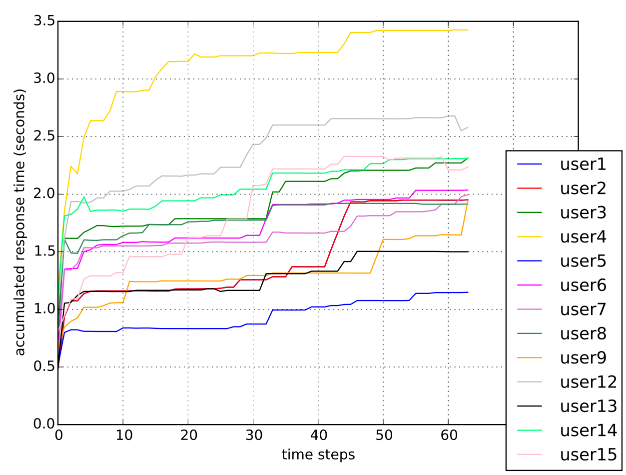

Fig. 8. accumulated response time vs time step for several users extracted from the main data base for $\bar{\lambda}_2 = [0.8573741847324399, 0.14262581526756013]$

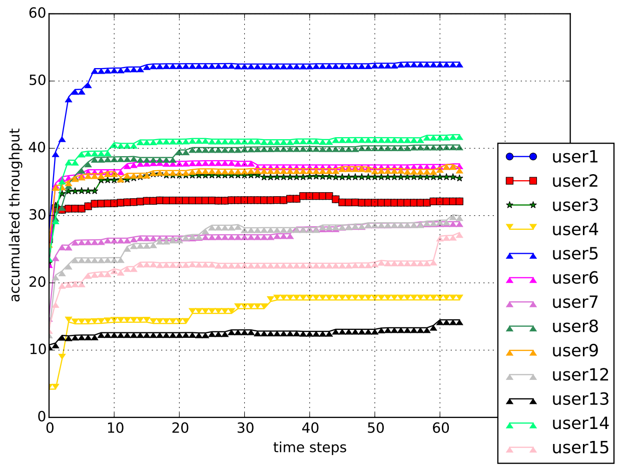

Fig. 9. accumulated throughout vs time step for several users extracted from the main data base for $\bar{\lambda}_2 = [0.8573741847324399, 0.14262581526756013]$

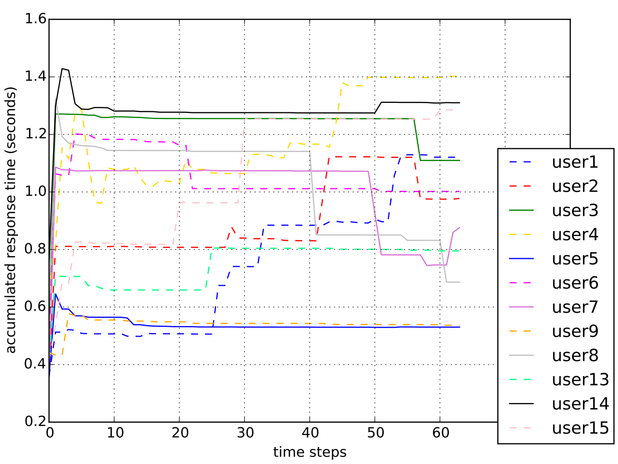

Fig. 10. accumulated response time vs time step for several users extracted from the main data base for $\bar{\lambda}_3 = [0.1696287781131175, 0.8303712218868825]$

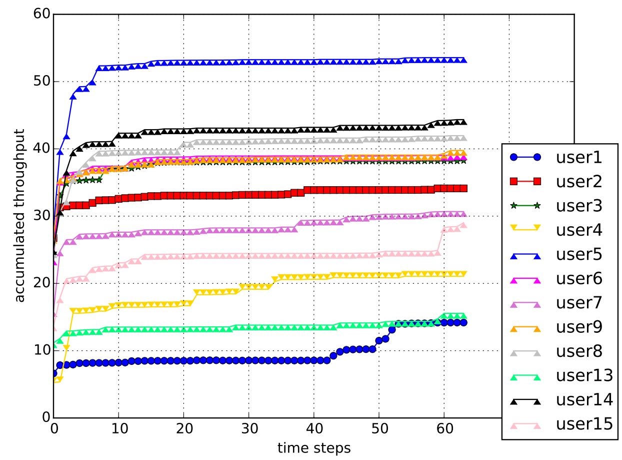

Fig. 11. accumulated throughput vs time step for several users extracted from the main data base for $\bar{\lambda}_3 = [0.1696287781131175, 0.8303712218868825]$