Interactive QoS-aware Services Selection for the Internet of Things

ABSTRACT

ACM Classification Keywords
H.5.m.

Author Keywords
Authors' choice; of terms; separated; by semicolons; include commas, within terms only; required.

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.

[4] 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[4]** :
  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 [2]. 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 [4]. 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 [3]: 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 $C_{\text{threshold}}$

CHI'16, May 07–12, 2016, San Jose, CA, USA

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

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 [4] 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.)

PROBLEM FORMULATION

In this section, we describe the problem of QoS-aware service selection and its relation to our approach. We utilize Markov Decision Process (MDP) concept to describe the service composition problem. We use Vector-valued MDP (VMDP) to model multi-objective service composition under uncertainties. First, it is required to describe abstract and concrete services model.

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** $AS_i = \{cs_1^i, \cdots, cs_n^i\}$ 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.*

Vector-valued Markov Decision Process

Formally, a *Markov Decision Process (MDP)* with reward value is defined as follow:

DEFINITION 3. *A **Markov Decision Process (MDP)** [6] is defined by a tuple (S, A, P, r, γ) where:*

- *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: $P(s'|s, a)$ encodes the probability of going to state s' when the agent is in state s and chooses action a.*
- *Reward Function: $r : S \times A \longrightarrow \mathbb{R}$, quantifies the utility of performing action a in state s.*
- *Discount Factor: $\gamma \in [0, 1)$ indicates how less important are future rewards in compared to the immediate ones.*

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 $v^\pi : S \longrightarrow \mathbb{R}$ which is defined as expectation of sum of rewards w.r.t the policy π:

$$v^\pi(s) = r(s, a) + \gamma \sum_{s' \in S} P(s'|\pi(s), a) v^\pi(s') \qquad (1)$$

Therefore, the preference relation among policies is defined as below:

$$\pi \succeq \pi' \Leftrightarrow \forall s \in S \; v^\pi(s) \geq v^{\pi'}(s) \qquad (2)$$

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. :

$$\pi^* \text{s.t.} \; \forall \, \pi, \; \pi^* \succeq \pi \qquad (3)$$

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

$$v^*(s) = \max_{a \in A} r(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v^*(s') \qquad (4)$$

By extending the MDP to a vector-valued MDP (VMDP), we will have the modified following definition:

DEFINITION 4. *[1] A **Vector-valued MDP (VMDP)** is defined by a tuple $(S, A, P, \bar{r}, \gamma, \beta)$ where the vector-valued reward function $\bar{r}$ is defined on $S \times A$ and $\bar{r}(s, a) = (r_1(s, a), \cdots, r_d(s, a)) \in \mathbb{R}^d$ is the vector valued reward defined by $\bar{r}$ in (s, a).*

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

MDP for Service Compositions

By modeling the service composition as a 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 VMDP modeling to select the optimal concrete service in each abstract services satisfying user's priorities. This service composition model can be called as VMDP-Service Composition (VMDP-SC) as:

DEFINITION 5. *(the idea of this definition comes from Web Service Composition MDP (WSC-MDP) [5, 7]) A* ***VMDP-Service Composition (VMDP-SC*** *is a tuple* $(AS, CS(.), P, \bar{r}, AS_T, \gamma)$*, where (We define* $\gamma = 1$ *because our models are horizontally finite.)*

- *AS is a finite set of abstract services of the world.*

- $SC(sa)$ *is the set ([4] used "class" concept instead of set?) of available concrete services for the abstract service* $sa \in SA$*.*

- $P(as'|as, sc)$ *is the probability of invoking the concrete service sc in abstract activity as and resulting in the abstract activity* as'*.*

- $\overline{QoS} : AS \times CS \longrightarrow \mathbb{R}^d$ *is a reward function that indicates the value of concrete service* $cs \in CS(s)$ *after invoking in abstract service as.* $\overline{QoS}(sa, sc)$ *reward is the generated QoS vector value after invoking cs in as. Notice that d is the number of QoS attributes and we have* $\overline{QoS}(sa, sc) = (qos_1(sa, sc), \cdots, qos_d(sa, sc))$*.*

- AS_T *is the set of terminal services. It means the execution of the service composition terminates by arriving in one of these sates.*

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 1):

$$\bar{v}^{\pi}(as) = \overline{QoS}(as, \pi(cs)) + \gamma \sum_{s' \in S} P(as'|\pi(as), as)\bar{v}^{\pi}(as') \tag{5}$$

By assumption, this function is mentioned as a QoS vector value function (we may change the name later.).

Now, comparing two workflows/policies boils down to comparing two vectors. The optimal workflows satisfying various users with different presences among the QoS attributes are not the same. Thus, a model presenting user preferences and importance over the QoS attributes is required. For this reason, we define user preferences over the QoS attributes as a linear combination of these attributes and their given weights to each attribute as bellow.

$$QoS(as, cs) = \sum_{i=1}^{d} \bar{w}_i qos_i = \bar{w} \cdot \overline{QoS}(as, cs) \tag{6}$$

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:

$$W = \{(w_1, w2, \cdots, w_d) \mid \sum_{i=2}^{d} w_i \leq 1 \text{ and } w_1 = 1 - \sum_{i=2}^{d} w_i\} \tag{7}$$

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:

$$\bar{v}^a \succeq_P \bar{v}^b \Leftrightarrow \forall\, i\ a_i \geq b_i \tag{8}$$

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

$$\bar{v}^a \succeq_K \bar{v}^b \Leftrightarrow \forall\, \bar{w} \in W\ \bar{w} \cdot \bar{v}^a \geq \bar{w} \cdot \bar{v}^b \tag{9}$$

- 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:

$$\begin{cases} \min\ \bar{w} \cdot (\bar{v}^a - \bar{v}^b) \\ \text{subject to}\ \ \bar{w} \in W \end{cases} \tag{10}$$

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.

INTERACTIVE REINFORCEMENT ALGORITHMS SERVICE COMPOSITION

The basic general algorithm will be explained here [8]. The algorithm will be modified with some optimization and heuristic methods in order to be implementable on huge size models.

Assume the problem is modeled as a VMDP-Sc. We present Algorithm 3 to find the optimal workflow (policy) satisfying user preferences.(it should be modified by considering the terminal services SA_T and finite horizon MDPs.)

Data: VMDP-SC($AS, A(), P, \bar{r}, \gamma$), a W polytope of user weights on objectives, precision ϵ
Result: The optimal service selection policy for the given user.
$t \longleftarrow 0$
$\pi_{\text{best}} \longleftarrow$ choose random policy
$\forall s\ \bar{v}_0(s) \longleftarrow (0, \cdots, 0)$ zero vector of dimension d[1]
$\mathcal{K} \longleftarrow$ set of constraints on Λ
repeat
  $t \longleftarrow t+1$
  for ***each*** $as \in AS$ **do**
    best $\longleftarrow (0, \cdots, 0)$
    for ***each*** $cs \in A(as)$ **do**
      $\bar{v}_t(as) \longleftarrow$
      $\overline{QoS}(as, cs) + \sum_{as'} P(as'|as, cs)\bar{v}_{t-1}(as')$
      (best , $\mathcal{K}$) $\longleftarrow$ getBest(best, $\bar{v}_t, \mathcal{K}$)
      $\bar{v}_t(as) \longleftarrow$ best
      if *best* $= \bar{v}_t(as)$ **then**
        $\pi_{\text{best}(as) \longleftarrow cs}$
      end
    end
  end
until $||\bar{v}_t - \bar{v}_{t+1}|| \leq \epsilon$;
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 $\mathcal{K}$
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 $(\bar{v}_{\text{best}}, \mathcal{K})$

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

REFERENCES

1. Pegah Alizadeh, Yann Chevaleyre, and François Lévy. 2016. Advantage based value iteration for Markov decision processes with unknown rewards. In *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*. 3837–3844.
2. Jorge S. Cardoso, Amit P. Sheth, John A. Miller, Jonathan Arnold, and Krys Kochut. 2004. Quality of service for workflows and web service processes. *J. Web Sem.* 1, 3 (2004), 281–308.
3. Jason Flinn and M. Satyanarayanan. 1999. Energy-aware Adaptation for Mobile Applications. *SIGOPS Oper. Syst. Rev.* 33, 5 (Dec. 1999), 48–63.
4. Mohamed Essaid Khanouche, Yacine Amirat, Abdelghani Chibani, Moussa Kerkar, and Ali Yachir. 2016. Energy-Centered and QoS-Aware Services Selection for Internet of Things. *IEEE Trans. Automation Science and Engineering* 13, 3 (2016), 1256–1269.
5. Ahmed Moustafa and Minjie Zhang. 2013. Multi-Objective Service Composition Using Reinforcement Learning. In *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings*. 298–312.
6. Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.
7. Hongbing Wang, Xuan Zhou, Xiang Zhou, Weihong Liu, Wenya Li, and Athman Bouguettaya. 2010. *Adaptive Service Composition Based on Reinforcement Learning*. Springer Berlin Heidelberg, 92–107.
8. Paul Weng and Bruno Zanuttini. 2013. Interactive Value Iteration for Markov Decision Processes with Unknown Rewards. In *IJCAI '13 - Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, Beijing, China, 2415–2421. https://hal.archives-ouvertes.fr/hal-00942290