Neurocomputing 000 (2017) 1–9

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Identification and off-policy learning of multiple objectives using adaptive clustering

Thommen George Karimpanal*, Erik Wilhelm

Engineering Product Development, Singapore University of Technology and Design, 8 Somapah Road, Singapore 487372, Singapore

ARTICLE INFO

Article history:
Received 4 March 2016
Revised 1 March 2017
Accepted 21 April 2017
Available online xxx

Keywords:
Reinforcement learning
Q-learning
Off-policy
Adaptive clustering
Multiobjective learning

ABSTRACT

In this work, we present a methodology that enables an agent to make efficient use of its exploratory actions by autonomously identifying possible objectives in its environment and learning them in parallel. The identification of objectives is achieved using an online and unsupervised adaptive clustering algorithm. The identified objectives are learned (at least partially) in parallel using $Q-$learning. Using a simulated agent and environment, it is shown that the converged or partially converged value function weights resulting from off-policy learning can be used to accumulate knowledge about multiple objectives without any additional exploration. We claim that the proposed approach could be useful in scenarios where the objectives are initially unknown or in real world scenarios where exploration is typically a time and energy intensive process. The implications and possible extensions of this work are also briefly discussed.

1. Introduction

Intelligent agents are characterized by their abilities to learn from and adapt to their environments with the objective of performing specific tasks. Very often, in reinforcement learning [1], and in machine learning in general, algorithms are structured to be able to fulfill one specific objective, usually specified in terms of a particular region in the feature space that is associated with a high reward. In general, environments are likely to contain multiple features, and different regions in the feature space may specify different objectives that could be assigned to the agent to learn. In real-world scenarios, however, the ability to efficiently learn more than one objective during a single deployment could drastically improve the agent's usefulness. In order to achieve this, the agent would need to be aware of regions in the feature space that could possibly play a role in its future tasks.

Embodied artificial agents or intelligent robots are typically equipped with a variety of sensors that enable them to detect characteristic features in their environment. In the context of reinforcement learning, when such an agent is placed in an unknown environment and is assigned an objective, it carries out some form of exploratory behavior in order to first discover a region in the feature space that fulfills this objective. Further exploratory actions may help improve its value function estimates, which in turn lead to improved policies to achieve the objective. We shall refer to this original task as the *primary objective*, and to its associated feature vector as the *primary objective feature vector* ($\vec{\psi}$). During exploration, it is likely that the agent comes across other 'interesting' regions which contain features that stand out with respect to the agent's history of experiences. We shall refer to these regions of the feature space as *secondary objectives*, and to the associated feature vectors as *secondary objective feature vectors* ($\vec{\phi}$). Although these regions could be of interest to the agent for future tasks (which are currently unknown), they may be irrelevant to the task at hand. Hence, it is justified for the agent to ignore them and continue performing value function updates for the primary objective assigned to it.

However, the agent's future tasks may not remain the same and a new task assigned to it may correspond to a particular combination of features that it encountered while learning policies for the primary objective. In such a case, the fact that this region in the feature space had been previously encountered cannot be leveraged since they were not relevant to the agent at that point of time, and were hence ignored.

The above mentioned approach would result in a considerable amount of wasteful exploration. This is because each new task assigned to the agent would require a fresh phase of discovery and learning of the associated feature vector and value functions, respectively. A more efficient approach would be to keep track of possible secondary objectives and learn them in parallel using off-policy methods [1,2]. In the context of off-policy learning, this can

* Corresponding author.
E-mail addresses: thommen_george@mymail.sutd.edu.sg, thommengk@gmail.com (T.G. Karimpanal), erikwilhelm@sutd.edu.sg (E. Wilhelm).

http://dx.doi.org/10.1016/j.neucom.2017.04.074
0925-2312/

be done by treating the policies corresponding to the secondary objectives as target policies, and learning them while executing the behavior policy which is dictated by the primary objective. Depending on the objectives, the actions executed by the behavior policy may not be optimal with respect to the secondary objectives. However, using off-policy learning, it is possible to at least partially learn the value functions for the secondary objectives, thereby significantly improving the efficiency of exploration. In applications such as robotics where exploration is known to be costly in terms of time, energy and other factors, such an approach could prove to be practical.

In this work, we present a framework in which an unsupervised, adaptive clustering algorithm is designed and used to cluster regions of the feature space into different groups based on the similarity of their associated features. Off-policy methods are used to simultaneously learn target policies corresponding to these clusters, each of which is treated as a secondary objective. The clustering of features occurs as and when they are seen by the agent while learning the primary task. The value function updates can be performed using suitable off-policy methods, namely, tabular $Q-$ learning, $Q-\lambda$ [3] or other more recent off-policy methods [4] such as off-policy LSTD(λ) [5,6], off-policy TD(λ) [2,7], GQ(λ) [8] etc., The results presented here, however, correspond to the $Q-\lambda$ algorithm.

The primary objectives have an influence on the discovery and learning of the secondary objectives, but only through its behavior policy. As long as the agent executes some exploratory actions while learning to perform its primary task, secondary objectives can be discovered and at least partially be learned. In fact, even a purely exploratory policy can be used. These aspects are discussed in further detail in Section 5.

Ideally, our approach would obviate the need for a fresh phase of discovery and learning when the objective is changed. However, the aim here is not to learn all the secondary objectives perfectly, but to identify them via the adaptive clustering algorithm, and learn them at least partially through off-policy learning. Doing so could provide the agent with a good initialization of value function weights so that optimal policies for the identified possible objectives could be learned in the future, if needed.

2. Background

Reinforcement learning deals with developing strategies for an agent to act in its environment with the objective of maximizing the expected value of a scalar reward. Most research in reinforcement learning is based on the formalism of Markov Decision Processes (MDPs) [9]. In this framework, an agent in state $s \in \mathcal{S}$ takes an action $a \in \mathcal{A}$ to transition into a new state s' with a probability $P(s, a, s')$. At each state, the agent receives a scalar reward $R(s, a)$. All reinforcement learning methods can be thought of as ways to maximize the expected reward accumulated over time as the agent interacts with the environment. The outcome of these methods is a mapping from states to actions, referred to as a policy. If the learning agent learns the value function for the policy being executed, it is referred to as *on-policy* learning, and if it learns the value function for an objective irrespective of the policy being executed, it is called *off-policy* learning.

In this work, our goal is to identify secondary objectives and learn their corresponding policies in parallel while the agent executes its behavior policy based on its primary objective. Hence, *off-policy* learning methods are a natural choice for the stated goal. We use the $Q-\lambda$ algorithm, which is an extension of tabular $Q-$ learning that is suitable for application in continuous state spaces. The update equation for the tabular case is shown in Eq. (1)

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

where $Q(s, a)$ is the $Q-$value corresponding to state s and action a. s' is the next state, and a' is a bound variable that can represent any action in the action space $\mathcal{A}$. α is the learning rate and γ is the discount factor.

The $Q-\lambda$ algorithm performs a similar, but more involved update with weight vectors, and involves the use of eligibility traces [10]. Here, replacing traces are used for the $Q-\lambda$ updates [11]. The update equations for the $Q-\lambda$ algorithm are mentioned below:

$$\delta \leftarrow \delta + \gamma max_{a'} Q(s',a') \quad (2)$$

$$w \leftarrow w + \alpha\delta e \quad (3)$$

$$e \leftarrow \gamma\lambda e \quad (4)$$

where w is the weight vector, e is the eligibility trace vector, λ is the trace decay rate parameter and δ is defined as

$$\delta = R(s,a) - Q(s,a) \quad (5)$$

The elements of the eligibility trace vector (replacing traces) are initialized with a value of 1 if the corresponding features are active. Otherwise, they are initialized with a value of 0.

The $Q-$values mentioned in Eqs. (2) and (5) are stored in the form of weight vectors as:

$$Q(s,a) = \sum_{i \in \mathcal{F}_{act}(s,a)} w_i \quad (6)$$

where $F_{act}(s, a)$ is the set of active features for an agent in state s, taking an action a. A more detailed summary of the algorithm can be found in [1].

Although off-policy methods such as the ones described above have been well known and widely used over the years, their use for autonomously handling multiple independent objectives has been limited, primarily owing to very few precedents on unsupervised identification of objectives in an agent's environment. Off-policy approaches with function approximation have also been known to have long standing issues with stability until recently [12]. Although approaches for handling multiple independent objectives in parallel are rather limited, a number of multi-objective reinforcement learning approaches that handle multiple conflicting objectives exist. A comprehensive survey of such methods can be found in [13].

The horde architecture of Sutton et al. [12] has been shown to be able to learn multiple pre-defined objectives in parallel using independent reinforcement learning agents in an off-policy manner. The knowledge of these tasks is stored in the form of generalized value functions which makes it possible to obtain predictive knowledge relating to different goals of the agent. Modayil et al. [14] and White et al. [15] also focus on learning multiple objectives in parallel using off-policy learning. Apart from this, Sutton et al. [16] used off-policy methods to simultaneously learn multiple options [17], including ones not executed by the agent. They mention that the motivation for using off-policy methods is to make maximum use of whatever experience occurs and to learn as much as possible from them, which is an idea that is reflected in this work.

In the works mentioned above, the multiple objectives that are learned in parallel are pre-defined. However, in this work, we focus on the case where the agent has no foreknowledge of the objectives in its environment. The objectives are identified by the agent itself via clustering. Hence, the agent learns independently in the sense that as it moves through its environment, it identifies potential objectives and at least partially learns their associated value functions in parallel.

A similar approach is seen in Mannor et al. [18], where clustering is performed on the state-space to identify interesting regions. However, their approach was not online and the purpose of their

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

work was to use these regions to automatically generate temporal abstractions.

We use a variant of the K-means clustering algorithm [19,20] to cluster features that are characteristic of secondary objectives. The approach is similar to that of Bhatia [21], where an adaptive clustering approach is described. The difference lies in the fact that in our method, in addition to the mean, statistical properties such as the variance and number of members in each cluster are updated online and used for clustering as and when the environment is sensed by the agent.

In general, the algorithm also bears similarities to some aspects of adaptive resonance theory [22]. The procedure for finding and updating the winning cluster in our approach is similar to that for comparing input vectors to the recognition field, and updating recognition neurons towards the input vector in adaptive resonance theory. Perhaps the main differences in our approach are the nature and function of the threshold/vigilance parameter. In our approach, the threshold is related to the variance of the cluster, which varies dynamically as more members are acquired by the clusters. However, in both approaches, the threshold has an effect on the resolution of the clusters. Overall, our clustering approach is simpler, and it is only focused on being able to identify clusters in an online manner, without much consideration to factors such as biological plausibility. The details of the algorithm are discussed further in Section 4.

3. Description

In order to demonstrate the proposed approach for identifying and learning multiple objectives, we consider an agent in a 30x30 continuous space which contains obstacles, a region lit up by a light source, and a bumpy/rough area. We assume that characteristic features corresponding to these regions can be detected by the agent using its on-board sensors: a set of range sensors, a light detecting sensor, and an inertial motion unit (IMU) to sense changes in surface roughness. The range sensors on the robot are radially separated from each other by 72 degrees as shown in Fig. 1, and are capable of sensing the presence of obstacles within 1 unit distance. A sample of the environment is shown in Fig. 2.

Initially, the agent has no foreknowledge of the environment, and can move forwards and backwards, sideways and diagonally up or down to either side. In addition to this, it can also hold its current position. Thus, a total of 9 deterministic actions (called the action set $\mathcal{A}$) are available for execution. These actions are executed sequentially according to the behavior policy, which depends on the primary objective assigned to the agent. The time step for action execution is set to be 200 ms and the agent's velocity is set to be 8 units/s for the relevant actions. The features are a function of the environment and of the state of the agent, which is composed of the agent's *(x,y)* position and its heading direction. Deriving these features from the agent's state is critical to learning, and is described below.

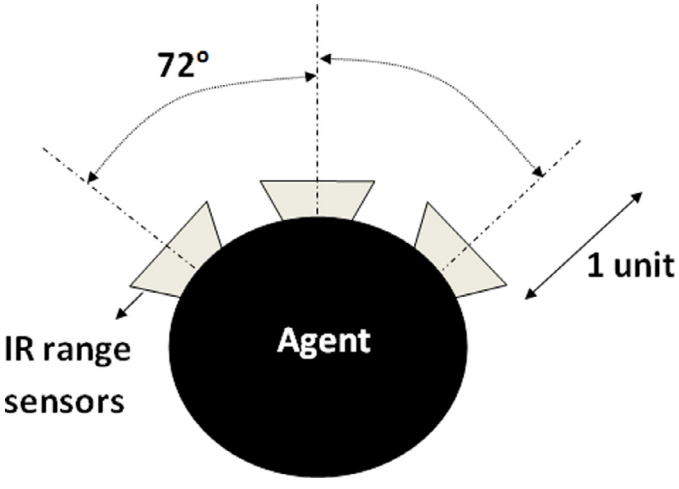

Fig. 1. The simulated agent and its range sensors.

3.1. Agent features

The agent is capable of sensing different features in the environment using its sensors. The sensors are simulated to have 5% Gaussian white noise. We shall refer to the resulting feature vector as the *environment feature vector* ($\vec{F_e}$). For learning policies using linear function approximation, additional features for the agent's localization are needed. We shall refer to the vector of these features as the *agent feature vector* ($\vec{F_a}$). Hence, the full feature vector for the agent consists of both these feature vector components ($\vec{F} = \vec{F_e} \cup \vec{F_a}$). All features used in this work are binary (1 or 0) for the sake of simplicity.

The feature vector $\vec{F_e}$ consists of the following:

1. Feature indicating either the presence or absence of obstacles as seen by any of the three range sensors.
2. Feature corresponding to the presence or absence of light
3. Feature corresponding to rough or smooth floor surfaces, as reported by the IMU
4. Feature indicating whether the agent lies within the range of the specified target location

The agent feature vector $\vec{F_a}$ is composed of 30 binary features corresponding to each dimension in the 2-dimensional space. It is concerned with the localization of the agent, and is used for learning the required policies. In $\vec{F_a}$, the feature value is equal to 1 for the agent's current position and 0 for all other positions in the space. Hence, the full feature vector consists of 64 (60 localization and 4 environment) feature elements.

Only $\vec{F_e}$ is passed into the clustering algorithm to identify different regions of interest, whereas the full feature vector is used for the $Q - \lambda$ updates.

4. Methodology

Section 3 described the simulated environment, the agent and the features it is capable of sensing. In this section, we describe the methodology used to identify regions of interest in the feature space and how these regions, treated as secondary objectives, can be learned using off-policy methods.

4.1. Adaptive clustering

As described earlier, the feature vector sensed by the agent consists of features relating to the environment as well as features for localization of the agent. The agent is initially assigned an arbitrary primary objective, which is specified in terms of $\vec{\psi}$, which is a particular configuration of $\vec{F_e}$. In specifying $\vec{\psi}$, apart from the binary values that each feature can take, a 'don't care' case is also included. During the task specification, if a primary objective feature is associated with the 'don't care' case, it implies that any feature value sensed for that feature is considered acceptable during the search for $\vec{\psi}$ in the feature space. In learning the primary objective, the agent learns a policy that takes it from any arbitrary state in the environment to a state where $\vec{F_e}$ matches the feature vector described by $\vec{\psi}$

As the agent moves through the environment in search of the feature vector specified by $\vec{\psi}$, it is continuously presented with new $\vec{F_e}$ vectors. Our approach is to cluster these features as and

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

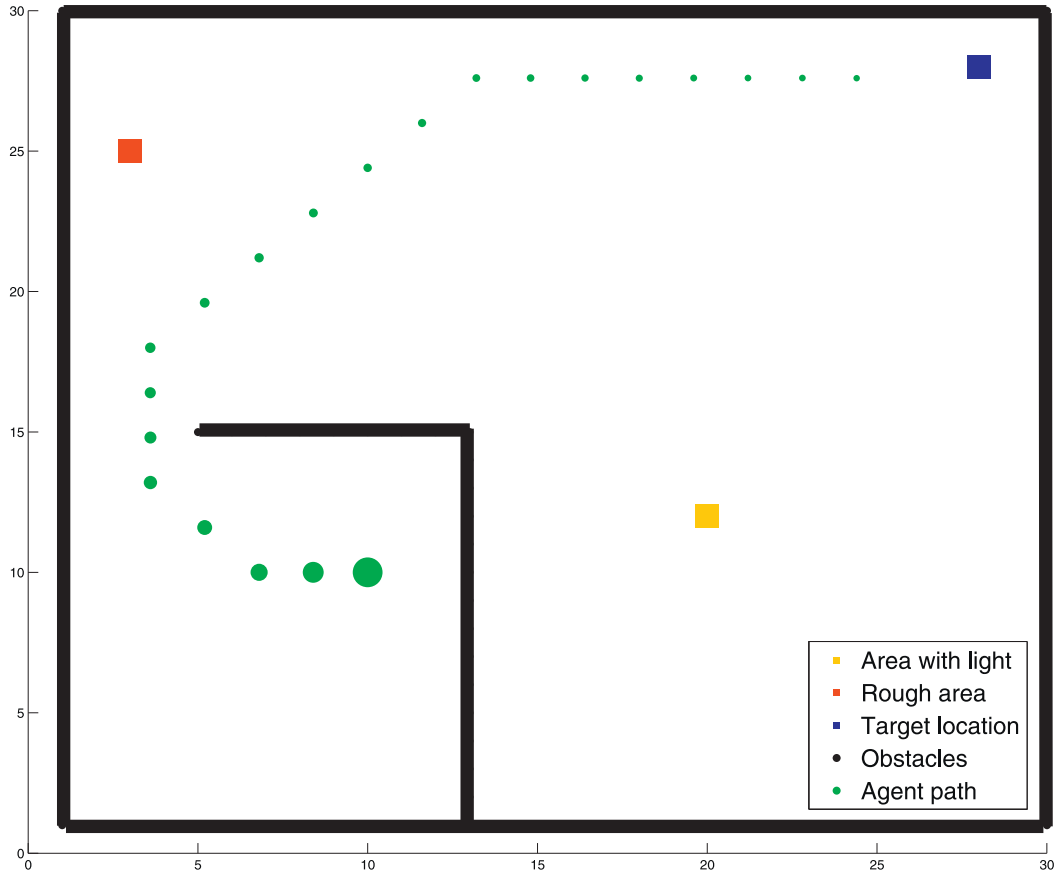

Fig. 2. Trajectory corresponding to one of the agent's policies to navigate to the target location in the simulated environment. The environment contains features such as a region with light, a rough region, obstacles and a target location.

when they are seen. The K-means [20,23] algorithm is a simple and popular algorithm used for unsupervised clustering. However, it requires prior knowledge of the number of clusters present in the feature space. This does not suit our application, as we assume no prior knowledge about the environment. The algorithm was therefore modified in order to make it adaptive, so that new feature vectors that seem different from the others may 'seed' new clusters. This is done by continuously updating the statistical properties such as the mean, variance and number of members in each cluster, and by measuring the closeness of new features $\vec{F_e}$ to the mean vectors of the different clusters.

Each new cluster that is seeded is initially set to have non-zero variance, which we shall refer to as the *seed variance*. This is done in order to initially maintain a certain level of uncertainty about the cluster. The uncertainty reduces as more number of samples are observed. As the agent moves through the environment, the Euclidean distance between the environment feature vector $\vec{F_e}$ that it sees, and the mean vector of each cluster is calculated, and the cluster corresponding to the minimum distance is chosen as the 'winning' cluster. Next, the element-wise absolute distance between the mean of the winning cluster and $\vec{F_e}$ is computed. For each element, if this distance lies within 'n' standard deviations of the mean of that feature element, then the $\vec{F_e}$ belongs to that cluster; if not, a new cluster is seeded. So 'n' can be considered a tolerance parameter for the clustering algorithm. Each time a cluster receives a new member, the mean and variance of each of the jth feature element in the cluster is updated online using the corresponding elements of $\vec{F_e}$. The straightforward equations governing the updates are defined by the first and second statistical moments of the sensor measurement in Eqs. (7) and (8), respectively.

$$\mu_j \longleftarrow (N_C * \mu_j + F_e^j)/(N_C + 1) \tag{7}$$

$$\sigma_j{}^2 \longleftarrow (N_C * (\sigma_j^2 + \mu_j^2) + F_e^{j2})/(N_C + 1) - \mu_j^2 \tag{8}$$

$$N_C \longleftarrow N_C + 1 \tag{9}$$

where μ_j and $\sigma_j{}^2$ are respectively the mean and variance of the jth feature element in the cluster, whereas N_C is the number of members in cluster C. The structure of the clustering algorithm is summarized in Algorithm 1.

Overall, the algorithm serves to cluster the feature space in an unsupervised and adaptive manner without prior knowledge of the number of clusters that exist in the space. Each of the identified clusters is treated as a secondary objective which is learned in parallel with the primary objective using off-policy methods.

4.2. Off-policy learning

The clustering algorithm described in Section 4.1 groups feature vectors $\vec{F_e}$ into different clusters in an adaptive and unsupervised manner. As and when each new cluster is seeded, an associated set of weight vectors (to learn the corresponding Q function) is also created. The mean vectors of each of these clusters is treated as a secondary objective feature vector $\vec{\phi}$, and the associated set of weight vectors is updated using the $Q-\lambda$ algorithm, based on actions resulting from the behavior policy. So during each episode of learning of the primary objective, the secondary objectives

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

Algorithm 1 Adaptive clustering algorithm.

Inputs: Feature vector $\vec{F_e}$, variance threshold parameter n, number of existing clusters K (initially set to 1), existing clusters C and their properties: mean $\vec{\mu}$, standard deviation $\vec{\sigma}$ (elements initialized with non-zero seed variance for a new cluster) and number of members N_{C_K} (initialized to 1 for a new cluster)
for i=1:K **do**
  $d_i = Euclidean\_distance(\vec{F_e}, \vec{\mu}_i)$
end for
$win = \{argmin(d)\}$
if $|(F_e^j - \mu_{win}^j)| \geq n * \sigma_{win}^j$ for each feature F_e^j in $\vec{F_e}$, **then**
  $K = K + 1$
  $\vec{F_e} \in C_K$
else
  $\vec{F_e} \in C_{win}$
  Update the mean and variance of each element in the winning cluster
  $\mu_{win}^j \longleftarrow (N_{C_{win}} * \mu_{win}^j + F_e^j)/(N_{C_{win}} + 1)$
  ${\sigma_{win}^j}^2 \longleftarrow (N_{C_{win}} * ({\sigma_{win}^j}^2 + {\mu_{win}^j}^2) + {F_e^j}^2)/(N_{C_{win}} + 1) - {\mu_{win}^j}^2$
  Update the number of members in the winning cluster
  $N_{C_{win}} \longleftarrow N_{C_{win}} + 1$
end if

identified by the clustering algorithm are learned simultaneously using off-policy learning. At the same time, new secondary objectives are identified by the clustering algorithm. If 'M' secondary objectives are identified, the $Q-\lambda$ updates are performed 'M' times in addition to the one time that the update is carried out for the primary objective. For these additional updates, the scalar rewards are dictated by the associated secondary objective. The reward structure used here is simple, and all objectives are treated equally. A reward of 100 is awarded for successfully achieving an objective, and a living penalty of 10 is associated with each step that does not correspond to the fulfillment of an objective. In addition to this, irrespective of the objective, a penalty of 100 is assigned for bumping into obstacles. More sophisticated reward structures that reflect the relative importance of the different objectives may be explored in the future. The overall algorithm is summarized in Algorithm 2.

5. Results

In this section, we summarize the results obtained by applying the methodology described in Section 4 to the agent and environment described in Section 3. The sample environment used for the simulations are shown in Figs. 2 and 5. In these figures, larger markers corresponding to the agent's path signify points closer to the starting position of the agent. The configuration of the obstacles in the environment is set up to be similar to the 'puddle world' problem [24], in the sense that in order for the agent to navigate to the required location, it may need to temporarily move away from its target location.

The agent executes an ϵ- greedy policy while learning a primary objective, during which it senses features $\vec{F_e}$ in its environment, and continuously sorts them into new or existing clusters as dictated by Algorithm 1. Fig. 3 shows the clusters identified by the algorithm after the $Q-\lambda$ algorithm is applied to learn the primary objective of navigating to the target location. In Fig. 3, a total of 7 clusters can be seen, each marked with a distinct texture and number. It is also seen that regions that have an overlap of different types of features are sorted as different clusters. For example, the region near the top right corner of Fig. 3 contains a cluster (marked as cluster 7) which corresponds to the overlap between an area around the target location and the presence of an

Algorithm 2 Identifying and learning objectives using clustering and off-policy methods.

Inputs: Primary objective feature vector ($\vec{\psi}$), variance threshold parameter n, number of existing clusters K (initially set to 1), starting state (x_{start}), weight vector w_O, $Q-\lambda$ parameters for primary objective: discount factor (γ), learning rate (α), exploration parameter (ϵ), decay rate parameter for eligibility traces (λ) number of iterations for $Q-\lambda$ (N_iter), existing clusters C and their properties: mean $\vec{\mu}$, standard deviation $\vec{\sigma}$ and number of members N
for i=1:N_iter **do**
  state=x_{start}
  $\vec{F_e} = getfeaturesfromstate(state)$
  while $\vec{F_e} \neq \vec{\psi}$ **do**
    Take ϵ-greedy action and visit new state x_{new}
    $\vec{F_{e}}_{new} = getfeaturesfromstate(x_{new})$
    Cluster $\vec{F_{e}}_{new}$ using algorithm 1
    if New clusters are formed, **then**
      Seed $w_{new\_cluster}$ and update K
    end if
    if $\vec{F_{e}}_{new} == \vec{\psi}$, **then**
      reward=high
    elsereward=low
    end if
    Update w_O using $Q-\lambda$ equations
    for j=1:K **do**
      $\vec{\phi} = \vec{\mu}_j$
      if $\vec{F_{e}}_{new} == \vec{\phi}$ **then**
        reward(j)=high
      elsereward(j)=low
      end if
      Update w_j using $Q-\lambda$ equations
    end for
    $x = x_{new}$
    $\vec{F_e} = \vec{F_{e}}_{new}$
  end while
end for

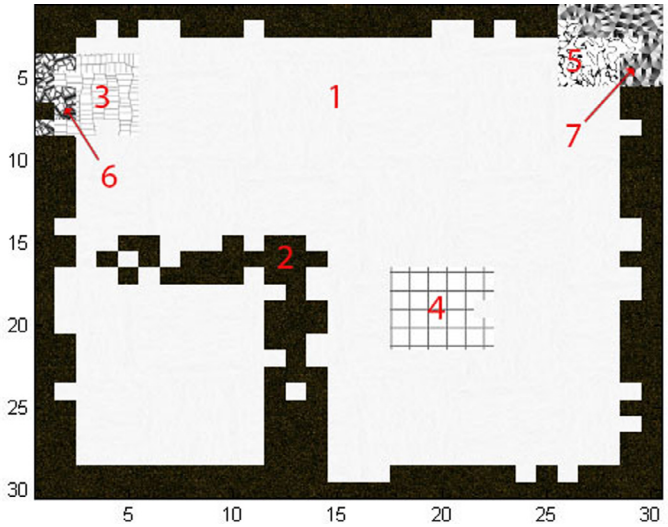

Fig. 3. Different clusters detected by the agent for the environment shown in Fig. 2.

obstacle. In Fig. 4, it is seen that during episode 1, this overlapping area is not distinguished as a separate cluster. This changes as the episodes proceed, and the overlapping area is eventually identified as a distinct cluster after episode 6. A similar overlap exists (marked as cluster 6 in Fig. 3) around the area with high floor roughness near the top left corner of the environment. This

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

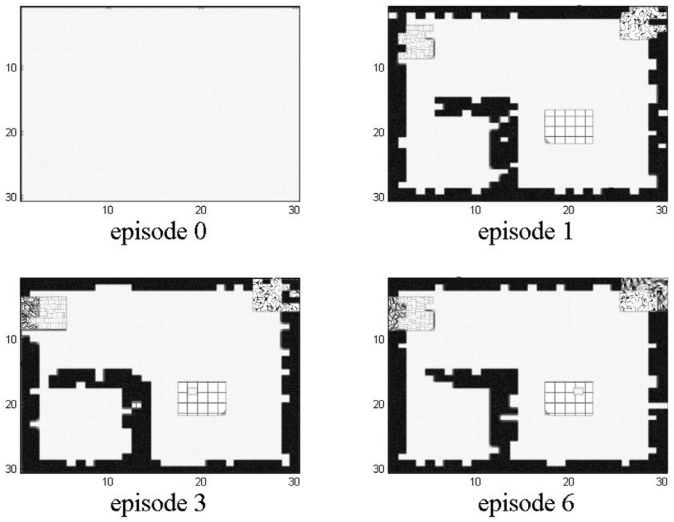

Fig. 4. Progression of cluster formation with episodes of the $Q-\lambda$ algorithm.

Table 1
Average number of clusters formed as clustering parameters seed variance and clustering tolerance (n) are varied.

	n=0.1	n=1	n=1.1	n=1.5	n=2
Seed variance=0.1	6.82	6.65	1.93	1.36	1.39
Seed variance=1	6.77	6.33	1.47	1.19	1
Seed variance=100	6.49	6.51	1.63	1.06	1

shows that with a larger number of samples, the clustering algorithm is capable of distinguishing different combinations of feature elements in the feature space in an unsupervised manner.

Table 1 shows the average number of clusters identified as the seed variance and the clustering tolerance n are varied. The values shown are compiled for 50 $Q-\lambda$ runs with an exploration parameter $\epsilon = 0.3$ for 1000 episodes. The other parameters are the learning rate $\alpha = 0.3$, the discount factor $\gamma = 0.9$ and the trace decay rate parameter $\lambda = 0.9$. These parameters were kept constant for the $Q-\lambda$ runs. The results shown in Table 1 suggest that the clustering is sensitive to the clustering tolerance, as we may have expected. The lower the value of n, the larger is the number of clusters identified. As per Algorithm 1, the condition for new clusters to be formed is:

$$|(F-\mu)| \geq n\sigma \tag{10}$$

where F is the value of the feature element and μ and σ are the mean and standard deviation of the associated 'winning' cluster. From Chebyshev's inequality, the probability of clusters forming is bounded by:

$$P(|(F-\mu)| \geq n\sigma) \leq 1/n^2 \tag{11}$$

When $n \leq 1$, the term on the right hand side of Eq. (11) is ≥ 1. Since probabilities cannot exceed 1, all cases of $n \leq 1$ are equivalent in this sense. When $n > 1$, the probability reduces, and the clustering performance drops. This could provide some explanation for the trends seen in Table 1. It also suggests that the clustering tolerance n should ideally be set to a value ≤ 1 if clusters are to be identified effectively.

In addition to this, the performance of the clustering algorithm is observed to be more or less independent of the seed variance. This is because the variance of each cluster is continuously updated with each visit to a state. As more samples are obtained, the initial seed variance assigned to a cluster is quickly corrected to be closer to its true value. For the given environment and agent, the clusters were mostly identified during the early episodes of $Q-$ learning. Fig. 4 shows a typical progression of cluster formation with the number of episodes.

Table 2
Percentage of starting positions picked uniformly from the environment, that lead to acceptable policies for the primary and selected secondary objectives.

No. of episodes	Objectives	ϵ=0.1	ϵ=0.3	ϵ=0.7	ϵ=1
100	Primary objective	51.97	60.32	71.35	81.62
	Light area	12.59	17.84	26.70	72.37
	Rough area	15.43	16.78	28.52	60.48
300	Primary objective	69.92	72.57	80.76	82.13
	Light area	12.65	18.38	29.03	80.05
	Rough area	16.16	18.27	27.41	81.94
1000	Primary objective	78.62	82.35	88.24	90
	Light area	16.22	21.30	52.16	85.54
	Rough area	17.19	21.38	41.08	85.41

The clusters identified by the adaptive clustering algorithm are passed on as secondary objectives to be learned using off-policy learning. The mean vector of these clusters, which describe the features represented by the cluster are then used to construct the feature vectors of the respective secondary objectives ($\vec{\phi}$).

For the case of feature vectors $\vec{F_e}$ with a large number of elements, the number of clusters identified is likely to be large. For example, when 60 additional features were added to the environment feature vector described in Section 3, a total of 748 different clusters were formed. In such cases, it may be more practical to choose a certain number of clusters based on some predetermined criteria, and learn their associated policies. An example of one such criterion would be the average value of the temporal difference (TD) error across the state-action space, with secondary objectives corresponding to lower average error values being preferred. The hypothesis is that since the reward structures for the different objectives are similar, objectives with the lowest average TD error are likely to have have been learned more reliably. Hence, the objectives could be sorted in this manner according to the reliability of their associated $Q-$functions.

Once the clustering algorithm identifies a secondary objective, its corresponding weight vectors are initialized, and its value function is learned by making use of whatever experience could be gained from the agent's behavior policy. At the end of each episode of learning, the agent's starting position is reset to a random non-goal state. Ideally, after $Q-$learning, the agent should be able to generate optimal trajectories starting from anywhere in its environment subject to the assumption that each state-action pair is visited infinitely often. Table 2 shows the percentage of starting positions, picked uniformly from the environment shown, which lead to acceptable policies. The variation of this quantity with the number of episodes and different exploration parameters is also shown. The policies being evaluated are generated by having the agent take a series of greedy actions (as per the value functions it has learned) till the goal state is reached. A policy (and its corresponding trajectory) is considered acceptable if the path resulting from it is similar to the one computed using the A-star algorithm [25], which computes the optimal trajectory between two points, given perfect information regarding the environment. Policies whose resulting paths are more than 50% longer than those computed using A-star are not considered acceptable. The tolerance (50%) used here may seem very high, but this is because the aim of our approach is to provide the value functions of the secondary objectives with good initializations using whatever experience occurs; it is not to learn the secondary objectives perfectly. We posit that given the same behavior policy, the agent will be able to learn the optimal value functions corresponding to the secondary objectives much faster when starting with good initial estimates. The comparison with A-star is performed in order to evaluate the general quality of the value functions for different

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

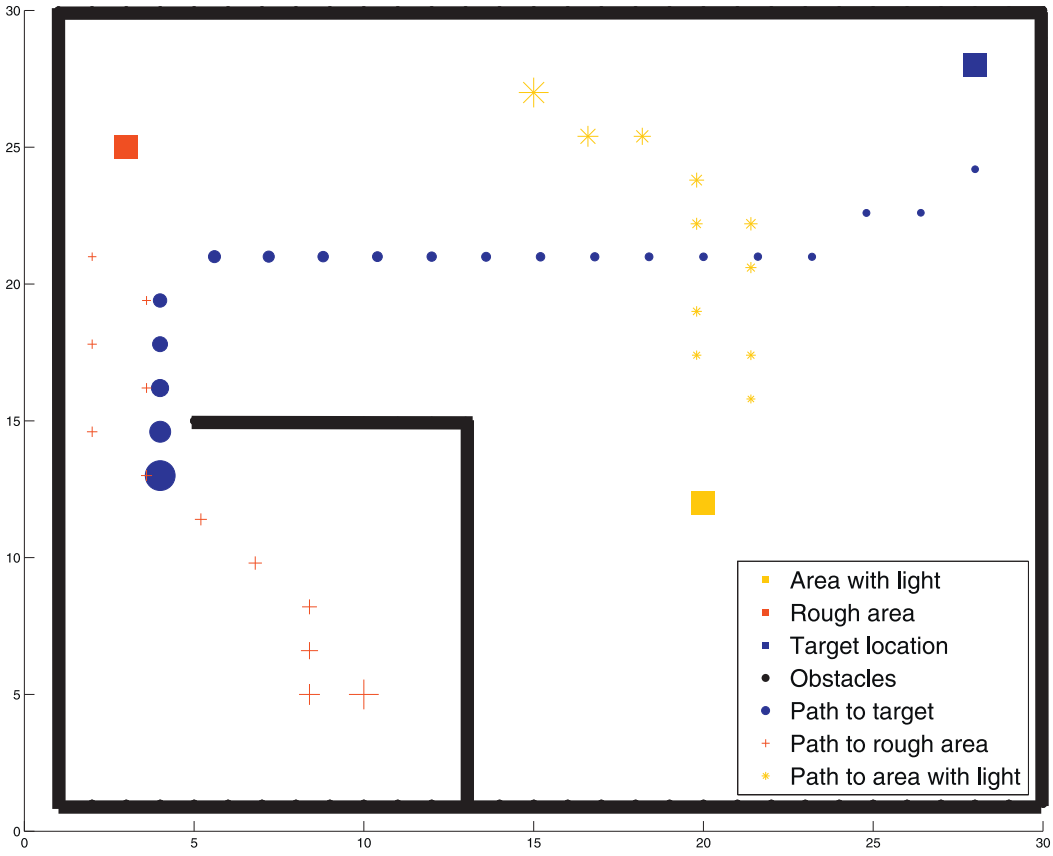

Fig. 5. Trajectories corresponding to the policies for different objectives learned by executing the behavior policy for the original task.

objectives. Each of the clusters shown in Fig. 3 is treated as a secondary objective, but only the values for meaningful secondary objectives such as navigating to the regions with light or those corresponding to a rough area have been tabulated.

The values from Table 2 suggest that in general, the value functions resulting from policies that are more exploratory in nature result in acceptable policies from a greater percentage of starting positions. This is expected, as more state-action pairs have a chance to be visited when ϵ is set to be large. In general, the percentage also goes up marginally with an increased number of learning episodes. This is natural, as a larger number of episodes allow greater opportunity for more state-action pairs to be visited more frequently.

Fig. 5 shows some of the sample learned trajectories for both the primary objective (navigating to the target location) as well as the two selected secondary objectives. The trajectories leading to the 'light' and 'rough' areas in Fig. 5 correspond to policies that were learned by first identifying the relevant regions in the feature space as secondary objectives, and then simultaneously learning (partially) their associated action-value functions through off-policy learning. If each of the 'N' secondary objectives were to be learned sequentially using $Q-$learning, 'N' additional phases of exploration and learning would have been required. Here, the value function for all the secondary objectives are learned at least partially from the experience gained while learning to perform the primary objective. Although the percentages in Table 2 seem to attain high values for the secondary objectives only under more exploratory behavior policies ($\epsilon = 0.7$ and $\epsilon = 1$), some knowledge of the corresponding objectives is gained even when the behavior policy is set to be relatively greedy. Even this partial knowledge of the secondary objectives could help provide some initial estimates of the value function when optimal policies corresponding to these objectives are required to be learned. In this manner, the efficiency of exploration is improved to some extent, irrespective of whether the agent's behavior policy is greedy or highly exploratory.

This point is further emphasized through Fig. 6, where the number of episodes to convergence is measured and plotted for different objectives under different behavior policies. Here, convergence is defined to be achieved when the agent is able to successfully navigate (have a trajectory length close to that specified by the A-star algorithm) to the required regions from the majority of locations in the environment. In the simulations summarized by Fig. 6, different objectives are set as primary ones, and the agent is made to learn them while discovering and learning secondary objectives in parallel. Once the primary objectives converge, the partially converged weights of the secondary objectives are made to dictate the behavior policy. Then, the number of episodes for these secondary objective policies to converge is measured by checking for convergence after each episode. Each data point in Fig. 6 is obtained by averaging the values of 50 $Q-\lambda$ runs with the corresponding ϵ values. The data points represented by solid shapes indicate primary objectives, whereas non-solid shapes represent secondary objectives.

From Fig. 6 the convergence for secondary objectives is observed to be faster in comparison to the case where the same objective is learned from scratch as a primary one. This is because for the secondary objectives, learning is initialized with weight vectors that are already partially converged owing to the off-policy learning that occurred while learning the primary objective. For example, in Fig. 6 (a) (ϵ=0.7), the primary objective is set to be to navigate towards the target location. The corresponding value function weights converge in about 60 episodes. During this period, secondary objectives are simultaneously identified and learned. The partially converged weights for two of these objectives (light and

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

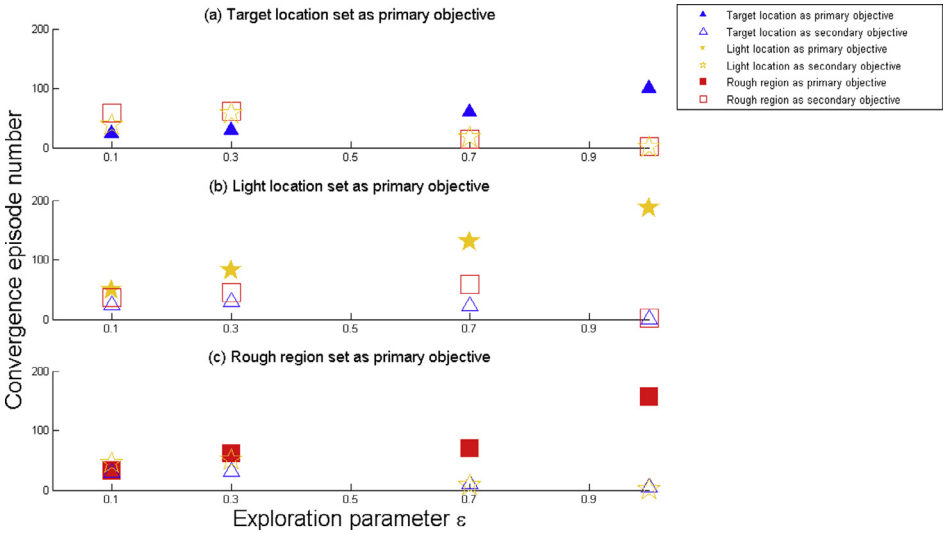

Fig. 6. Number of episodes for convergence for different values of ϵ with different objectives set as primary.

rough region) are used to initialize the learning for the corresponding objectives till convergence. It is seen from Fig. 6(a) (ϵ=0.7) that for the secondary objective of navigating to the rough region, convergence is achieved in only about 17 episodes on average. This is much faster than the case where the 'rough region' objective is learned from scratch as a primary objective, where convergence takes place after about 70 episodes as indicted in Fig. 6(c) (ϵ=0.7). For the secondary objective of navigating towards the light, convergence is achieved in 14 episodes (Fig. 6(a), (ϵ=0.7)), whereas it would have taken about 130 episodes (as seen from Fig. 6(b) (ϵ=0.7)) if the light objective were to be learned from scratch as a primary objective.

In general, for more exploitative behavior policies (ϵ=0.1 and ϵ=0.3 in Fig. 6), improvements may still exist, but it is less drastic. For example, when $\epsilon = 0.3$, the objective of navigating to a region with light takes about 25 episodes fewer to converge when deployed as a secondary objective as compared to when deployed as a primary one. The reduced improvement is due to the fact that agents under a greedy policy seldom deviate much from their path towards the primary objective. As a result, secondary objectives are visited less frequently unless they happen to lie along the optimal path towards the primary objective.

Hence, the effectiveness of the proposed methodology depends on the agent's behavior policy (whether exploitative or exploratory) as well as on the configuration of different objectives in the environment. However, in an arbitrary environment, our approach could possibly enable a significant reduction in the learning time (represented here by the number of episodes for convergence) required to learn the optimal policy for a secondary objective, even when relatively greedy policies are employed.

6. Discussion

As demonstrated in Section 5, the value function weights, even if partially converged, can make good starting points for carrying out subsequent $Q-$ learning episodes if improvement in the value function estimates is needed. Although we used the $Q-\lambda$ algorithm in this work, other off-policy methods could also be used, perhaps in conjunction with suitable abstraction techniques such as tile coding [24,26].

In employing the approach described here, it is to be noted that the secondary objectives identified during clustering may or may not be of relevance to the agent in the future. Assessing the relevance or relative importance of these objectives could be an area for further research. In addition to this, the construction of the reward structure in a more informed manner could also be explored further.

Nevertheless, we believe our approach could be useful in several fields, with direct applications in transfer learning [27], where it could provide jumpstart improvements [28] when the partially learned weights are transferred within or across agents. It can also be useful in multi-agent applications [29,30], as the value function information of the secondary objectives could be communicated to another agent whose primary objective is similar in nature to one of the original agent's secondary objectives. This could be a much more efficient approach, as each agent need not explore the environment from scratch. The exploration performed by other agents could be leveraged by subsequent agents to carry out their individual tasks.

7. Conclusion

The methodology developed and presented here demonstrates how the discovery and learning of potential objectives in an agent's environment is possible. Potential objectives are identified using an online, unsupervised and adaptive clustering algorithm. The identified objectives are then learned in parallel using off-policy methods. Both clustering as well as off-policy learning are demonstrated using a simulated agent and environment. The performance of the clustering algorithm with respect to its input parameters is tabulated and the findings are explained. The clustering algorithm is shown to be capable of identifying most of the distinct regions in the environment during the early episodes of $Q-$ learning. Simulations conducted to validate the utility of this approach reveal that the agent is able to at least partially learn multiple objectives in parallel without any additional exploration. This is especially true when the behavior policy itself is exploratory in nature. The future scope, possible extensions to this work and its applications to fields such as transfer learning and multi-agent systems are also briefly discussed. Although the efficiency of our approach depends to some extent on the nature of the behavior policy and the configurations of objectives in the environment, we believe it presents a potential to dramatically improve the efficiency of exploration for reinforcement learning agents in unknown environments.

References

[1] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, 1, MIT press Cambridge, 1998.

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074

[2] D. Precup, R.S. Sutton, S. Dasgupta, Off-policy temporal-difference learning with function approximation, in: Proceedings of International Conference on Machine Learning (ICML), 2001, pp. 417–424.
[3] C.J.C.H. Watkins, Learning from delayed rewards, (Ph.D. thesis), University of Cambridge England, 1989.
[4] M. Geist, B. Scherrer, et al., Off-policy learning with eligibility traces: a survey., J. Mach. Learn. Res. 15 (1) (2014) 289–333.
[5] H. Yu, Convergence of least squares temporal difference methods under general conditions, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 1207–1214.
[6] M.G. Lagoudakis, R. Parr, Least-squares policy iteration, J. Mach. Learn. Res. 4 (Dec) (2003) 1107–1149.
[7] D. Precup, Eligibility traces for o-policy policy evaluation, Computer Science Department Faculty Publication Series (2000) 80.
[8] H.R. Maei, R.S. Sutton, Gq (λ): a general gradient algorithm for temporal-difference prediction learning with eligibility traces, in: Proceedings of the 3rd Conference on Artificial General Intelligence, 1, 2010, pp. 91–96.
[9] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st, John Wiley & Sons, Inc., New York, NY, USA, 1994.
[10] R.S. Sutton, Learning to predict by the methods of temporal differences, Mach. Learn. 3 (1) (1988) 9–44.
[11] S.P. Singh, R.S. Sutton, Reinforcement learning with replacing eligibility traces, Mach. Learn. 22 (1–3) (1996) 123–158.
[12] R.S. Sutton, J. Modayil, M. Delp, T. Degris, P.M. Pilarski, A. White, D. Precup, Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction, in: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 761–768.
[13] D.M. Roijers, P. Vamplew, S. Whiteson, R. Dazeley, et al., A survey of multi-objective sequential decision-making., J. Artif. Intell. Res. (JAIR) 48 (2013) 67–113.
[14] J. Modayil, A. White, R.S. Sutton, Multi-timescale nexting in a reinforcement learning robot, Adapt. Behav. 22 (2) (2014) 146–160.
[15] A. White, J. Modayil, R.S. Sutton, Scaling life-long off-policy learning, in: Proceedings of IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL), IEEE, 2012, pp. 1–6.
[16] R.S. Sutton, D. Precup, Intra-option learning about temporally abstract actions, in: In Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufman, 1998, pp. 556–564.
[17] R.S. Sutton, D. Precup, S. Singh, Between MDPS and semi-MDPS: a framework for temporal abstraction in reinforcement learning, Artif. Intell. 112 (1–2) (1999) 181–211.
[18] S. Mannor, I. Menache, A. Hoze, U. Klein, Dynamic abstraction in reinforcement learning via clustering, in: Proceedings of the 21st International Conference on Machine Learning, ACM, 2004, p. 71.
[19] J.A. Hartigan, M.A. Wong, Algorithm as 136: a k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.
[20] M.R. Anderberg, Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks, 19, Academic Press, 2014.
[21] S.K. Bhatia, Adaptive k-means clustering., in: Proceedings of the Conference on Florida Artificial Intelligence Research Society (FLAIRS), 2004, pp. 695–699.
[22] G.A. Carpenter, S. Grossberg, Adaptive resonance theory, in: Encyclopedia of Machine Learning and Data Mining, Springer, 2016, pp. 1–17.
[23] H.C. Romesburg, Cluster analysis for researchers, Lifetime Learning Publications, Belmont, CA, 1984.
[24] R.S. Sutton, Generalization in reinforcement learning: successful examples using sparse coarse coding, Adv. Neural Inf. Process. Syst. 8 (1996) 1038–1044.
[25] P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (1968) 100–107.
[26] S.A. Whiteson, Adaptive Representations for Reinforcement Learning, University of Texas at Austin, 2007.
[27] M.E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: a survey, J. Mach. Learn. Res. 10 (2009) 1633–1685.
[28] A. Lazaric, Transfer in reinforcement learning: a framework and a survey, in: Reinforcement Learning, Springer, 2012, pp. 143–173.
[29] M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: Proceedings of the 10th International Conference on Machine Learning, 1993, pp. 330–337.
[30] L. Busoniu, R. Babuska, B. De Schutter, A comprehensive survey of multiagent reinforcement learning, IEEE Trans. Systems Man Cybern. Part C Appl. Rev. 38 (2) (2008) 156.

Thommen George Karimpanal is a Ph.D. candidate at Singapore University of Technology and Design. He received his Bachelor's degree (B.Tech.) in Mechanical Engineering from National Institute of Technology, Jalandhar, India in 2010 and his Master's degree (M.Sc.) in Mechatronics from National University of Singapore in 2014. His current area of research includes reinforcement learning and optimization for control.

Erik Wilhelm received B.S. and M.S. degrees in from the University of Waterloo, Waterloo, Ontario, in 2007 and Dr.Sci. ETH-Zurich in 2011, and after post-doctoral studies at MIT he joined the Engineering Product Development Faculty at the Singapore University of Technology and Design. His research interests include powertrain design, energy storage and conversion, optimal and robust control, applied machine learning, transportation systems, pervasive sensing.

Please cite this article as: T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.074