The AAAI-17 Workshop on
Human-Machine Collaborative Learning
WS-17-11

Using Options to Accelerate Learning of New Tasks According to Human Preferences

**Rodrigo Cesar Bonini, Felipe Leno da Silva,
Edison Spina, Anna Helena Reali Costa**
Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil
{rodrigo_cesarb,f.leno,spina,anna.reali}@usp.br

Abstract

Over the years, people need to incorporate a wider range of information and multiple objectives for their decision making. Nowadays, humans are dependent on computer systems to interpret and take profit from the huge amount of available data on the Internet. Hence, varied services, such as location-based systems, must combine a huge quantity of raw data to give the desired response to the user. However, as humans have different preferences, the optimal answer is different for each user profile, and few systems offer the service of solving tasks in a customized manner for each user. Reinforcement Learning (RL) has been used to autonomously train systems to solve (or assist on) decision-making tasks according to user preferences. However, the learning process is very slow and require many interactions with the environment. Therefore, we here propose to reuse knowledge from previous tasks to accelerate the learning process in a new task. Our proposal, called *Multiobjective Options*, accelerates learning while providing a customized solution according to the current user preferences. Our experiments in the *Tourist World Domain* show that our proposal learns faster and better than regular learning, and that the achieved solutions follow user preferences.

Introduction

In recent years, human activities have impacted several branches and gradually have been modifying our behavior and needs. For the development of any modern society, good quality information technology is necessary to support and ensure constant progress, especially assisting in routine and recurring tasks, providing the necessary knowledge for decision-making as fast as possible. Understanding and identifying the dimensions of human preferences for decision-making is a key component of any modern recommender system.

Nowadays, computer systems aim at supplying this growing demand for quick and good information through varied services one. These systems must take into account different preferences and restrictions, and solve tasks such as "find a free ambulance that can get to a traffic collision place in less than fifteen minutes". Typical applications of location-based systems include navigation services (Smith et al. 2004), tourist information systems (Hinze and Voisard 2003), emergency response, and disaster management (Erharuyi and Fairbairn 2003). However, these services do not always take into account user-preferences, because they are limited in their ability to evaluate alternative decisions and solve multiple and conflicting objectives when solving tasks. For example, a Travel Recommendation System may receive the user query "find me a cheap yet comfortable hotel in San Francisco, CA". While it is easy to find the cheapest hotel in the city, this place may fall outside the definition of *comfortable* for most users. Therefore, the system must balance the *cheap* and *comfortable* metrics according to the current user's preferences, and the optimal answer will be different for another user.

For this kind of functionality, it is necessary to integrate some kind of *Multi Criteria Decision Making* (MCDM) technique (Triantaphyllou 2013). MCDM is a decision support methodology based on the idea that humans use various decision criteria to determine the best solution for a problem. MCDM tecnhiques range from simple additive weighting criteria to more sophisticated methods (Rinner and Malczewski 2002). Agichtein *et al* (2008) conducted a work in which community-provided answers to questions from varied topics were evaluated through the use of MCDM in conjunction with Data Mining techniques. User preferences were described by numeric weights (that can be learned through observation of the user behavior), and the top ranked answers achieved the same effectiveness of expert-provided answers (for the current user). But, in this way the systems need information from users and experts.

A way to support decision making without the need of a specialist is through the use of Reinforcement Learning (RL) (Sutton and Barto 1998), which is a framework that can be used to solve tasks when the computer (agent) is expected to learn autonomously how to solve its tasks. The main idea of RL algorithms is that the agent follows a trial and error procedure and learns by observing the outcome of interactions with the environment. At each decision step, the current state is observed, an action that affects the environment is chosen and applied by the agent, and finally the agent observes how much that action helped to the task completion through a reward function. Under certain conditions, an agent can learn how to optimally solve tasks by executing this process multiple times.

RL has been successfully applied in many complex problems (Tesauro 1995; Seo and Zhang 2000; Ng et al. 2006; Singh et al. 2002; Mnih et al. 2015), showing strong evidence that it can be very effective for decision-making on issues of information, because it can work autonomously. For example, Seo and Zhang (2000) proposed a RL system to increase the chance of success of customized internet searches. User preferences are estimated from the history of searches, keywords, and number of visits to particular sites.

Consider the following scenario to understand the synergy between the techniques of interest: A traveler is in an unknown city and decided to extend his stay. It is late at night and the traveler needs to find a hotel. With the currently used location-based systems it is possible to find hotels near traveler's position. However, the traveler wants the hotel that best meets his preferences, with a reasonable price for the room, private bathroom, and late checkout time. All of these criteria are subjective and therefore have varied importance levels for different travelers. Still, the system should be able to give the optimal recommendation for the current traveler.

The main problem when applying RL is that the classical algorithms learn very slowly through interactions of trial and error type, taking a very long time, since RL classical approach needs many steps to explore the state-action space. The problem is further intensified when we describe it by using multiple reward functions (which is the case for MCDM). The extension of the classical RL paradigm, Multiobjective Reinforcement Learning MORL (Van Moffaert, Drugan, and Nowé 2013) solves tasks with multiple reward functions $\{R_1, \dots, R_i\}$ by balancing all objectives as well as possible, considering the criteria informed by humans. As domains become progressively complex over the years, scalability gains more importance for these methods.

Transfer Learning(TL) (Taylor and Stone 2009) is one of the solutions proposed to accelerate learning in RL tasks. The option-based solutions (Sutton, Precup, and Singh 1999) allow to reuse previously acquired knowledge in new tasks, generalizing and transferring knowledge between domains, agents, and tasks, consequently accelerating learning in RL tasks. For instance, it is relatively easier for a traveler travel to a mountain if he/she has been in another similar mountain in other part of the world before.

The *Options Framework* (Sutton, Precup, and Singh 1999) was proposed to alleviate scalability issues through the use of options, that are high-level actions that encapsulate a partial solution, often representing the solution of a subproblem. For example, in a travel by car between two cities, an option could be the trip until half way to fuel the car. While in the original *Options Framework* each option was provided by a domain expert, later works like the *PolicyBlocks Algorithm* (Pickett and Barto 2002), propose autonomous option-discovery methods through the evaluation of previous task solutions. Furthermore, learned options are reported to accelerate learning even when transferred across similar problems. However, to the best of our knowledge our proposal is the first work to provide option-based methods that can be used in multiobjective problems.

Thus, the idea here is to allow a user to define his/her criteria of interest for each application task, and to use RL to learn new solutions according to his/her preferences and to accelerate learning by taking into account previous knowledge in other tasks. Thereunto, we here propose a method hereafter called Multiobjective Options (MO-Opt) to learn options in MORL tasks in order to accelerate learning, arguing that MORL algorithms can also benefit from option-based and TL solutions to accelerate learning according to humans preferences.

Foundation and Related Work

Before describing our proposal, we define the concepts which form the foundation of our work. We first introduce the relevant basic concepts of RL. Then we present relevant concepts about MORL. Finally, we describe the extension of RL to Options.

Reinforcement Learning

Many sequential decision-making problems may be modeled by a Markov Decision Problem (MDP) (Puterman 1994), and RL is a extensively used solution for MDPs. In the RL framework (Sutton and Barto 1998), an agent learns by interacting with an environment over a series of discrete time steps. In each decision step k, the agent observes the current state s_k and applies an action a_k. Then, the next state is defined by a transition probability function (unknown by the agent in learning scenarios) and the agent receives a reward r_k. The agent goal in an MDP is to learn an optimal policy π^* that maps each state to the actions that leads to the greatest expected cumulative sum of rewards.

Formally, an MDP is composed of $< S, A, T, R >$, where:

- S is a finite set of possible environment **states**.
- A is a finite set of **actions** that can be executed by the agent.
- $R : S \times A \times S \to \mathbb{R}$ is the **reward function** that maps the agent actuation to a numerical reward.
- $T : S \times A \times S \to [0, 1]$ is the **state transition function**, where $T(s_k, a_k, s_{k+1})$ defines the probability of a state transition from s_k to s_{k+1} after the execution of a_k at k.

We are here interested in learning problems where T and R are unknown to the agent, which can only observe the current state and the reward signal.

As the output from the transition and reward functions cannot be predicted in learning problems, the MDP can be solved through interactions with the environment, what can be accomplished by the Q-Learning algorithm (Watkins and Dayan 1992). Q-Learning iteratively learns a Q-table, that aims to estimate the cumulative discounted reward associated to each state-action pair: $Q : S \times A \to \mathbb{R}$. At each decision step Q is updated following:

$$\begin{aligned} Q_{k+1}(s_k, a_k) \leftarrow (1 - \alpha) Q_k(s_k, a_k) + \\ \alpha[r_{k+1} + \gamma \max_a Q_k(s_{k+1}, a)] \end{aligned} \tag{1}$$

where $r_{k+1} = R(s_k, a_k, s_{k+1})$, $0 < \alpha \leq 1$ is the learning rate and γ is the discount factor.

Q-Learning eventually converges to the true Q function: $Q^*(s,a) = E\left[\sum_{i=0}^{\infty} \gamma^i r_i\right]$, and Q^* can be used to define an optimal policy as:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{2}$$

RL also shows strong evidence that it can be very effective in decision-making, because it can provide solutions automatically and in real time. For example assisting in the identification of public interest areas, informing traffic accidents and hospitals (van Treeck and Ebner 2013). RL has also been used to optimize solutions that increase the chance of success in internet searches in accordance with characteristics of each person, obtaining as much information as possible about a user profile (Seo and Zhang 2000).

However, the standard Q-Learning may be inefficient in environments with large state spaces, as it may need many steps to reach the convergence. Furthermore, the standard Q-Learning only takes into account single-objective tasks, but many RL tasks cannot be easily described by a single reward function, and are rather described by multiple reward functions. Thereupon, an approach to help solving this kind of problem is detailed in the following.

Multiobjective Reinforcement Learning

A single reward function cannot easily describe some tasks, as they require the maximization of multiple, usually conflicting objectives (Liu, Xu, and Hu 2015; Ngai and Yung 2011). Multiobjective decision-making has long been recognized as an important research subject (Van Moffaert, Drugan, and Nowé 2013), which indeed led to the development of RL techniques to multiobjective problems. MORL algorithms learn policies that optimize a vector of reward functions, rather than a single one.

Thus, Multiobjective MDPs (MOMDPs) only diverge from regular MDPs with respect to the reward function, that is now composed of i **objectives**. Consequently, MOMDPs can be solved by learning a Q-table for each one of the objectives (using, e.g., Equation (1)), resulting in a vector of Q-tables:

$$\boldsymbol{MQ}(s,a) = [Q^1(s,a), Q^2(s,a), \ldots, Q^i(s,a)]^T \tag{3}$$

where $Q^i(s,a)$ is the Q-table learned using reward values from objective i. Depending on the scenario, the solution of an MOMDP can be either multiple policies that approximate the Pareto Front (Roijers et al. 2014) or a single policy that tries to balance all objectives (Khamis and Gomaa 2014).

As a single policy that maximizes all objectives simultaneously usually does not exists (specially when conflicting objectives are included in the MOMDP), the concept of an optimal policy is not clear in MORL. Nonetheless, the user (or designer) domain knowledge can be used to define a *scalarization function*, that projects $\boldsymbol{MQ}$ to a single numeric value:

$$SQ^f(s,a) = f(\boldsymbol{MQ}(s,a)), \tag{4}$$

where $f : [\mathbb{R}, \ldots, \mathbb{R}]^T \rightarrow \mathbb{R}$ is the *scalarization function*. A linear combination of the Q-values is a widely used *scalarization function* (Ngai and Yung 2011; Silva and Costa 2015; Zeng et al. 2010), which is defined as:

$$f(\boldsymbol{MQ(s,a)}, \boldsymbol{w}) = \sum_{i=1}^{i} w_i Q^i(s,a) \tag{5}$$

where $\boldsymbol{w}$ is a vector with hand-crafted specified weights according to the humans preferences in a problem. After an adequate scalarization function, a policy can be defined using SQ (Equation (4)) instead of a standard Q-table(Equation (1)).

MORL may be applied in many real-world tasks, and all of them would be benefited from faster RL algorithms. Therefore, our goal is to accelerate learning in such tasks. MORL achieved interesting results and can be used in a variety of domains (Khamis and Gomaa 2014; Ngai and Yung 2011; Zeng et al. 2010; Silva and Costa 2015; Brys et al. 2014). However, these approaches suffer from scalability issues and present slow learning, since solving problems with several and possibly conflicting objectives can often be computationally and sample expensive and take a long time. Thus, these issues hamper the studies and development of new techniques and new approaches in this topic.

Options

The *Options Framework* (Sutton, Precup, and Singh 1999) was proposed to alleviate RL scalability issues. Options extend the usual notion of actions, providing closed-loop partial policies for taking actions over a certain period of time. Options are high-level actions that aim at decomposing MDPs in subtasks and solving them, providing temporally extended courses of actions. Some examples of options may include a car passing in determined locations, a traveler going to a distant city or unlocking a door, in other words, tasks that require a certain number of primitive actions to achieve a desired subgoal.

An option for an MDP is a conditional sequence of primitive actions defined as a three-tuple, $\{\ \pi, \mathcal{I}, \beta\ \}$, consisting of a **policy** ($\pi : S \rightarrow A$), a set of **initiation states** ($\mathcal{I} \subseteq S$), and a **termination condition** ($\beta : S \rightarrow [0,1]$) (Sutton, Precup, and Singh 1999). The initiation set $\mathcal{I}$ is the subset of the state space in which the option can be executed, i.e, an option is available in state s_t iff $s_t \in \mathcal{I}$. When the option initiation condition is satisfied and the agent selects it, the policy is followed until a termination condition is met. The termination condition β is a probability function over states that defines the likelihood with which the agent ends the execution of an option when it is in that state. It may also be a time limit, that terminates deterministically when the execution of the option policy reaches a certain number of episodes parametrized by designer (Bernstein 1999). Then, when an option ends, the agent has the chance to select another option, and this process repeats over time, until a goal state is reached. Primitive actions are a special case of option that always lasts exactly 1 step.

We use $Q(s,o)$ to be the expected return given that the agent starts in state s and takes option o. Thus, the concept of an action-value function generalizes to an option-value function. When an option terminates, its value is changed according to the maximum option-value at the resulting state and the cumulative discounted reward obtained during its execution:

$$Q_{k+1}(s_k, a_k) \leftarrow (1-\alpha)Q(s_k, o_k)+ \\ \alpha\left[r_k + \gamma^t \max_{o \in O_{s_{t+k}}} Q_k(s_{t+k}, o) - Q_k(s_k, o_k)\right] \quad (6)$$

The idea behind using options in RL comes from the simple fact that the probability of an agent to behave in a certain way should be proportional to how often that behavior was successful in the long-dated past (Thrun and Schwartz 1995; Bowling and Veloso 1998; Bernstein 1999; Mcgovern and Barto 2001).

In general, the aforementioned algorithms, were proposed to work along specific kinds of problems, not being able to solve general problems with the same quality. The *PolicyBlocks Algorithm* is an interesting options-discovery algorithm due to its versatility and good results in very different tasks (Pickett and Barto 2002).

Algorithm 1 PolicyBlocks (2002)

```
1: $M \leftarrow$ a given set of single-objectives tasks M
2: $n \leftarrow$ desired number of options
3: use RL or DP to create a solution set L for M
4: $O \leftarrow$ empty option set
5: $C \leftarrow$ empty option candidate set
6: $L \leftarrow$ set of source policies $\pi^*$ learned for M
7: while $|L| > 0$ and $|O| < n$ do
8:     for each element $l$ of the power set of L do
9:         $c \leftarrow mrg(l)$
10:        $C \leftarrow C \cup c$
11:    end for
12:    $c^* \leftarrow arg \max_{c \in C} score(c)$
13:    $O \leftarrow O \cup c^*$
14:    for each element $l$ of the power set of L do
15:        $subtract(l, c^*)$ //subtract $c^*$ from $l$
16:    end for
17: end while
18: return $O$
```

The *PolicyBlocks Algorithm* (described in algorithm 1) is decomposed in a three step process: (1) Firstly it generates a set of option **candidates** by finding where the sample solutions match, in other words, here occurs the **merge process** between a pair of partial policies, ($mrg(\pi_1 \cap \pi_2 \cap \cdots, \pi_n)$). (2) After that, it **scores** the candidates based on the match and then chooses the highest scoring option. (3) Finally, **subtracts** this option from a source set of optimal policies L. Then, we have a set of options o that have been learned with the algorithm and can be used to accelerate the learning. However, the *PolicyBlocks Algorithm* is single-objective and unable to solve multiobjective problems and even more, obey the user preferences or priorities.

Hence, our proposed approach aims at discovering options for TL to accelerate learning in MORL problems.

Multiobjective Options

Most of previously proposed option-based methods are focused on single-objective problems (Thrun and Schwartz 1995; Bowling and Veloso 1998; Bernstein 1999; Mcgovern and Barto 2001). Notwithstanding, MORL approaches attained good results in tasks that have more than one objective, pondering tasks and these objectives (Khamis and Gomaa 2014; Ngai and Yung 2011; Silva and Costa 2015). However, the state-of-the-art option-based methods do not work in such multiobjective scenarios.

Aiming at accelerating learning in MORL problems, we here introduce MO-Opt, an approach based in the *Options Framework* for options-discovery and for TL with fixed state variables and actions (Taylor and Stone 2009) in multiobjective tasks according to humans preferences. The main idea of MO-Opt is firstly to learn options for each of the objectives separately (for which the *PolicyBlocks* algorithm, for example, can be used) and apply these options in the multiobjective problem, and secondly transfer the obtained knowledge to a new and different tasks, reusing the acquired knowledge in previous tasks. The learned options are intended to optimize a single objective, but may maximize the reward function of multiple objectives for some situations, or guide the agent towards trajectories which prioritize one objective without hampering the others, according to a human setting. The RL algorithm is able to identify when each option is useful, and we argue that these options can accelerate the learning process, guiding the agent's exploration to learn a given task faster.

There are many TL methods that can transfer knowledge between MDPs where the type of knowledge transferred can be primarily characterized by its specificity. We here assume that the state and action spaces are fixed for the source and target tasks, but the state transition function may vary (Taylor and Stone 2009).

Algorithm 2 describes our proposal. Firstly we initialize a set of options ϕ. Then, we learn a set of optimal policies L_i for each objective $i \in \theta$ by using a standard RL algorithm, where θ is the set of objectives. After that, we use the standard *PolicyBlocks Algorithm* (any other options-discovery algorithm could also be used) to learn a set of options for each objective separately (including them in ϕ as common options, i.e. partial policies, to accelerate the learning of the new task). Then, the Multiobjective learning algorithm is executed in the new tasks using $\phi \cup A$, to finally learn the policy for the MORL domain.

Algorithm 2 MO-Opt for Transfer Learning

```
1: $\phi \leftarrow \emptyset$
2: for each objective $i \in \theta$ do
3:     learn in $h$ episodes a set of policies $L_i$ for $i$
4:     $\phi \leftarrow \phi \cup PolicyBlocks(L_i, n, M)$
5: end for
6: transfer options to a target task $M_{target}$
7: run MORL in the new task using $\phi \cup A$
```

Experimental Evaluation

Problem Domain

We evaluated our proposal in a task designed to show how to make the machine provide a route more quickly satisfying the preferences of a particular traveler through learning from past experiences, representing situations where MO-Opt can solve a MORL problem through the use of options. In order to evaluate the effectiveness of our proposal, we implemented the *Tourist World Domain* (illustrated in Figure 1), where we have a tourist (represented by a car) and *points of interest* (represented by the statues of liberty), in which the tourist can pass or not according to her preference. The main challenge in this problem is to maximize the amount of *points of interest* at the same time minimizing the time (number of steps) until the tourist reaches her goal, but always respecting the tourist will. Notice that *points of interest* are often outside the optimal path towards the goal position, then the agent must balance the two objectives and reason over the objective to be pursued at a certain moment.

In order to evaluate our proposal, we also implemented another instance(task) of the *Tourist World Domain* (Figure 2), implemented using BURLAP (MacGlashan 2015), in which we expect that the agent will learn faster through the reuse of knowledge from the first map.

In these tasks, the action set of the car is $A = \{north, south, east, west\}$, and a *point of interest* visit is counted when the agent is in the same position of it. Posterior visits on the same spot will not be counted again, and a *point of interest* visit awards a reward of +1 in the second reward function. Besides that, a reward of +1 is awarded in the first reward function when the agent achieves the goal position. Otherwise, the reward is 0 for both objectives. In the case of the agent perform some action that would hit a wall, she remains at the same state, and episodes end when the agent reaches the goal position.

Experiments

Firstly, we performed in the Task 1 (see Figure 1) of our *Tourist World Domain* 5 iterations of the Q-learning algorithm for each objective separately, with each iteration during 100 episodes, each one providing an optimal policy to *PolicyBlocks*, which chooses for each objective the 3 best scored options between the power set of the 5 initially obtained, as indicated by (Pickett and Barto 2002).

Then, we applied the learned options to a different task (Figure 2), executing 100 learning episodes using the *Scalarized Q-Learning algorithm* (Silva and Costa 2015) twice: the first time with a standard Q-Learning without options; and the second using MO-Opt, in order to evaluate the relative effectiveness of the learned options to the learning process.

In the first scenario, the scalarized algorithm was configured with the weights w_1 = 0.75 and w_2 = 0.25 to simulate a tourist in a hurry to get to its goal without caring too much to go through *points of interest*. Then, we evaluate the scenario w_1 = 0.5 and w_2 = 0.5, where a touristic is not in a hurry but cannot expend too much time, and finally w_1 = 0.25 and w_2 = 0.75, where the tourist is willing to visit as many places of interest as possible. The reward functions are combined using a linear combination of the aforementioned weights(equation 4), where the w_1 represents the weight of the objective 1 and w_2 the weight of the objective 2. We also set the learning rate α = 0.2 and the discount rate γ = 0.9. The observed performances are averages over 100 executions of this procedure.

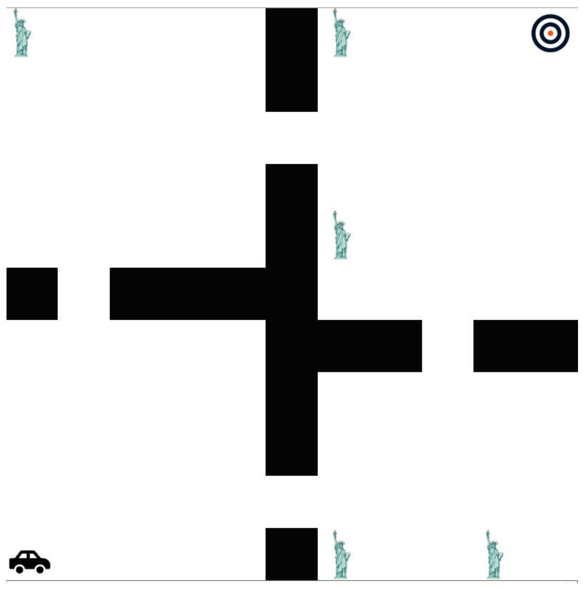
Figure 1: Task 1 of *Tourist World Domain* for learn options in Multiobjective Reinforcement Learning. The tourist aims at passing through all points of interest (represented by statues of liberty) in the environment while minimizing its amount of steps until it reaches its goal (represented by the target).

Results and Discussion

We here presents the results of our experiments

Scenario I - Tourist in a Hurry

Figure 3 shows the average cumulative reward achieved by our proposal and the plain scalarized Q-learning in this scenario. The cumulative rewards codify the agent performance for that user preference, where the agent is expected to achieve a higher reward as fast as possible. MO-Opt learns faster at the beginning of the learning process, achieving an average reward of 0.16 after roughly 20 learning episodes, while Q-Learning did not achieve this result until the training ended. We observed that, in this experiment, the car usually visited 2 *points of interest* (that were closer to the fastest path), prioritizing the amount of steps.

The difference between the average cumulative reward indicates that MO-Opt provides both a better jump-start and asymptotic performance in this scenario.

Scenario II - Balanced Situation

Figure 4 shows the performances in the second scenario. MO-Opt was again better than Q-Learning, but because of the greater importance to the objective 2, the car spent more

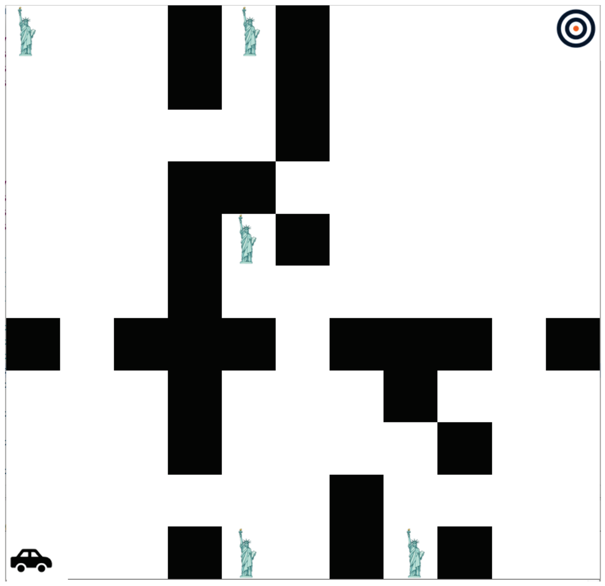
Figure 2: Task 2 of *Tourist World Domain* with the use of options learned from task 1. The tourist aims at passing through all points of interest (represented by statues of liberty) in the environment while minimizing its amount of steps until it reaches its goal (represented by the target)

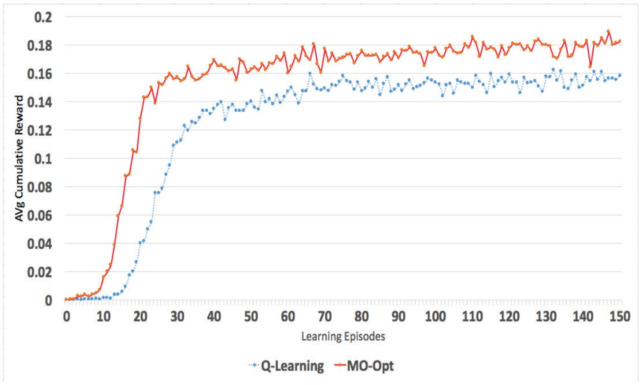

Figure 3: The average cumulative reward (linear combination of the two objectives) for 150 episodes after the transfer learning process from the task 1 to the task 2 with $w_1 = 0.75$ and $w_2 = 0.25$.

steps to learn how to prioritize both objectives. In this scenario, the car usually visited 3 *points of interest*, aiming at visit more points without delaying the travel too much.

Again, the final performance of our algorithm was slightly better.

Scenario III - Unhurried Tourist

Finally, Figure 5 shows the results of the last scenario. In order to visit as many points of interest as possible, the car spent more steps until reaching the goal. In average, 4 *points of interest* in average were visited in this scenario.

In this scenario, we can see that our algorithm again

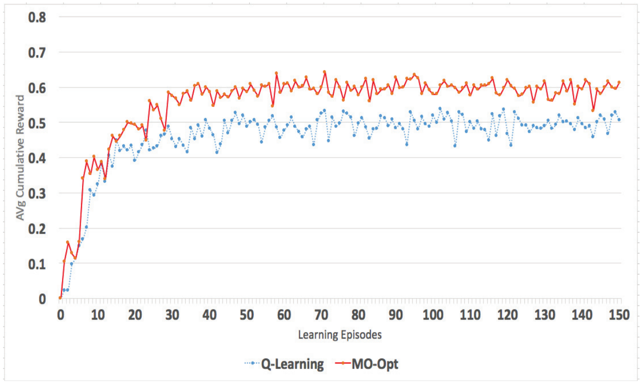

Figure 4: The average cumulative reward (linear combination of the two objectives) for 150 episodes after the transfer learning process from the task 1 to the task 2 with $w_1 = 0.5$ and $w_2 = 0.5$.

learned faster and better.

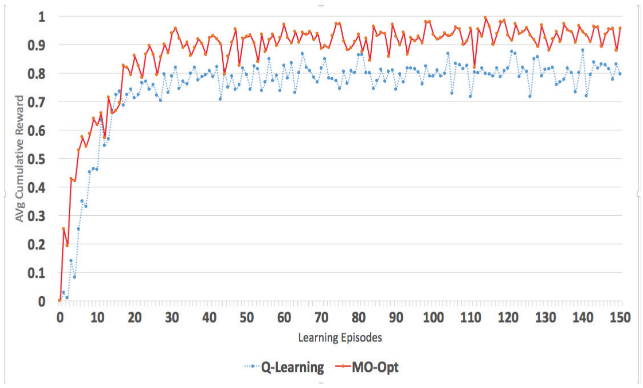

Figure 5: The average cumulative reward (linear combination of the two objectives) for 150 episodes after the transfer learning process from the task 1 to the task 2 with $w_1 = 0.25$ and $w_2 = 0.75$.

As a conclusion for our experiments, the asymptotic function was better for MO-Opt in all cases, reaching a higher average reward earlier than the standard Q-Learning with all evaluated cases.

These results show that our approach can be successfully used to accelerate MORL learning according to human preferences, without breaking them and without having in performance.

Conclusion and Further Works

We here proposed the MO-Opt option-discovery algorithm to accelerate learning in MORL tasks, while obeying user preferences. The option-based method helped the machine to give a solution faster while providing solutions according to multiple human preferences.

Our experiments in the *Tourist World Domain* showed that our options-discovering method outperformed regular learning by succesfully transferring knowledge from similar tasks taking into account several user profiles. This result shows that our approach works faster than classical RL techniques and can be adapted according to user preferences.

MO-Opt is a promising algorithm which allows knowledge reuse and generalization, thus accelerating learning in multiobjective tasks. The next step of our research is to investigate some methods to estimate weights (human preferences) without user direct inputs. An adaptation of the W-Learning (Liu, Xu, and Hu 2015) algorithm might be a way to do this.

In this way, our approach may be used to set criterion-based priorities (such as importance, relevance, urgency, personal enjoyment) to properly consider specific attributes to each task. We believe that it may also works for support decision-making for interventions (such as crime, traffic, accidents, emergency hospitals, and meteorological phenomena) in an autonomous manner, assisting and improving people lives.

A possible way to autonomously discovering human preferences is through the integration of approaches that reflect the human cognitive process. For this purpose, studies must be carried out in order to check how the machine may work collaboratively with humans to solve problems with multiple conflicting objectives in real and large-scale domains. We intend to perform experiments in more complex and realistic scenarios, such as Smart Grids, Autonomous Vehicles, Flow Shop Machines, Social Media, Video Game Playing, and Travelling Salesman Problems.

Another possible approach to be analyzed is how to build options from human input in mutiobjective problems, verifying how it can affect the human decision-making and accelerate the learning process in several systems.

Finally, we also intend to improve our methods by allowing the use of options with stochastic policies(Koga, da Silva, and Costa 2015), aiming at improving and accelerating the knowledge generalization provided by our approach.

Acknowledgements

We are grateful for the support from the CEST Group, CNPq (grant 311608/2014-0), and São Paulo Research Foundation (FAPESP), grant 2015/16310-4.

References

Agichtein, E.; Castillo, C.; Donato, D.; Gionis, A.; and Mishne, G. 2008. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 183–194.

Bernstein, D. S. 1999. Reusing old policies to accelerate learning on new mdps. Technical report, University of Massachusetts, Amherst.

Bowling, M., and Veloso, M. 1998. Reusing learned policies between similar problems. In *Proceedings of the AI* AI-98 Workshop on New Trends in Robotics*.

Brys, T.; Harutyunyan, A.; Vrancx, P.; Taylor, M. E.; Kudenko, D.; and Nowé, A. 2014. Multi-objectivization of reinforcement learning problems by reward shaping. In *International Joint Conference on Neural Networks (IJCNN)*, 2315–2322. IEEE.

Erharuyi, N., and Fairbairn, D. 2003. Mobile geographic information handling technologies to support disaster management. *Geography* 88(4):312–318.

Hinze, A., and Voisard, A. 2003. Location-and time-based information delivery in tourism. In *International Symposium on Spatial and Temporal Databases*, 489–507. Springer.

Khamis, M. A., and Gomaa, W. 2014. Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Engineering Applications of Artificial Intelligence* 29:134–151.

Koga, M. L.; da Silva, V. F.; and Costa, A. H. R. 2015. Stochastic Abstract Policies: Generalizing Knowledge to Improve Reinforcement Learning. *IEEE Transactions on Cybernetics* 45(1):77–88.

Liu, C.; Xu, X.; and Hu, D. 2015. Multiobjective reinforcement learning: A comprehensive overview. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on* 45(3):385–398.

MacGlashan, J. 2015. Brown-UMBC reinforcement learning and planning (BURLAP), http://burlap.cs.brown.edu/index.html.

Mcgovern, A., and Barto, A. G. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, 361–368.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Ng, A. Y.; Coates, A.; Diel, M.; Ganapathi, V.; Schulte, J.; Tse, B.; Berger, E.; and Liang, E. 2006. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*. Springer. 363–372.

Ngai, D. C. K., and Yung, N. H. C. 2011. A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers. *Intelligent Transportation Systems, IEEE Transactions on* 12(2):509–522.

Pickett, M., and Barto, A. G. 2002. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, 506–513.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

Rinner, C., and Malczewski, J. 2002. Web-enabled spatial decision analysis using ordered weighted averaging. *Journal of Geographical Systems* 4(4):385–403.

Roijers, D. M.; Vamplew, P.; Whiteson, S.; and Dazeley, R. 2014. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research (JAIR)* 48:67–113.

Seo, Y.-W., and Zhang, B.-T. 2000. A reinforcement learning agent for personalized information filtering. In *IUI*, 248–251.

Silva, F. L., and Costa, A. H. R. 2015. Multi-objective reinforcement learning through reward weighting. In *TRI 2015, joint with IJCAI*, 25 – 36.

Singh, S.; Litman, D.; Kearns, M.; and Walker, M. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research* 16:105–133.

Smith, J.; Mackaness, W.; Kealy, A.; and Williamson, I. 2004. Spatial data infrastructure requirements for mobile location based journey planning. *Transactions in GIS* 8(1):23–22.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1st edition.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1):181–211.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10:1633–1685.

Tesauro, G. 1995. *TD-Gammon: A Self-Teaching Backgammon Program*. Boston, MA: Springer US. 267–285.

Thrun, S., and Schwartz, A. 1995. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*. MIT Press. 385–392.

Triantaphyllou, E. 2013. *Multi-criteria decision making methods: a comparative study*, volume 44. Springer Science & Business Media.

Van Moffaert, K.; Drugan, M. M.; and Nowé, A. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. In *ADPRL*, 191–199. IEEE.

van Treeck, T., and Ebner, M. 2013. How useful is twitter for learning in massive communities? an analysis of two moocs. *Twitter & Society* 411–424.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3):279–292.

Zeng, F.; Zong, Q.; Sun, Z.; and Dou, L. 2010. Self-adaptive multi-objective optimization method design based on agent reinforcement learning for elevator group control systems. In *WCICA*, 2577–2582. IEEE.