Towards the Use of Quality of Service Metrics in Reinforcement Learning: A Robotics Example

J. F. Inglés-Romero[1], J. M. Espín[1], R. Jiménez[2], R. Font[1] and C. Vicente-Chicote[3]

[1] Biometric Vox S.L., Spain
[2] Infomicro Comunicaciones S.L., Spain
[3] University of Extremadura, QSEG, Escuela Politécnica de Cáceres, Spain
juanfran.ingles@biometricvox.com

Abstract. Service robots are expected to operate in real-world environments, which are inherently open-ended and show a huge number of potential situations and contingencies. This variability can be addressed applying reinforcement learning, which enables a robot to autonomously discover an optimal behavior through trial-and-error interactions with the environment. The process is carried out by measuring the improvements the robot achieves after executing each action. In this regard, RoQME, an Integrated Technical Project of the EU H2020 RobMoSys Project, aims at providing global robot Quality-of-Service (QoS) metrics in terms of non-functional properties, such as safety, reliability, efficiency or usability. This paper presents a preliminary work in which the estimation of these metrics at runtime (based on the contextual information available) can be used to enrich the reinforcement learning process.

Keywords: Reinforcement Learning, Quality of Service, RoQME.

1 Introduction

With the advance of robotics and its increasingly growing use in all kinds of real-world applications, service robots are expected to operate (at least safely and with a reasonable performance) in different environments and situations. In this sense, a primary goal is to produce autonomous robots, capable of interacting with their environments and learning behaviors that allow them to improve their overall performance over time, e.g., through trial and error. This is the idea behind *Reinforcement Learning* (RL) [1], which offers a framework and a set of tools for the design of sophisticated and hard-to-engineer behaviors.

In RL, an agent observes its environment and interacts with it by performing an action. After that, the environment transitions to a new state providing a reward. The goal is to find a policy that optimizes the long-term sum of rewards. One of the fundamental problems of RL is the so-called cursing of the objective specification [1]. Rewards are an essential part of any RL problem, as they implicitly determine the desired behavior. However, the specification of a good reward function can be highly complex. This is, at least in part, because it requires to accurately quantify the rewards, which does not fit well with the natural way people express objectives.

The RoQME Integrated Technical Project (ITP), funded by EU H2020 RobMoSys Project [2], aims at contributing a model-driven tool-chain for dealing with system-level non-functional properties, enabling the specification of global Robot Quality of Service (QoS) metrics. RoQME also aims at generating RobMoSys-compliant components, ready to provide other components with QoS metrics. The estimation of these metrics at runtime, in terms of the contextual information available, can then be used for different purposes, e.g., as part of a reward function in RL.

Integrating QoS metrics in the rewards can enrich the learning process by extending the quality criteria considered, for example, with non-functional properties, such as user satisfaction, safety, power consumption or reliability. Moreover, RoQME provides a simple modeling language to specify QoS metrics, in which qualitative descriptions predominate over quantitative ones. As a result, RoQME limits the use of numbers, promoting a more natural way of expressing problems without introducing ambiguity in its execution semantics.

This paper presents a work in progress towards the use of QoS metrics in RL. Santa Bot, a "toy" example in which a robot delivers gifts to children, will help us illustrate the problem in simple terms.

The rest of the paper is organized as follows. Section 2 describes the Santa Bot example. Section 3 introduces the RoQME modeling language. Section 4 shows some simulation results for the Santa Bot example. Section 5 reviews related work and, finally, Section 6 draws some conclusions and outlines future work.

2 Santa Bot: an illustrative example

This section introduces the example that will be used throughout the paper to illustrate the proposed approach. The goal is to present the reinforcement learning problem in simple terms to explain the role that the RoQME QoS metrics can play. The example takes place in a shopping mall where a robot, called Santa Bot, distributes gifts to children. In the area set up for this purpose, a number of children waits in line to receive a gift from Santa Bot. Each child has a (finite) list of wishes, containing his/her most desired toys in order of preference. Unfortunately, these lists were made in secret and are unknown to Santa Bot. Santa Bot will try to guess the best gift for each child to meet their expectations and thus maximize their joy.

2.1 Formalizing the example scenario

Let us consider a queue of M children waiting to receive a gift, i.e., $child_1$, $child_2 \dots child_M$. As the order prevails, $child_i$ will receive a gift after $child_{i-1}$ and before $child_{i+1}$. Moreover, we identify all the different types of toys with natural numbers, i.e., *Toys* = {1, 2, 3…, K}. At time *t*, the Santa Bot bag contains a number of instances of each toy k, denoted by $N_k^t \in \{0,1,\dots,N_k^0\}$, being N_k^0 the initial amount. As gifts are delivered, the number of instances of a toy decreases, remaining at 0 when the toy is no longer available, therefore, $N_k^t \geq N_k^{t+1}$. In this scenario, the action of

Santa Bot is limited to deciding what gift is given to each child. Listing 1 specifies the effect of this action.

```
Being $child_i$ the child to receive a gift at time $t$
deliver gift $k$ to $child_i$
    [pre-condition]   $N_k^t > 0$
    [post-condition]  $N_k^{t+1} = N_k^t - 1$
```

Listing 1. Specification of the delivery action in the example scenario.

Although Santa Bot can adopt different strategies for delivering the gifts in its bag, the best approach will be the one that maximizes children satisfaction. In this case, satisfaction is associated with the ability to fulfill the children's wishes, expressed in their wish lists. Thus, we consider that each child has a wish list represented by a *n*-tuple $(a_1, a_2, \dots, a_n)$, where entries are toy identifiers ($a_i \in Toys$), and show uniqueness ($\forall i, j \in \{1,2,\dots,n\}, a_i = a_j \Leftrightarrow i = j$) and order ($a_i$ is preferred over a_j if and only if $i < j$). Moreover, the function $WL(child_i) = (a_1, a_2, \dots, a_n)$ links children to wish lists, such that $WL(child_1) = (2, 6, 1)$ indicates that the first child in the queue wants toys 2, 6, and 1, in order of preference.

Equation 1 shows a possible definition of satisfaction for $child_i$ receiving a toy *j*. This function provides a score that determines the goodness of a decision, so the higher its value the better.

$$S(child_i, j) = \sum_{\forall a_k \in WL(child_i)} C(k) \cdot \delta(j - a_k) \tag{1}$$

Being C a decreasing positive function and δ the Kronecker delta function, i.e., $\delta(x) = 1$ when x=0, otherwise it is 0. It is worth noting that Equation 1 produces 0 if the decision does not match any option in the wish list, otherwise it increases as the selected gift has a higher position in this list. Finally, the result of the problem is the entire sequence of decisions made for all the children, i.e., $d = (d_1, d_2, \dots, d_M)$, where d_i indicates the toy delivered to $child_i$. Equation 2 shows the overall satisfaction considering the complete sequence of decisions *d*.

$$S_d = \sum_{\forall i} S(child_i, d_i) \tag{2}$$

2.2 The reinforcement learning problem

Santa Bot poses an optimization problem whose optimal solution would be feasible using integer linear programming if all wish lists were known. However, since this is not the case, the robot is expected to autonomously discover the optimal solution through trial-and-error interactions with its environment. In *Reinforcement Learning* (RL) [1], an agent observes its environment and interacts with it by performing an action. After that, the environment transitions to a new state providing a reward. The goal of the algorithm is to find a policy that optimizes the long-term sum of rewards.
The main elements of a RL problem (states, transitions, actions and rewards) are usually modeled as a *Markov Decision Process* (MDP) [3]. Fig. 1 shows the basic MDP

specification for the Santa Bot example. It is worth noting that, for the sake of simplicity, we will not delve into the details of RL and MDP.

The Santa Bot environment considers two sets of states: *In-front* and *Leaving*. The former indicates that a child is in front of Santa Bot waiting for a gift. In this situation, the state is defined in terms of the gifts available (i.e., N_k^t) and some observable features of the child. In the example, we have supposed that the robot can perceive the apparent age, the gender and the predominant color of the child's clothes. Ideally, these features will include sufficient information to trace preferences and common tastes among children with similar aspects. Actually, a successful learning process should be able to detect these tendencies and exploit them by making good decisions.

Once the robot performs the delivery action, the environment transitions from *In-front* to *Leaving*. Note that it returns to *In-front* when a new child arrives. *Leaving* integrates the satisfaction of the child with the gift, which is represented by a QoS metric. This metric provides a real value in the range [0,1] indicating how much the child liked the gift (being 1 the highest degree of satisfaction). The reward function will depend on this value, e.g., see Equation 3, where the reward changes in a linear way from 0 to α according to the satisfaction.

$$reward = \alpha \cdot satisfaction,\ \alpha \in \mathbb{R}^+ \tag{3}$$

The reward function is an essential part of any RL problem, as it implicitly determines the desired behavior we want to achieve in our system. However, it is very difficult to establish a good reward mechanism in practice. Note that Equation 3 seems simple because we have moved the complexity to the specification of the QoS metric, i.e., to how the robot measures satisfaction. The following section illustrates how RoQME can alleviate the complexity of specifying rewards by supporting the definition of QoS metrics.

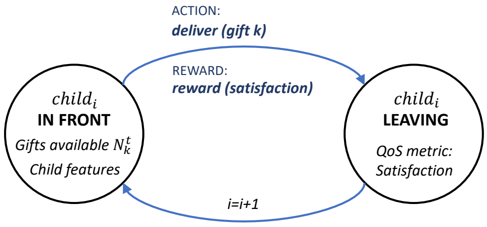

Fig. 1. Simple Markov Decision Process for the Santa Bot example.

3 Modeling robot QoS metrics

RoQME aims at providing robotics engineers with a model-driven tool-chain allowing them to: (1) specify system-level non-functional properties; and (2) generate RobMoSys-compliant components, ready to provide other components with QoS metrics defined on the previous non-functional properties. In the following, we use the Santa Bot example to present the main modeling concepts of RoQME and how they are translated into QoS metrics at runtime. More information about the RoQME meta-model and its integration into RobMoSys can be found in [4].

The previous section left open the specification of the QoS metric for measuring the satisfaction of a child after receiving a gift (hereinafter, simply denoted as satisfaction). It is worth noting that QoS metric is not a modeling concept in RoQME, but rather, a runtime artifact implicitly bound to a non-functional property. Non-functional properties, which can be thought of as particular quality aspects of a system, are included in the modeling language, thus, in our example, satisfaction is modeled as a non-functional property using the keyword `property` (see line 4 in Listing 2). Regarding the execution semantics, a RoQME model abstracts a *Belief Network* [3], in which properties are represented by unobserved Boolean variables. In this sense, the variable associated with satisfaction would indicate whether or not Santa Bot is optimal in terms of this property. The runtime quantification of this belief results in the corresponding QoS metric value. For example, a resulting value of 0.67 can be understood as the probability of the gift being satisfactory for the child.

The belief of a property (i.e., the QoS metric) fluctuates over time according to the evidences observed by the robot in its environment (contextual information). RoQME allows specifying observations (`observation`) as conditions in terms of context variables (`context`), so that the detection of an observation will reinforce (or undermine) the belief. In the belief network, observations are evidence variables that exhibit a direct probabilistic dependence with the property.

Lines 5-8 in Listing 2 show four observations for the Santa Bot example. These observations use the following context variables: (1) *face*, which indicates the facial expression of the child after receiving a gift; and (2) *age*, the apparent age of the child perceived by the robot. Note that the robot will continuously feed RoQME with this contextual information. Each observation will reinforce or undermine the child satisfaction according to the emotion expressed by the child. We have assumed that surprise and anger are stronger emotions than joy and sadness, and thus the first ones should have a higher influence on satisfaction than the second ones. Moreover, observations 1 and 4 are conditioned by age, which means that strong reactions tend to be more or less frequent depending on the age. Note that toddlers may tend to react more vividly than school-aged children. Therefore, it is used to normalize the effect of the observation among children of different ages.

```
1  context face : eventType {JOY, SURPRISE, NEUTRAL, SADNESS, ANGER}
2  context age : enum {TODDLER, PRESCHOOLER, SCHOOL_AGED , ADOLESCENT}
3  context prevSatisfaction : number
4  property satisfaction : number prior prevSatisfaction
5  observation obs1 : SURPRISE reinforces satisfaction highly conditionedBy age
6  observation obs2 : JOY reinforces satisfaction
7  observation obs3 : SADNESS undermines satisfaction
8  observation obs4 : ANGER undermines satisfaction highly conditionedBy age
```

Listing 2. A simple RoQME model for modeling children satisfaction.

Finally, the context variable *prevSatisfaction* provides the satisfaction of the child who received the gift just before the current one. We want to use this information to model possible influences between two consecutive children, e.g., a child expressing anger could have an effect on the behavior of the following child. This is implemented by defining a bias in the prior probability of satisfaction (see line 4).

As we have already mentioned, a RoQME model translates all its semantics into a belief network. Fig. 2 shows the qualitative specification of the network resulting from the model in Listing 2. Note that, for the sake of clarity, we have omitted probabilities. This belief network will be the "brain" of a generated RobMoSys-compliant component aimed at measuring satisfaction. In general, the generated component will estimate the value of each non-functional property, specified in the RoQME model, by successively processing the available contextual information, either from internal (e.g., robot sensors) or external (e.g., web services, other robots, etc.) sources. The contextual information received by the component will be sequentially processed by: (1) a context monitor that receives raw contextual data and produces context events; (2) an event processor that searches for the event patterns specified in the RoQME model and, when found, produces observations; and, finally (3) a probabilistic reasoner that computes a numeric estimation for each metric. This information could then be used by other components, e.g. the robot task sequencer could integrate the RL process to adapt the robot behavior according to the provided QoS metrics.

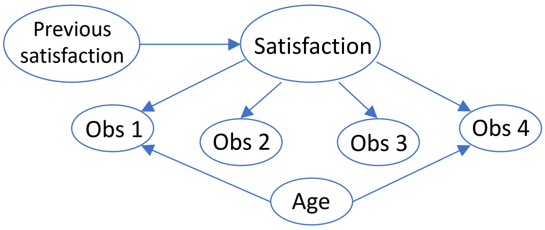

Fig. 2. Belief network resulting from the RoQME model developed in Listing 2.

4 QoS metrics in action

This section presents the simulation results obtained on the Santa Bot example. Before detailing the results, let us describe the simulation setting.

Queues of children. We have generated random queues of length 50 with children showing different features (i.e., age, gender and clothes color). In particular, we have considered four age groups: (1) toddlers, (2) preschoolers, (3) school-aged children and (4) adolescents, with the following density function [0.4, 0.3, 0.2, 0.1], and uniformly distributed gender and clothes colors (5 colors).

Wish lists. For each child, we have produced a 3-item list from 20 different toys. Preferences were distributed depending on the children features (age, gender and clothes color). These dependencies create tendencies that are expected to be detected and exploited by the learning process. The correlation is clearly exposed in Fig. 3, where the heat map represents favorite toys in relation to children features.

Children reactions. As we described in Section 3, RoQME estimates satisfaction considering as contextual information the age and the face expression of the children.

While the former has already been defined, the children reactions need to be established. For that, we have assigned face expressions to each child receiving a particular gift according to his/her features and wish list. The idea is to introduce inclinations similar to those described in Section 3. It is obvious that the algorithm will not be able to learn if the RoQME model for satisfaction does not reflect reality.

Simulations. We have executed the simulation on 1000 episodes, where each episode consists of a new queue of 50 children. In addition, we have considered that Santa Bot has an infinite number of gifts available for each type of toy. The left side of Fig. 4 shows the learning process in an initial state (episode 25), in which the exploration of the states (i.e., choosing a random action) is preferred to acquire new information and discover the best actions. This can be seen in the upper heat map, where Santa Bot delivers gifts following a uniform approach. As for the cumulative rewards represented in the lower heat map, it begins to show the correlations we have introduced in the data (see Fig. 3). On the other hand, the right side of Fig. 4 shows the learning process in an advanced state (episode 1000), in which the exploitation (i.e., choosing the best action according to the information already learned at that moment) is preferred over exploration. The upper heat map shows how the process seems to prioritize the gifts that have provided greater reward. As for the lower map, it is similar to Fig. 3, which means the learning process was successful.

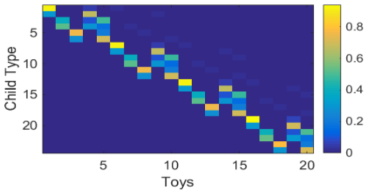

Fig. 3. Heat map of gift probabilities by child type.

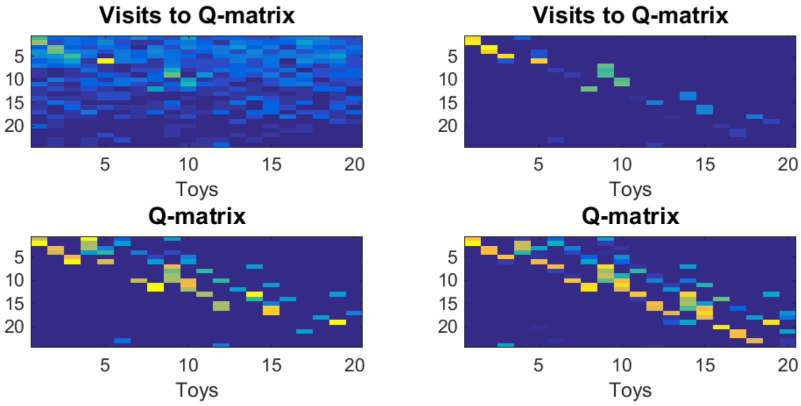

Fig. 4. Up, the number of visits to the Q-matrix cell; down the Q-matrix. (Left) Episode 25 of the learning algorithm. (Right) Episode 1000 of the learning algorithm.

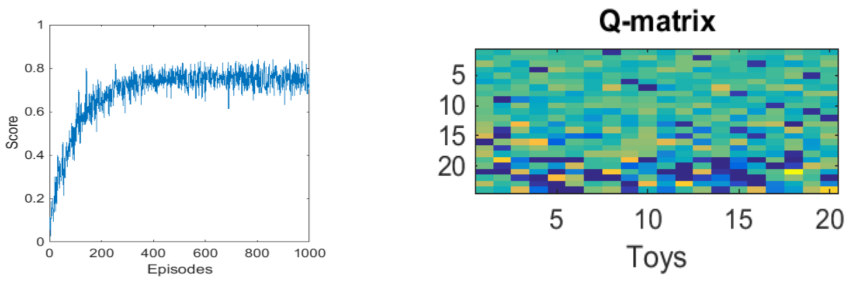

Fig. 5. (Left) Score evolution over 1000 episodes. (Right) Q-matrix after 1000 episodes, in which the RoQME model estimates satisfaction wrongly.

The left side of Fig. 5 shows the evolution of the learning process, in which the score seems to stabilize after 400 episodes. Finally, the system has achieved an average score of 72.03% and a maximum score of 84.67% with respect to the optimal solution (the one with known wish lists). Finally, we have modified the RoQME model to observe the effect of modeling wrongly. The right side of Fig. 5 shows that, in this case, the process fails to learn.

5 Related work

Reinforcement learning emerged as a combination of optimal control (using dynamic programming) and trial-and-error learning (inspired by the animal world) [5]. In the last years, thanks to more powerful computing systems and new deep learning techniques, RL has received an impulse in domains such as video games and simulations [6]. In the field of robotics, optimization problems have a temporal structure, where RL techniques seem to be suitable. However, there may be cases showing high dimensionality continuous states and actions, with states not fully observable and noise-free. Those cases generally result in modeling failures that can be accumulated over time, so training with physical agents is needed [1].

In the literature, we can find numerous RL techniques applied to diverse robotic tasks. For example, robot manipulators aimed at learning how to reach a certain position or open a door [7-9]; or mobile robots that learn how to move in crowded spaces [10]. Recently, RL has been used to teach a car how to drive in a short period of time with just a camera and feedback of speed and steering angle [11].

Despite the large number of applications, one of the fundamental problems of RL is the so-called cursing of the objective specification [1]. Rewards are a crucial part of any RL problem, as they implicitly determine the desired behavior we want to achieve. However, the specification of a good reward function can be highly complex. In this sense, domain experts may be required to define the proper reward function.

To alleviate this problem, different techniques have been used to define reward functions. A perfect scenario would be a human supervising the agent and providing feedback about the reward that the system should give according to its actions, but this is expensive and error prone. In [12], authors use EEG-signals of a human user as reward function, so that the user unconsciously teaches the agent to act as he wants.

Another way is, firstly, to train a learner to discover which people actions are better than others, and then use it in RL to give reward to the agent simulating a person. Regarding the perspective applied in [13], it creates a Bayesian model based on feedback from experts. On the contrary, in [14], non-expert people are considered to train the learner with reinforcement learning techniques.

Some rules defined at design time in a lax way can be introduced in the learning process as a bias to enhance the behavior as in [10], where the robot has to respect some social rules of circulation.

Regarding QoS, in [15] the system tries to autonomously improve the quality of the services of the robot by adapting its component-based architecture and applying RL to meet the user preferences. The system learns how to estimate the non-functional properties in the process as it has not prior knowledge about them.

6 Conclusions and future work

This paper presents a preliminary work about the integration of RoQME QoS metrics into the reward strategy of RL problems. Moreover, we have introduced and formalized Santa Bot, an optimization "toy" example inspired by Santa Claus, used to illustrate the explanations in simple terms. In the following, we highlight some remarks:

- The execution semantics of a QoS metric relies on a belief network, which is a well-known mathematical abstraction that has been successfully applied to many domains, such as medical diagnosis and natural language processing. Consequently, we can benefit from existing tools and techniques that are used for the analysis and simulation of probabilistic networks.
- The RoQME modeling language allows users to transparently specify the qualitative part of the underlying belief network (i.e. nodes and arcs of the directed acyclic graph). The RoQME framework is in charge of automatically completing the quantitative part of the network (i.e., the conditional probability tables). As quantification is often referred to as a major obstacle in building probabilistic networks [2], RoQME eases the modeling process by abstracting probabilities. In this sense, although there are many probabilistic programming languages [16] that can be used to specify belief networks, unlike RoQME, they usually need a detailed specification of probabilities.
- Although the specification of RoQME QoS metrics does not need to be addressed by domain experts, a RoQME model that does not sufficiently represent reality will have a great impact on the learning process.
- We have simulated the Santa Bot example considering an unlimited number of gifts. Although this relaxation of the problem has not affected the explanations, it is pending to take more advantage of the example and to apply our approach to more realistic robotics scenarios.

For the future, we plan to continue exploring the potential of RoQME QoS metrics applied to RL. We also intend to study ways of improving the QoS modeling process.

Acknowledgements

RoQME has received funding from the European Union's H2020 Research and Innovation Programme under grant agreement No. 732410, in the form of financial support to third parties of the RobMoSys project.

References

1. Kober, J., Bagnel, J. and Peters, J.: Reinforcerment learning in robotics: A survey. The international Journal of Robotics Research 0(0), 1-37, (2013)
2. RobMoSys website, http://robmosys.eu, last accessed 2018/07/24.
3. Russel, S. and Norvig, P.: Artificial intelligence: A modern approach. 3er edn. Upper Saddle River, NJ, USA: Prentice Hall Press, (2009)
4. Vicente-Chicote, C., Inglés-Romero, J.F. and Martinez, J.: A Component-Based and Model-Driven Approach to Deal with Non-Functional Properties through Global QoS Metrics. 5th International Workshop on Interplay of Model-Driven and Component-Based Software Engineering (ModComp). Copenhagen, Denmark (2018)
5. Sutton, R. and Barto, A.: Reinforcement learning: an introduction. 1st edn. The MIT Press, Cambridge, Massachusetts, USA, (2017)
6. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing Atari with Deep Reinforcement Learning, NIPS Deep Learning Workshop 2013.
7. Gu, S., Holly E., Lillicrap T. and Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. 2017 IEEE International Conference on Robotics and Automation (ICRA), 3389-3396, (2017)
8. Kalakrishnan, M., Ludovic R., Pastor P. and Schall S.: "Learning force control policies for compliant manipulation." 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (2011): 4639-4644.
9. Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y, and Levine, S.: Collective robot reinforcement learning with distributed asynchronous guided policy search. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017): 79-86.
10. Chen, Yu Fan, Everett, M., Liu, M. and How, J.P.: Socially aware motion planning with deep reinforcement learning. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017): 1343-1350.
11. Kendall, A., Hawke, J., Janz, D., Mazur P., Reda D., Allen J.M. and Lam, VD.: Learning to Drive in a Day, arXiv preprint arXiv:1807.00412
12. Iturrate, I., Montesano, L. and Minguez, J.: Robot reinforcement learning using EEG-based reward signals. 2010 IEEE International Conference on Robotics and Automation (2010): 4822-4829.
13. Wilson, A., Fern, A. and Tadepalli, P.: A Bayesian Approach for Policy Learning from Trajectory Preference Queries. NIPS (2012).
14. Christiano, P. F., Leike,J. Brown, T., Miljan M., Shane L. and Amodei, D.: Deep Reinforcement Learning from Human Preferences. NIPS (2017).
15. Wang H., Zhou X., Zhou X., Liu W., Li W. and Bouguettaya A.: (2010) Adaptive Service Composition Based on Reinforcement Learning. In: Maglio P.P., Weske M., Yang J. and Fantinato M. (eds) Service-Oriented Computing. ICSOC 2010. Lecture Notes in Computer Science, vol. 6470. Springer, Berlin, Heidelberg.
16. Probabilistic-programming.org, http://probabilistic-programming.org/wiki/Home, last accessed 18/08/2018.