Evolving Systems
https://doi.org/10.1007/s12530-019-09290-9

ORIGINAL PAPER

Clustering subspace generalization to obtain faster reinforcement learning

Maryam Hashemzadeh[1] · **Reshad Hosseini**[1,2] · **Majid Nili Ahmadabadi**[1]

Received: 2 November 2018 / Accepted: 29 May 2019

Abstract
In reinforcement learning, very low and even lack of spatial generalization capability result in slow learning. Exploiting possible experience generalization in subspaces, a sub-dimension of the original state representation, is one approach to speed up learning in terms of required interactions. The target of this paper is to alleviate the detriment of the perceptual aliasing in the subspaces further to enhance the benefit of their generalization. We augment an extra dimension to the subspaces coming from a clustering process on the state-space. Through this framework, called Clustered-Model Based Learning with Subspaces (C-MoBLeS), states with similar policies are categorized to the same cluster and the agent can exploit the generalization of the subspace learning more by this localization. Therefore, the agent uses generalization of the subspaces which are in the cluster of the agent's state. Several experiments show that C-MoBLeS increases the learning speed effectively.

Keywords Model-based reinforcement learning · Experience generalization · Clustering space

1 Introduction

Reinforcement learning (RL) provides a learning framework to attain an autonomous agent that interact with environment to learn an optimal policy. It has been demonstrated that excellent performance in many applications can be obtained through employing RL framework, including real problems (e.g. helicopter flight Ng et al. 2006, quadrupedal locomotion Kohl and Stone 2004, and spoken dialog system Singh et al. 2002), game playing with various genres and difficulties (e.g. TD-Gammon Tesauro 1995, Atari 2600 video games Mnih et al. 2015; Oh et al. 2015; Hausknecht and Stone 2015; Schulman et al. 2015; Stadie et al. 2015; Mnih et al. 2016, StarCrafts Vinyals et al. 2017; Synnaeve et al. 2016, chess Lai 2015, and Go Silver et al. 2016) and robotics (e.g. robotic manipulation Gu et al. 2017; Rajeswaran et al. 2017; Popov et al. 2017, robots navigation Zhu et al. 2017; Bruce et al. 2017, assistive robots to learn social behavior Hemminghaus and Kopp 2017, and human-robot to learn complex motion sequences Li et al. 2018; Modares et al. 2017).

Although RL has a variety range of applicability among interactions learning methods, in its basic incarnation it has a fundamental limitation which is the number of required interactions for learning. To attain an optimal policy, the RL agent needs performing interactions in an order that is over linearly related to the size of the learning environment. In huge domains especially when there is curse of dimensionality, this issue results in very slow learning and high regret. A successful approach for solving this problem is experience generalization (Sutton and Barto 2018) from one part to a larger part of the environment. There are many methods for experience generalization such as state coding (Sutton 1995), hierarchical learning (Kulkarni et al. 2016), skill acquisition (Gupta et al. 2017; Shoeleh and Asadpour 2017), multi agent learning (Wang et al. 2018; Stone and Veloso 2000), using neural networks (Ackley and Littman 1990) and deep reinforcement learning (Arulkumaran et al. 2017). Each of these methods has its own constrain and is suitable for a specific domain.

✉ Maryam Hashemzadeh
m.hashemzadeh.b@gmial.com

Reshad Hosseini
reshad.hosseini@ut.ac.ir

Majid Nili Ahmadabadi
mnili@ut.ac.ir

[1] School of ECE, College of Engineering, University of Tehran, Tehran, Iran

[2] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Published online: 04 July 2019

Using generalization in subspaces is another successful approach for experience generalization. This framework was used in Daee et al. (2014) for accelerating single-step learning and in Hashemzadeh et al. (2018) for multi-step learning. In the method proposed in Hashemzadeh et al. (2018) called MoBLeS, the agent uses multiple model-based learning corresponding to each subspace. Each subspace corresponds to a mapping from the original state-space to a lower dimensional state-space. Then through decision making phase, the agent relies more on a subspace which has more consistent experiences with the original task. The main idea behind this frameworks is that learning in simpler tasks is faster than a complex task and each subspace is a simpler task than the original. Therefore, the framework tries to generalize experiences in the subspaces by mapping learning in the simpler tasks to the original to expedite learning. Previous results showed that this approach can increase the learning speed in multi-step learning in terms of required experiences, especially in the early trials which the agent has not enough experiences in all states (Hashemzadeh et al. 2018).

In the aforementioned framework, several states of the original learning task is mapped to one state of a subspace that results in perceptual aliasing (Chrisman 1992) in the subspaces. In this framework, careful integration of the original task model and the models of the subspaces led to reduce detriment of perceptual aliasing (PA) and increase benefit of the generalization of the subspaces (Hashemzadeh et al. 2018). But there, many-to-one mapping from states of the original learning task to a state of a subspace is not local and the detriment of PA still affects the learning speed.

In this paper we decrease PA issue in the subspaces, existing in the MoBLeS method, further by localizing their generalization. We first cluster state-space into some clusters and then calculate generalization of experiences in each subspace-cluster separately. In the decision making phase, the agent utilizes learning of the subspaces from the cluster of the current state.

The approach for estimating models of different spaces and integrating the policies explained in the Sects. 3.1.2 and 3.1.3 is similar to that of MoBLeS framework. The main contribution of this paper is introducing clustered subspaces as a way to overcome the PA issue of the subspaces.

To summarize, the main contributions of this paper in compare to the MoBLeS framework proposed in Hashemzadeh et al. (2018) are:

- We illustrate through an example that PA problem in the subspaces can be resolved by clustering the subspaces. This part is explained in the beginning part of Sect. 3.
- We propose C-MoBLeS algorithm in Sect. 3.1 for using clustering in the subspaces. The main part of this algorithm, which is different that the MoBLeS algorithm, is introducing clustering and proposing a method to solve it explained in Sect. 3.1.1.
- Through several experiments on the simulated and real data sets, we show the efficacy of the proposed method. The results show significant improvement over the MoBLeS algorithm.

The rest of the paper is organized as follows. First, we give an overview of the generalization in the subspaces and utilizing it for accelerating model-based learning as explained in Hashemzadeh et al. (2018). Next, we introduce the proposed method and how clustering can help to enhance the generalization in the subspaces. Afterwards, we assess the performance of the framework in several 2D environments. We finalized the paper by a discussion and giving several ideas for future works.

1.1 Assumption and key definitions

We assume the tasks are static and they are finite discrete-time Markov Decision Process (MDP) with discrete state-action. The agent does not know MDP model, but the length of the reward interval is known as a prior knowledge.

In the following, we define some phrases used throughout the paper:

Markov decision process It comprises of a tuple $\{S, A(.), p(.,.,.), R(.,.)\}$, where S is a finite state set, $A(s)$ is a finite action set available in the state s, $p(s, a, s')$ is the transition probability from state s to state s' by taking action a, and $R(s, a)$ is the immediate reward by doing a in s.

State-space It is the state space whose states contain all features of the task observable by the agent. If the feature set contains k features $F = \{F^1, F^2, \ldots, F^k\}$, then the state-space is the Cartesian product of all k features. This space is shown by S and s_j denotes a state of this space.

Subspace It is the space for which the dimensionality of each state is less than the number of all features of the original task. In fact, the Cartesian product of each subset of the feature set can be a subspace.

Clustered subspace It is the state space in which a new discrete element is added to the features of a subspace. Adding this new element can be seen as assigning a cluster label to each state. The states of each clustered-subspace are denoted by two indices $s_{(x,c)}$, the first index x denotes a state in the subspace X and the second index c denotes the cluster label of that state.

Learning spaces It is the union of spaces used in the learning procedure. A learning space can be the state-space, one of the subspaces or one of the clustered-subspaces.

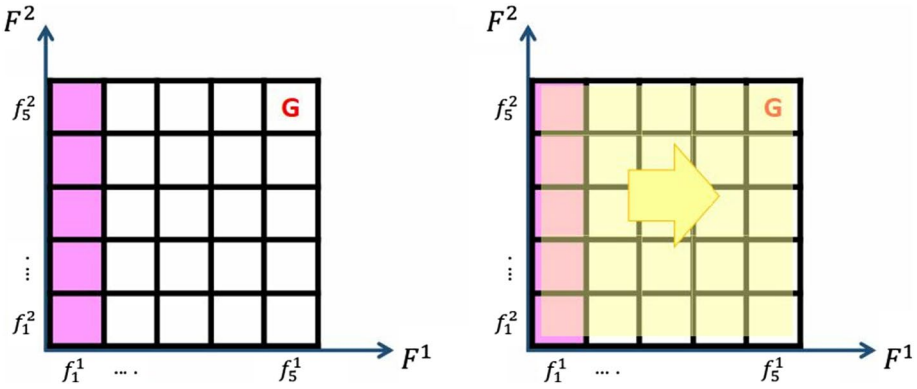


Fig. 1 A 5×5 maze for showing generalization in the subspaces. It has two subspaces according to two features $\{F^1\}$ and $\{F^2\}$. The agent can go up, down, right, or left to achieve the goal specified by "G". According to the goal location, if the agent learns the action right is the more preferable action in the columns, the learning speed can boost (Hashemzadeh et al. 2018)

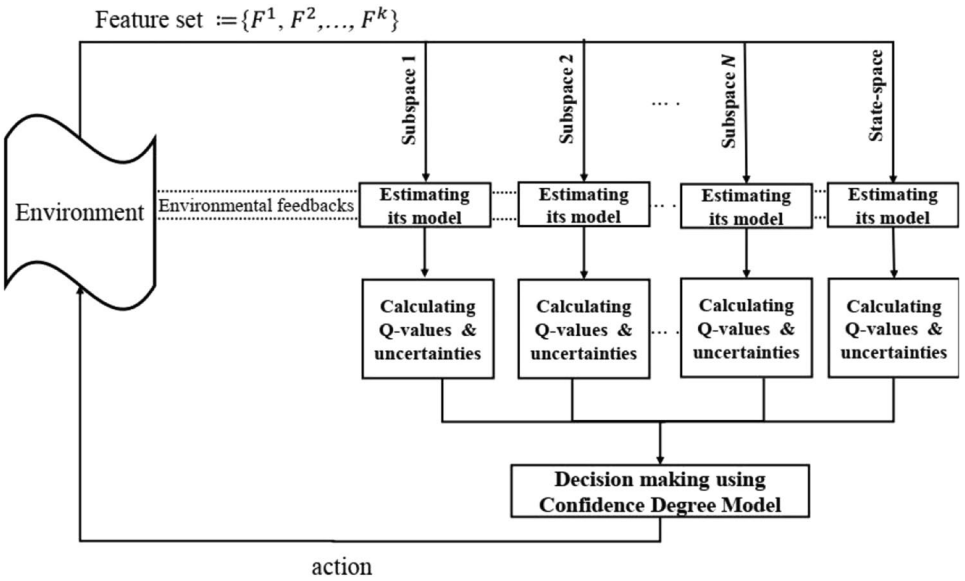


Fig. 2 A schematic overview of the MoBLeS framework (Hashemzadeh et al. 2018). The parameters of MDPs for both subspaces and state-space are updated. Then the Q-values and their uncertainties are calculated. Based on the Q-values and uncertainties, a weight is assigned to each subspaces and the state-space to determine their reliability which is used to choose more beneficial action

2 Overview of model based learning in the subspaces

The idea behind the generalization of the subspaces is illustrated in Fig. 1. The task has only two features, $\{F^1\}$ and $\{F^2\}$ corresponding to x and y positions. Therefore a subspace could be x position or y position. According to the goal location, if the agent learns that action *right* in subspace X or *up* in subspace Y are preferable, then it can use this knowledge when goes to a new state. MoBLeS tries to utilize this information by mapping the experiments of the states in the subspaces to the corresponding state in the original space.

Due to many to one mapping between the states of the original state-space and a state of a subspace, some information is lost for the subspaces and therefore the subspaces are partially observable to the task. This results in two consequents, the first one is faster learning in the subspaces due to their lower dimensions in comparison to the state-space and the second one is the PA issue in the states of the subspaces. MoBLeS uses uncertainties of the Q-values as a measure to determine how much the agent can rely on the subspaces experiences.

Figure 2 shows a schematic view of MoBLeS framework (Hashemzadeh et al. 2018). Based on the environmental feedbacks, the parameters of the MDPs of the state-space and subspaces are updated and the Q-values of all the states are calculated according to the Bellman equation. Then the upper bound and lower bound of the states are calculated to express the uncertainty over Q-values. Confidence Degree Model (CDM) is a measure in order to show the reliability of each subspace for the current state which is computed based on the q-values and their uncertainties.

The results show this framework can increase the learning speed in the early trials (Hashemzadeh et al. 2018). This is important, since many tasks are constrained in the number of the interactions and it is vital for them to decrease the number effectively. But it is not enough for later episodes. After

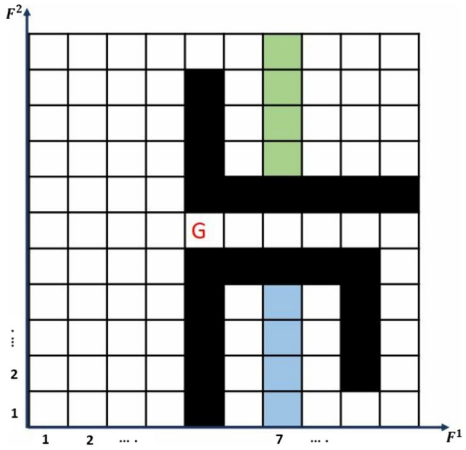


Fig. 3 A 10×10 maze having two features F^1 and F^2 and each of these features can be used as a subspace. Each column is a state of subspace $\{F^1\}$ and each row is a state of subspace $\{F^2\}$. The optimal policies of the states mapping to one state in a subspace are not necessarily similar. For example, consider the states of the 7th column that map to one state in the subspace $\{F^1\}$. The optimal policy of the blue part is apparently different from the optimal policy of the green part

some episodes the effect of the PA issue increases whereas the impact of the generalization decreases.

3 Method

The PA issue is the main limitation for using learning experiences in the subspaces. In fact, the generalization of the experiences in the subspaces is helpful as long as they do not misguide the agent from learning the optimal policy of the task. In MoBLeS, a state of a subspace corresponds to a number of states in the state-space without considering the policies of these states. Therefore, generalization of the subspaces usually is not very reliable and MoBLeS framework increases the speed usually in the early trials. To tackle this problem and to improve the generalization, we propose a framework for localizing the generalization of the subspaces by incorporating the policies of the states. The localization of the generalization can decrease the effect of the PA issue and the agent can rely more on the localized subspaces. This results in increasing the learning speed in compare to regular MoBLeS.

As an illustrative example of the aforementioned limitation, consider a simple maze environment of Fig. 3. This environment has two subspaces corresponding to its two features F^1 and F^2. Each state in the subspace $\{F^1\}$ presents a column in this maze, that means the whole column maps to a state in this subspace. As mentioned, the limitation of using subspaces in MoBLeS is caused by the disagreement of the optimal policies of states mapped to a state in a subspace. For example in the column with $F^1 = 7$, the appropriate actions in the green region are either going up or left, while the favorable actions in the blue region are either going down or right. Though the appropriate policies in these two regions are different, the subspace $\{F^1\}$ has only one optimal policy because all states in that column are mapped to one state in the subspace. For example, assume an agent has visited the states in the green region more often than the states in the blue region. If the agent go to the state in the blue region like the state (7, 2) the preferred actions based on the previous experiences are up or left which is the worse actions in this state.

To alleviate this limitation, we augment an extra dimension to the subspaces. This dimension is arisen from a clustering process on the state-space such that the states with similar policies go to the same cluster. Figure 4a shows that clustering can lead to more localized subspaces. In MoBLeS framework the whole column of a maze corresponds to a state of a subspace, while in the new framework each part of the column that corresponds to one cluster maps to one state in the subspace as shown with different colors in Fig. 4a.

Since only those states in the state-space that are in the same cluster maps to a state in the subspace, the PA issue becomes less if the states that go to one cluster have similar policies. Assume the maze of Fig. 3 is divided into several clusters as shown in Fig. 4b. It can be seen that the states of the 7th column are divided into three clusters as shown in Fig. 4c. We explained beforehand that in the MoBLeS framework, the subspace policy of the state (7, 2) is not reliable. This is not the case in the localized subspaces in the new framework. Because the preferred action of the state (7, 2) is based on the previous experiences of the blue region which has a consistent policy with the current state.

The previous example showed that if a good clustering is available, this can be incorporated in the localized subspaces to solve the limitation of the MoBLeS framework. A good clustering, however, is not known a priori and a strategy is needed to crease good subspaces based on the previous experiences. This strategy is explained in the remainder of this section.

3.1 C-MoBLeS algorithm

The algorithm of the new framework is called C-MoBLeS stemmed from the fact that the algorithm uses MoBLeS but for localized subspaces given by a clustering of the states in the state-space. The algorithm has three phases explained below. The description of these phases are given in details afterwards in different sections:

1. *Clustering state-space states* In this phase, a vector is assigned to each state of the state-space based on the previous experiences. Then, a clustering algorithm is applied on these vectors to cluster the states.

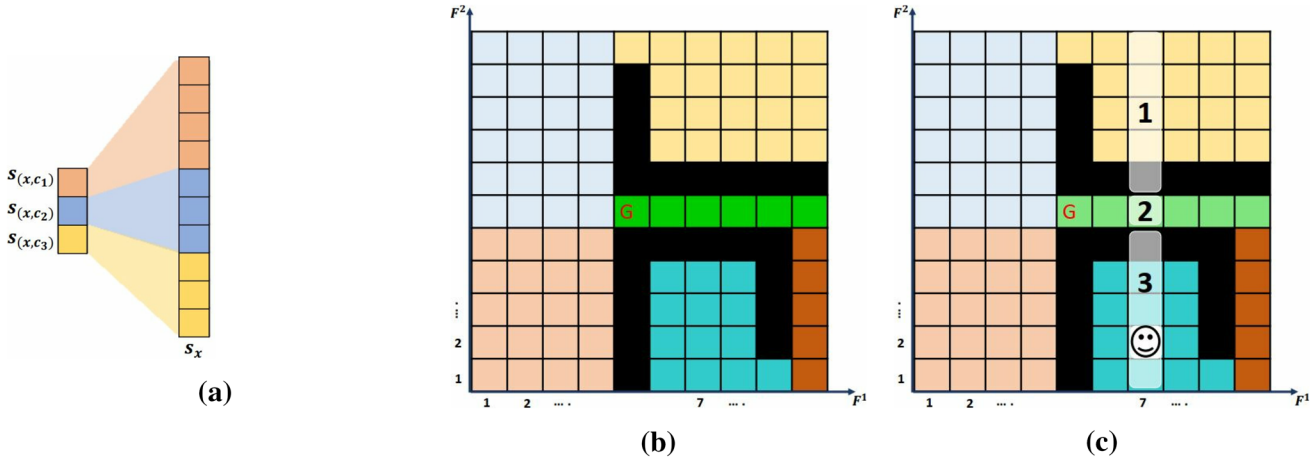


Fig. 4 Decreasing the effect of the PA issue in the subspaces by clustering states and using clustered-subspaces. **a** The states of the state-space mapping to one state in a subspace is shown in the right. These states now maps to three states in the clustered-subspace as shown in the left. Different colors correspond to different clusters, and the states in the same cluster and the same subspace maps to one state in the clustered-subspace. **b** An ideal clustering of the states in the maze of Fig. 3 such that states with similar optimal policies are assigned to one cluster. **c** Depicting the policies of the 7th column depend on its cluster. If the agent is in the state (7, 2), the policy suggested by the clustered-subspace is more useful than the normal subspace. It is because of the consistency of the policies in the blue partition that makes the policy suggested by the clustered-subspace nearly optimal. While the policy of the yellow part is completely converse and detrimental

2. *Estimating MDPs, Q-values and their confidence intervals* In this phase, the Q-values and the confidence interval of the Q-values is estimated for the state-space and the clustered-subspaces. The states of the state-space that map to a state in the clustered-subspace are those states that share the same feature of a subspace and go to the same cluster.
3. *Integrating the policies of clustered-subspaces and state-space* In this phase, a weight is assigned to the policy of each clustered-subspace and state-space based on their Q-values and uncertainties. This weight is used for selecting the best policy between the policies of clustered-subspaces or state-space.

3.1.1 Clustering states of the state-space

The goal in this step is putting similar states in terms of their policies into one cluster, so to obtain generalization by local experiences. Since we assume the agent does not have any prior knowledge of the task, it does not know a priori which states have similar policies to put them in one cluster. For this purpose, we propose an approach to cluster the states based on their previous experiences. Since at the beginning of learning the agent has little experience, the clusters are inaccurate. By gaining more experience, the clusters gradually will be more accurate.

For the clustering approach a representing vector is assigned to each state to measure the similarity. It contains features of the state and some extra values. For obtaining these extra values, the actions of a state are first ranked based on their estimated Q-values. The extra values are then just the ranks of each action.

The representing vector of a given state is given by

$$v = \left[f^1\, f^2 \ldots f^k\, \rho^1\, \rho^2 \ldots\, \rho^{|A|}\right]^T, \tag{1}$$

where f^i is the value of the feature F^i and ρ^i is the rank of the ith action.

The first part of the representing vector v forces adjacent states to go to one cluster, because we expect adjacent states to have similar policies. Second part of the v contains information of policies of each state and it encourages the states with similar policies to go to the same cluster. In fact, the second part is what we are are actually care about, but we use the first part as well to alleviate the lack of enough experiences at the early trials.

In the beginning of the learning, the clustering does not work well because we have little experience and the policies are far from optimal. By gaining more experience, the clustering approach does a better job in assigning states with similar policies to the same cluster. This is not a limiting factor for the proposed method for improving the performance of the MoBLeS framework. The main limitation of MoBLeS framework is the PA issue in the later stages, because in the beginning when the experiences is little, a good sub-optimal policy is valuable and MoBLeS works well at the early stages due to generalization in the subspaces even in the face of the PA issue. Our proposed framework solves the problem of PA issue in later stages, because by gaining more experiences the states with similar policies are grouped and

this leads to performance improvement in compare to the MoBLeS framework.

3.1.2 Estimating MDPs, Q-values and their confidence intervals

As explained beforehand, each learning space is an MDP space. The parameters of the state-space is updated like an usual MDP model. The states for the subspaces is enumerated by two terms, one is the index of the subspace and the other is the cluster value. Based on this definition of states, the parameters of the MDP model of the subspaces are updated using the common approaches used in model-based learning. Assume the agent is in $s_{(x,c)}$ and by performing action a goes to state $s_{(x',c')}$, then the approximation of the transition probability $\widehat{p}(s_{(x,c)}, a, s_{(x',c')})$ and the expected reward $\widehat{R}(s_{(x,c)}, a)$ in the clustered-subspaces are updated by:

$$\widehat{p}\left(s_{(x,c)}, a, s_{(x',c')}\right) = \frac{n\left(s_{(x,c)}, a, s_{(x',c')}\right) + 1}{n\left(s_{(x,c)}, a\right) + 1}, \tag{2}$$

$$\widehat{R}(s_{(x,c)}, a) = \frac{\widehat{R}\left(s_{(x,c)}, a\right) \times n\left(s_{(x,c)}, a\right) + r_i}{n\left(s_{(x,c)}, a\right) + 1}, \tag{3}$$

where $n(s_{(x,c)})$ is the number of times when the agent visits $s_{(x,c)}$ and $n(s_{(x,c)}, a, s_{(x',c')})$ is the number of times which the agent goes from $s_{(x,c)}$ to $s_{(x',c')}$ by taking the action a. The parameter r_i indicates the immediate reward through the transition.

Based on the estimated parameters of each MDP, the Q-values are estimated using policy evaluation (PE) algorithm (Puterman 1994). PE algorithm updates the value of $\widehat{Q}_\pi(s_{(x,c)}, a)$ using the neighboring states by the Bellman equation:

$$\widehat{Q}_\pi(s_{(x,c)}, a) = \widehat{R}(s_{(x,c)}, a) + \sum_{x'} \sum_{c'} \widehat{p}(s_{(x,c)}, a, s_{(x',c')}) \left[\gamma \max_{a'} \widehat{Q}_\pi(s_{(x',c')}, a')\right]. \tag{4}$$

For computing the upper and the lower bounds on the Q-values, the confidence intervals of the $\widehat{R}(.)$ and $\widehat{p}(.)$ are needed. These confidence intervals are computed using Hoeffding (Hoeffding 1963) and Weissman inequalities (Weissman et al. 2003), respectively.

Then the upper and lower bounds are estimated by UCRL-2 algorithm of Auer et al. (2009) (see also Algorithm 1 in Hashemzadeh et al. (2018) for the description of the algorithm with the notations similar to those used in the current paper). The algorithm solves the following equations for finding the upper Q_u and lower Q_l bounds:

$$Q_u(s_{(x,c)}, a) = \left(\widehat{R}(s_{(x,c)}, a) + \varepsilon_R\right) + \max_{\widetilde{P}(s_{(x,c)}, a) \in CI_p} \sum_{x'} \sum_{c'} \widetilde{p}(s_{(x,c)}, a, s_{(x',c')}) \left(\gamma \max_{a'} \widehat{Q}_u(s_{(x',c')}, a')\right), \tag{5}$$

$$Q_l(s_{(x,c)}, a) = \left(\widehat{R}(s_{(x,c)}, a) - \varepsilon_R\right) + \min_{\widetilde{P}(s_{(x,c)}, a) \in CI_p} \sum_{x'} \sum_{c'} \widetilde{p}(s_{(x,c)}, a, s_{(x',c')}) \left(\gamma \max_{a'} \widehat{Q}_l(s_{(x',c')}, a')\right), \tag{6}$$

where ε_R is the half of the confidence interval of the estimated reward $\widehat{R}(s_{(x,c)}, a)$, CI_p is the confidence interval of the estimated transition probability $\widehat{p}(s_{(x,c)}, a, s_{(x,c)'})$. Parameter $\widetilde{p}(s_{(x,c)}, a, s_{(x',c')})$ is an element of the probability vector $\widetilde{P}(s_{(x,c)}, a)$.

3.1.3 Integrating policies

Confidence degree model is a method developed in Hashemzadeh et al. (2018) for integrating the policies of the subspaces and the state-space. Very similar confidence degree model (CDM) is also used in this paper with a difference that instead of the subspaces we have clustered-subspaces here. The details of the CDM is explained in the following.

The goal of the CDM algorithm is to assign a weight to each clustered-subspace showing the reliability of that clustered-subspace for decision making. Finally, the policy of the clustered-subspace with the highest weight is selected at the current stage of the learning process if the weight is larger than one; otherwise, the decision of the state-space is chosen. This chosen policy is then used in an algorithm like ϵ-greedy for further exploring the state-space. The weight of each learning space called confidence degree (CD) is computed by the following equation:

$$CD(s_{(x,c)}, a) = \underbrace{f\left(\frac{|CI(s_j, a)|}{|CI(s_{(x,c)}, a)|}\right)}_{\text{Length Quantity}} \times \underbrace{g\left(\frac{|\widehat{Q}_\pi(s_j, a) - \widehat{Q}_\pi(s_{(x,c)}, a)|}{|CI(s_j, a) \cap CI(s_{(x,c)}, a)|}\right)}_{\text{Overlap-distance Quantity}}, \tag{7}$$

where $|CI(s_j, a)|$ is the length of the confidence interval of $\widehat{Q}(s_j, a)$ and $|CI(s_{(x,c)}, a)|$ is the length of the confidence interval of the $\widehat{Q}(s_{(x,c)}, a)$ and $f(.)$ is an increasing function. The length quantity measures the ratio of the uncertainties in

the estimated Q-values in the state-space and the clustered-subspaces. The more certain Q-value of a clustered-subspace suggests the more reliable policy.

The overlap-distance quantity assesses the consistency of the experiences in the clustered-subspaces with the experiences in state-space. If the intersection of the confidence intervals is large, we can conclude the agent has more common experiences in the $s_{(x,c)}$ and the s_j. The numerator of the overlap-distance quantity shows the difference between the estimated Q-values of a clustered-subspace state and its corresponding state in the state-space. So, smaller difference in the Q-values and large intersection in their confidence intervals shows the reliability of that clustered-subspace suggestion through decision making.

Therefore the CD of each clustered-subspace specifies the degree of the reliability to the current policy of each space. When the agent wants to choose an action, it assigns a probability to each action based on the CDM. The unnormalized probability in the CDM for the action a in s_j is given by:

$$Pr(s_j, a) = \begin{cases} p_\pi(s_{\alpha(s_j,a)}, a) & \text{if } CD(s_{\alpha(s_j,a)}, a) > 1 \\ p_\pi(s_j, a) & \text{if } CD(s_{\alpha(s_j,a)}, a) \leq 1, \end{cases} \tag{8}$$

where $\alpha(s_j, a) = \arg\max_{x \in S_j} CD(s_{(x,c)}, a)$, wherein S_j is the set of all subspaces and c is the cluster of the s_j. The parameter $p_\pi(., a)$ is the probability of choosing the action a based on the decision policy π in the corresponding states.

The Algorithm 1 shows an overview of the C-MoBLeS approach.

Algorithm 1 Clustered-Model based learning with subspaces

function C-MoBLeS($\pi, \theta, \delta_p, \delta_R$)

Require: π is the learning policy, θ is a small positive number for the stopping criterion used in three recursive algorithms for computing Q-values (PolicyEvaluation), upper bounds on the Q-values (OptimisticValue), and lower bounds on the Q-values (PessimisticValue), $LenR$ is a vector containing the length of the rewards, parameters $1 - \delta_R \in [0, 1]$ and $1 - \delta_p \in [0, 1]$ are the confidence coefficients.

initialize $Q(.) = 0,\ Q_u(.) = 0,\ Q_l(.) = 0,\ \widehat{R}(.) = 0,\ \widehat{p}(.) = \frac{1}{m(.)}$ ▷ $m(.)$ is a number of neighboring states if available, otherwise $m(.) = |S|$.

Initialize clusters of the states.
repeat for each episode
  Initialize s_j
  repeat in each step of episode
    $a = \text{CDM}(Q, Q_u, Q_l, s_j, \pi, c)$ ▷ Choose action a by using CDM. c is the cluster of s_j.
    Take action a, observe s'_j and $r(s_j, a, s'_j)$.
    Update the estimated model in the state-space and in the subspaces by using relations 2 and 3.
    $\widehat{Q}_\pi(s_j, a) = \widehat{R}(s_j, a) + \sum_{s'_j} \widehat{p}(s_j, a, s'_j) \left[\gamma \max_{a'} \widehat{Q}_u(s'_j, a')\right]$ ▷ Update Q-values, full backup Van Seijen and Sutton [2013].
    Compute an estimate for the confidence interval of $\widehat{Q}_\pi(s_j, a)$ using equations 5 and 6.
    for all $x \in S_j$ **do** ▷ S_j is the set of the subspaces
      $\widehat{Q}_\pi(s_{(x,c)}, a) = \widehat{R}(s_{(x,c)}, a) + \sum_{s'_{(x,c')}} \widehat{p}(s_{(x,c)}, a, s'_{(x,c')}) \left[\gamma \max_{a'} \widehat{Q}_u(s'_{(x,c')}, a')\right]$
      Compute an estimate for the confidence interval of $\widehat{Q}_\pi(s_{(x,c)}, a)$ using equations 5 and 6.
    end for
    $s_j = s'_j$
  until end of episode
  Cluster states of the state-space and save the index of the cluster of each state in C.
  Build clustered-subspaces using C
  Update $\widehat{p}$ and $\widehat{R}$ for the clustered-subspaces
  for all the subspaces and the state-space **do**
    $Q = \text{PolicyEvaluation}(\pi, \theta)$
    $Q_u = \text{OptimisticValue}(n, \widehat{p}, \widehat{R}, \text{LenR}, \delta_R, \delta_p, \gamma, \theta)$. ▷ Based on Algorithm 1 in Hashemzadeh et al. [2018].
    $Q_l = \text{PessimisticValue}(n, \widehat{p}, \widehat{R}, \text{LenR}, \delta_R, \delta_p, \gamma, \theta)$. ▷ Based on Algorithm 1 in Hashemzadeh et al. [2018].
  end for

Table 1 Values of the required parameters for algorithms

Parameter	Value	Parameter description
γ	0.9	Discount factor
ϵ	0.1	ϵ in the $\epsilon-greedy$ policy
θ	$\frac{0.01}{1+\log(\#episode)}$	Stopping criterion of the PE algorithm and the algorithms for computing the upper and lower bounds of the Q-values
$1-\delta_R$	0.9	Confidence coefficient of the estimated rewards in the state-space and subspaces
$1-\delta_p$	0.9	Confidence coefficient of the estimated transition probabilities in the state-space and subspaces

4 Experimental results

The thorough performance comparison between MoBLeS and other experience generalization techniques was done in Hashemzadeh et al. (2018). It was reported there that the model-free methods are slower than the model-based ones. Since this paper is on improving the performance of MoBLes and to avoid unnecessary clutter, we concentrate of comparing the performance of C-MoBLeS against MoBLeS, an incorporation of MoBleS in R-Max (R-Max + MoBLeS) and pure Model-Based framework (MB).[1]

Average accumulated reward after each episode is reported as a measure of learning speed for different methods. The policy of the agent is ϵ-greedy with constant ϵ. The parameter γ is the discounting factor for updating the Q-values in the Bellman equations. The confidence coefficients $1-\delta_R$ and $1-\delta_p$ are needed for computing the confidence intervals in MoBLeS and C-MoBLeS. The values of the parameters are set according to the Table 1. In the simulations, the results with the best set of the parameters are shown. Also, the f and g functions of Eq. (7) are defined as the following:

$$f(t) = \begin{cases} 0 & t \leq 1 \\ t & t > 1 \end{cases}, \tag{9}$$

$$g(t) = \begin{cases} \text{-erfinv}(2t-1) & t \leq 1/2 \\ 0 & t > 1/2, \end{cases} \tag{10}$$

where f is an increasing function and g is a decreasing function wherein *erfinv* is the inverse error function.

4.1 Experimental tasks

We assessed the performance of the C-MoBLeS in two types of tasks. The first set is some 2D maze environments and the second set is two environments of Real-Time Strategy Game StarCraft:Broodwar (SC:BW).

2D maze environments The first set is some 2D maze environments with different locations of barriers as shown in Fig. 5. In each state, the agent can go either up, right, left, or down. All of the examples are stochastic: if the agent performs an action it goes to the expected state by the probability of 0.9 and it goes to a random neighboring state by the probability of 0.1. After doing an action, the agent takes an immediate reward coming from the truncated Gaussian mixture densities of Table 2. If the agent collides with an obstacle its position is not changed. The episode of learning ends when either the agent goes to the goal state or the number of the steps reaches a threshold number.

As it can be seen in Fig. 5, all environments are room-like and some environments have additional obstacles for accentuating the problem of PA issue. These additional obstacles in the environment 3 are semi-random and in the environment 4 are vertical walls. Additionally, the environment 4 has two goals for making the task more difficult.

Real-time strategy game StarCraft:BroodWar The second set of tasks is two environments of SC:BW as shown in Fig. 6. For this task we used the simulated combat maneuvering model of SC:BW of Hashemzadeh et al. (2018). There are two examples of SC:BW, a RL agent against one enemy and a RL agent against three enemies. In this case the state-space comprises of the locations of all of the agents in a 2D grid space, their hit points and weapon ranges. The game is continued until either the hit points of the RL agent or the hit points of all the enemies become zero. Otherwise, an episode ends up when the number of the steps meets the threshold. The action space for the RL agent is shooting to a target enemy, moving toward a target enemy, or retreating. If the RL agent goes to the weapon range of an enemy, its hit points decrease by one number. The reward function is according to Table 3, where HP^i_t and HP^i_{t+1} are the hit points of the ith enemy and HP_t and HP_{t+1} are the agent hit points at time step t and $t+1$, respectively. r_a corresponds to the reward of one of the possible cases.

4.2 Results of the clustering approach

The clustering approach used in this paper is an agglomerative hierarchical clustering called UPGMA (unweighted Pair

[1] The codes for reproducing the results of this section can be downloaded from https://github.com/MHashemzadeh/C-MoBLeS.git.

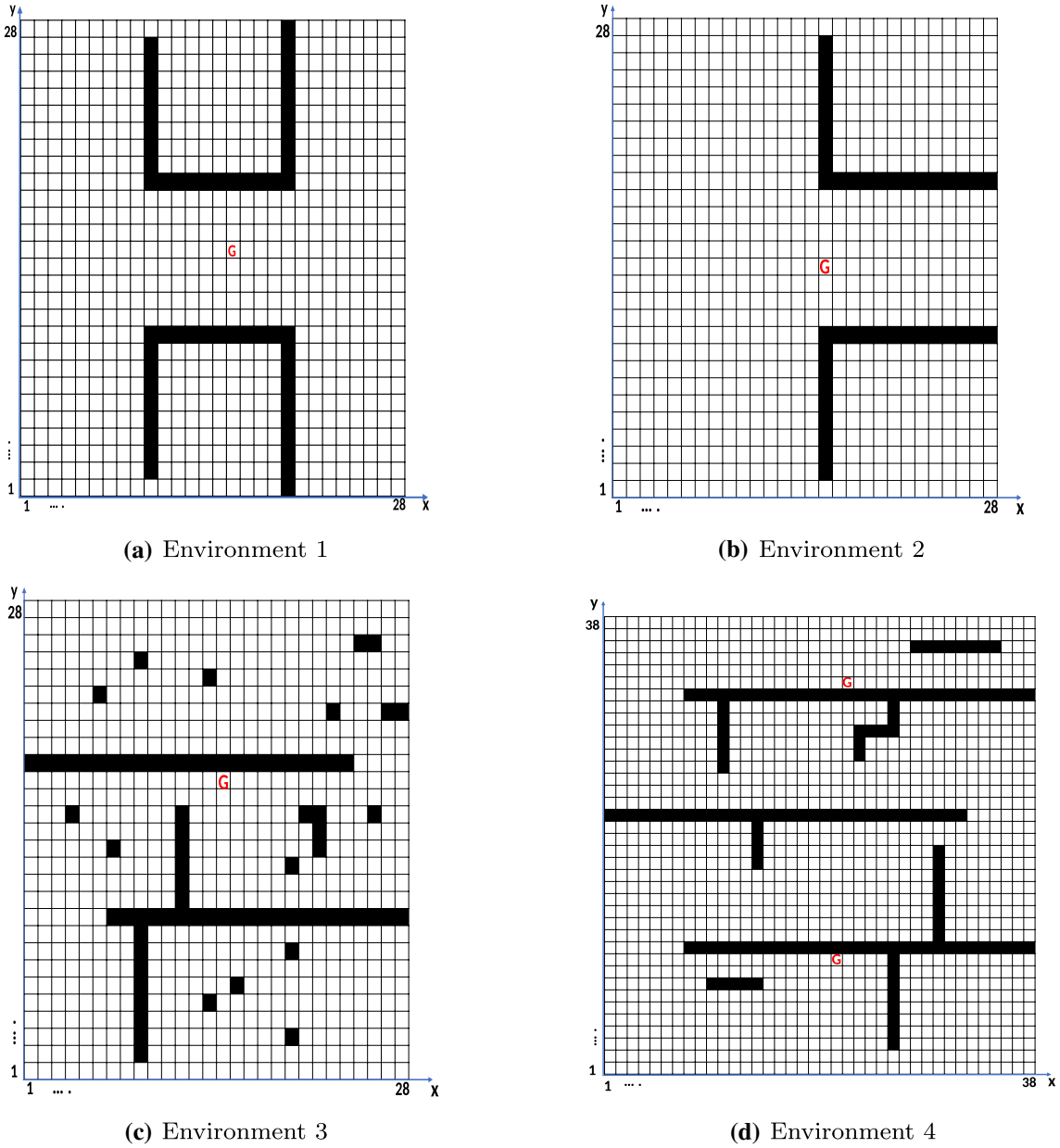


Fig. 5 Four different maze environments. The goal is determined by red "G" in the figures. The agent learns the average shortest path from starting state to the goal. Also the starting state is selected randomly in the environments. The agent can go either up, down, right or left

Group Method with Arithmetic Mean) with Euclidean distance as its metric (Michener and Sokal 1957). The clustered learning space is constructed as explained in Sect. 3.1.1. Before applying the clustering, each feature is normalized by dividing to its maximum value.

Clustering is performed only after each learning episode. Agglomerative clustering approaches start with single states as clusters and consequently merges clusters to construct larger clusters. Merging clusters stops if the distance between clusters is bigger than a threshold. To determine this threshold, we run the clustering approach until all states go to one cluster and the half of the maximum distance between merged clusters is used as the threshold.

The unvisited states are assigned to a cluster called *unvisited cluster*. The result of the clustering approach for the first and third environments after 10, 20, 50, and 100 episodes are shown in Figs. 7 and 8, respectively. Each color in the figures depicts the states belonging to a particular cluster. As it can be seen in the figures, the clustering approach approximately puts those states in one cluster that have similar policies.

Fig. 6 Real-Time Strategy Game StarCraft:Broodwar (SC:BW) environment (Hashemzadeh et al. 2018). It is an abstract combat maneuvering model of SC:BW between a RL agent unit and some enemy units

4.3 Performance results

The left plots of the Figs. 9 and 10 show the average accumulated rewards of C-MoBLeS and the other three methods for the Figs. 5a and 6 with one enemy respectively. The right plot shows the normalized weights of each learning spaces for both of the C-MoBLeS and MoBLeS methods. As it can be seen, C-MoBLeS receives more rewards after several episodes and can achieve higher accumulated rewards than MoBLeS.

In complex environments the PA issue is more problematic. In fact after some trials in MoBLeS the detriment of PA issue outweighs the benefit of the generalization and then it cannot work effectively. But the additional dimension in C-MoBLeS localizes the generalization of the subspaces resulting in clustering state-space into categories which contain states with similar policies. The right plot shows the weights of the subspaces in C-MoBLeS are more than corresponding weights in MoBLeS. The weights determine that the agent relies on the subspaces more than the state-space not only in the beginning of the learning, but also in later while in MoBLeS the weights of the subspaces decay very soon.

In Tables 4 and 5, the performance of C-MoBLeS for the maze and SC:BW environments are shown respectively. As shown in these tables, C-MoBLeS can weaken the negative effect of the PA issue on the learning and boost the benefit of the generalization. Therefore, the proposed C-MoBLeS algorithm attains much better results than the approaches without clustering, that is R-Max + MoBLeS and MoBLeS.

5 Discussion and conclusion

C-MoBLeS framework is a model-based reinforcement learning exploiting the generalization of the learning experiences by taking use of both subspaces learning and a clustering approach. The proposed method tries boosting learning speed more than MoBLeS (Hashemzadeh et al. 2018) in terms of required experiences. The main purpose of C-MoBLeS framework is reducing the disadvantageous of the PA issue and increasing the advantageous of the generalization of learning in the subspaces more than before. For doing so, we employ a clustering approach in the state-space in order to categorize states with similar policies in one cluster. When the agent wants to choose an action, only the experiences of those states in a subspace that are in the same cluster as the current state are taken into account. In fact by this approach, we localize the experiences in the subspaces.

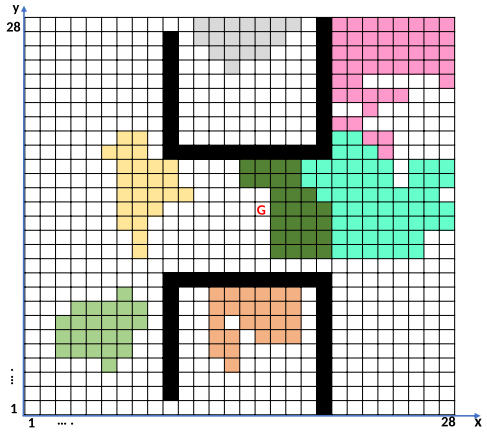
We demonstrated the performance of C-MoBLeS on two tasks, four mazes with different structures of barriers and two environments of combat maneuvering model of SC:BW which is a complex benchmark in reinforcement learning.

Table 2 Functions of the immediate rewards

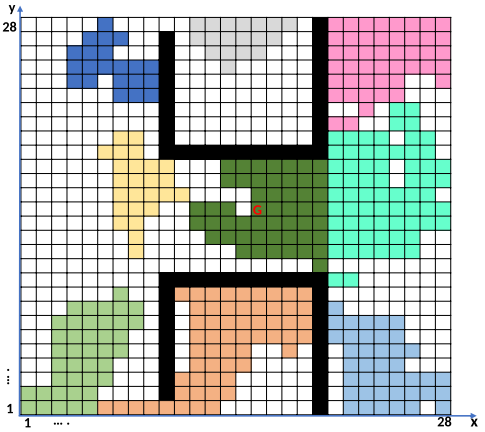
Reward description	Value
Colliding with a wall	$\sim \frac{1}{3}\mathcal{N}(-11.5, 0.2) + \frac{2}{3}\mathcal{N}(-10.5, 0.3),\ \in [-12, -10]$
Reaching the goal	$\sim N(+10, 0.02),\ \in [9.5, 11.5]$
Each step reward	$\sim \frac{1}{3}\mathcal{N}(-1.5, 0.2) + \frac{2}{3}\mathcal{N}(-0.5, 0.3),\ \in [-2, 0]$

Table 3 Distribution of rewards for SC:BW tasks

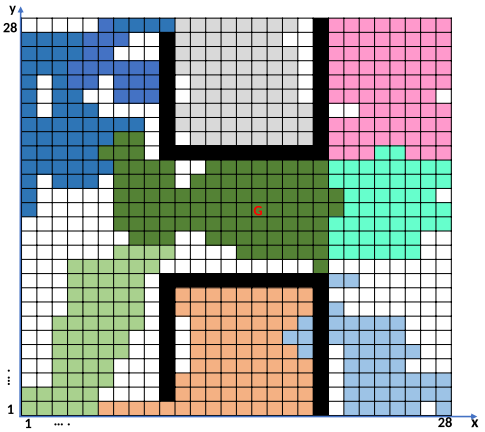
Reward description	Value
Shooting to a killed enemy	$\sim N(-10, 0.1),\ \in [-11, -9]$
Colliding with obstacles	$\sim \frac{1}{3}\mathcal{N}(-1.5, 0.2) + \frac{2}{3}\mathcal{N}(-0.5, 0.3),\ \in [-2, 0]$
Reaching the goal	$\sim \frac{1}{3}\mathcal{N}(97, 0.2) + \frac{2}{3}\mathcal{N}(100, 0.15),\ \in [95, 105]$
Each step reward	$\sim 10 \times \left(\sum_{i=1}^{\#\text{enemies}} \left(HP_t^i - HP_{t+1}^i\right) - \left(HP_t - HP_{t+1}\right)\right) + r_a - 1$



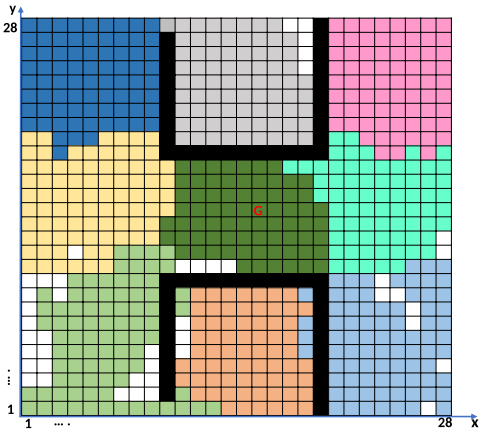
(a) Clustered states after 10 episodes.



(b) Clustered states after 20 episodes.



(c) Clustered states after 50 episodes.



(d) Clustered states after 100 episodes.

Fig. 7 Clustered states of the first environment after 10, 20, 50, and 100 episodes. Each cluster is depicted by a specific color. The white color displays *unvisited cluster*. After several episodes, the primary clusters are extended and transformed. In the later episodes, states with approximately similar policy are gathered into a cluster

The results showed learning in C-MoBLeS is faster than MoBLeS after beginning stages. Even in the complex task, it works dramatically well with a significant difference in the learning rate of the other methods. Although MoBLeS is able to increase the learning speed in the early stages of learning but it is unable to keep this later in the learning, while C-MoBLeS tries to keep higher learning rate not only in the first trials but also after that.

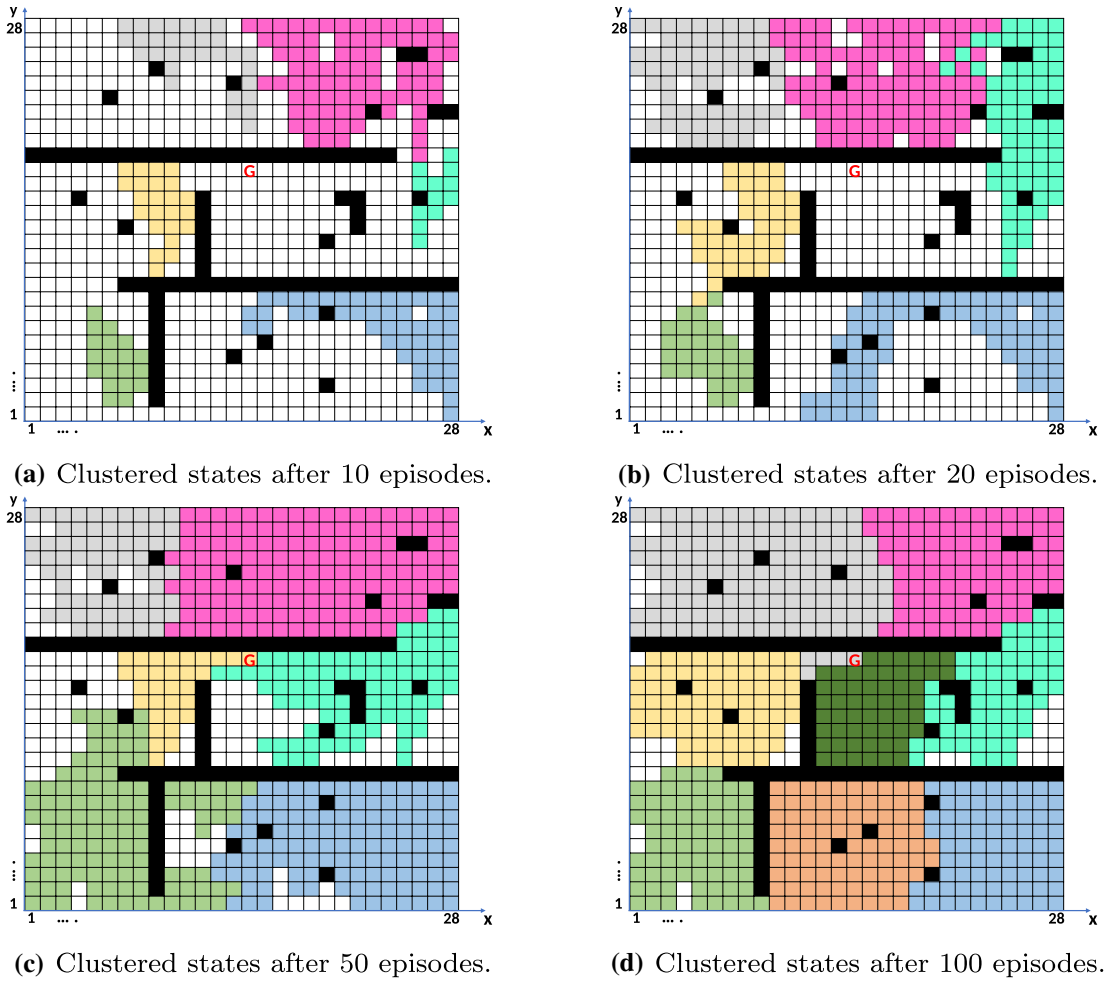
In the proposed framework, Cartesian product of each subspace and clustering index constitutes an MDP. The number of the parameters in MDPs grows linearly with the number of the clusters. Since the dimensionality of the subspaces is smaller than the state-space, the number of their states is also smaller and consequently their memory and computational complexity is lower. In the C-MoBLeS framework these complexities are multiplied by the number of the clusters and is higher than no clustering is used in MoBLeS.

An interesting point was in the simulations that we saw after a number of trials, clusters are not changed. This means that the policy in each cluster is approximately stable and cannot help the agent further. That can be a clue to stop using generalization in the subspaces and can be a good time to learn only based on the state-space.

In dynamic tasks when the goal changes, the clusters and policies should change when the goal changes. We think that in such tasks, our approach should also show some benefits, because the subspaces are lower-dimensional and can adapt to new changes of the environment faster.

There are some approaches one can continue in the future:

- Representing vector for clustering can be specialized for each subspace. In this work we employ a common set of elements to all subspaces. But according to the properties



(a) Clustered states after 10 episodes.

(b) Clustered states after 20 episodes.

(c) Clustered states after 50 episodes.

(d) Clustered states after 100 episodes.

Fig. 8 Clustered states of the second environment after 10, 20, 50, and 100 episodes

of each subspace, we can extract some more informative elements for its clustering.

- C-MoBLeS is a model-based learning and can be extended to model-free learning which has lower memory and computational complexity.
- In order to reduce the demand of memory and computational complexity, a procedure can be devised to eliminate the less informative subspaces and just continue learning by more helpful subspaces.
- C-MoBLeS can be adapted for learning in tasks with noisy sensors. Through this framework the agent can rely more on more accurate sensors.
- C-MoBLeS can be developed for continuous reinforcement learning.
- One can incorporate deep learning architectures in the proposed method for very complex tasks with large number of states.
- One potential approach instead of clustering over the subspaces is to use evolving fuzzy systems (Angelov 2014; Angelov and Kasabov 2005). Then the value of the confidence degree is calculated with respect to the fuzziness of the states.
- The clustering approach in C-MoBLeS can be enhanced with evolving clustering (Hyde et al. 2017; Baruah and Angelov 2012, 2013).

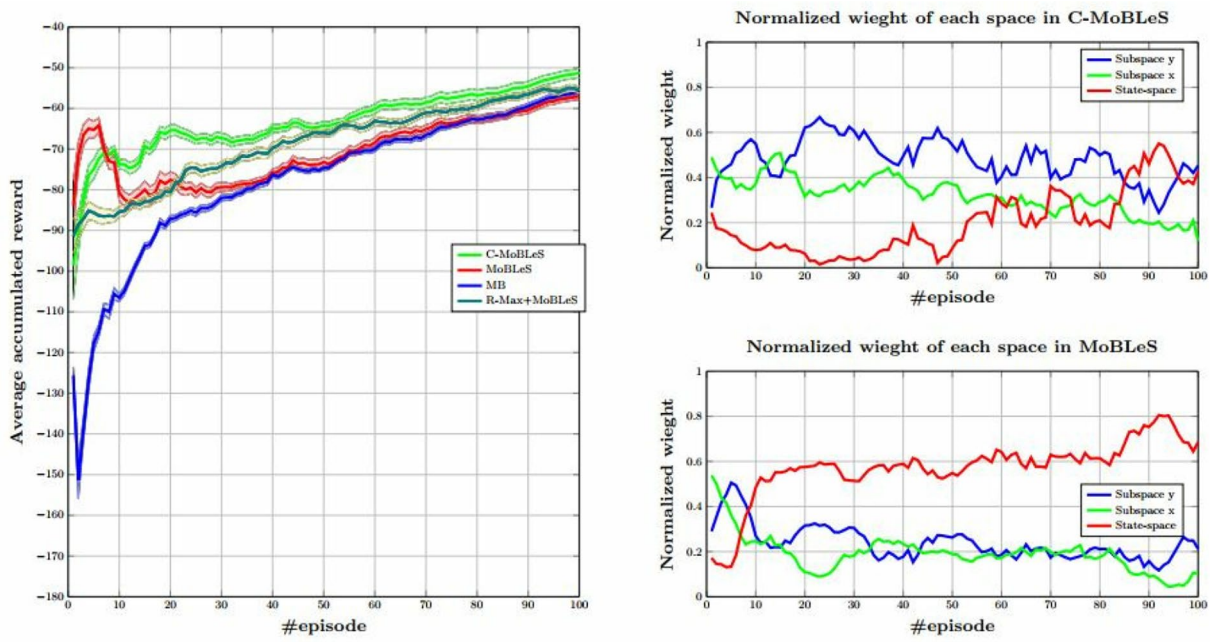


Fig. 9 Average accumulated rewards of C-MoBLeS, MoBLeS, R-Max + MoBLeS and MB in the first environment. Left plot shows average accumulated rewards as a function of the number of the episodes. The shaded area are the standard error of each plot. Right plots show normalized weights of each space in C-MoBLeS and MoBLeS. These weights are smoothed by rectangular window of length 5

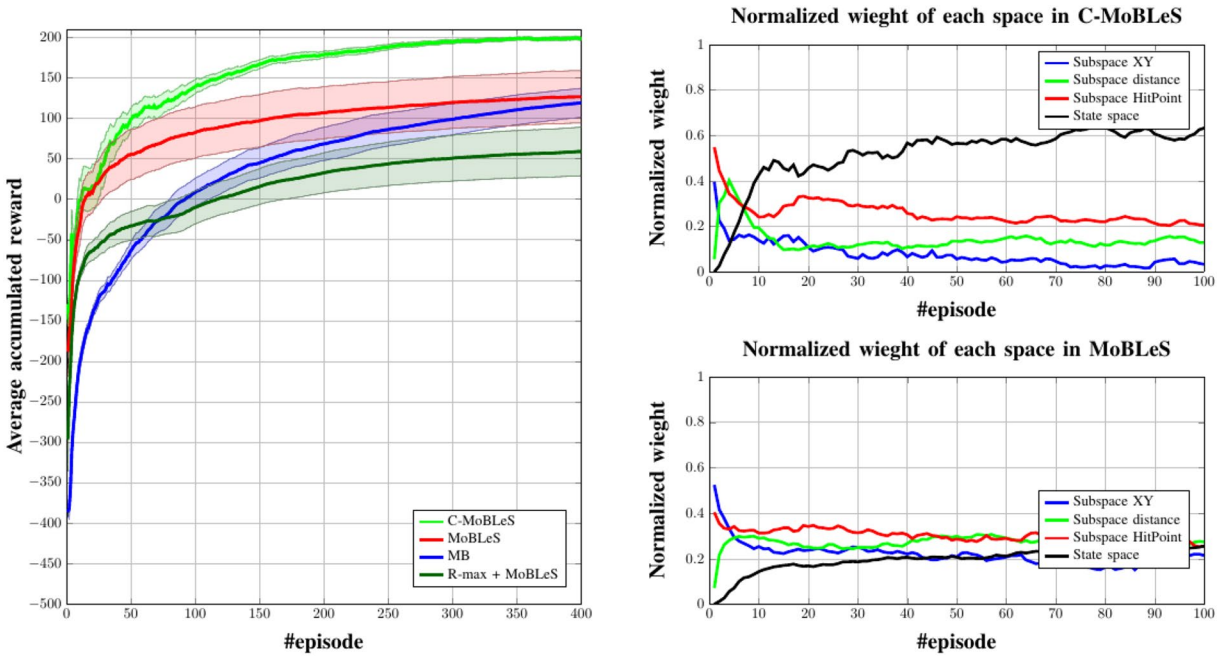


Fig. 10 Result of the first environment for SC:BW combat between the RL agent and an enemy. The description of the plots are the same as Fig. 9

Table 4 Average accumulated rewards of C-MoBLeS, MoBLeS, R-Max + MoBLeS and MB in Maze task

Environment	Method	10	20	40	60	80	100
Environment 1	C-MoBLeS	−73.1	−65.2	−64	−60	−56.5	−51
	MoBLeS	−80	−77.9	−75.5	−68	−63	−58.2
	R-Max + MoBLeS	−86.6	−80.1	−69.5	−64	−60	−55.8
	MB	−106.7	−87.3	−76.1	−70.1	−63.6	−57.2
Environment 2	C-MoBLeS	−77	−69.1	−62.4	−56.6	−52.5	−46.8
	MoBLeS	−85.3	−83.9	−72.1	−66	−57.4	−50.1
	R-Max + MoBLeS	−83	−80.1	−70.5	−59	−54	−47.6
	MB	−114.2	−61.54	−73.7	−66.2	−56	−49.7
Environment 3	C-MoBLeS	−86.6	−85	−80.3	−75	−68	−65.8
	MoBLeS	−96.2	−98	−88.2	−80.1	−70	−65.8
	R-Max + MoBLeS	−98	−96.3	−84	−77.5	−69	−62.6
	MB	−114.6	−103.5	−86.2	−78.4	−68	−59.7
Environment 4	C-MoBLeS	−89.6	−84	−79.2	−72.5	−69.6	−68
	MoBLeS	−96.2	−91.8	−88.4	−85.1	−78.8	−75
	R-Max + MoBLeS	−110.6	−98.8	−87	−82.3	−75.1	−69.8
	MB	−134	−117	−95.4	−88.2	−80.5	−78.3

The average accumulated rewards for the episode numbers 10, 20, 40, 60, 80, 100 for both one enemy and three enemies are shown

Table 5 Average accumulated rewards of C-MoBLeS, MoBLeS, R-Max + MoBLeS and MB in SC:BW task

Environment	Method	20	50	100	150	200	300	400
One Enemy	C-MoBLeS	16	98.7	138.9	166.6	178	194.2	199.1
	MoBLeS	19	54.5	83.3	97.1	106.6	119	126.2
	R-Max + MoBLeS	−53.2	−41.1	−9.8	16.4	29	45.4	58
	MB	−143.2	−61.5	9.7	45.4	67.9	98.4	118.4
Three Enemies	C-MoBLeS	−200.6	−183	−155.1	−140.1	−132.6	−125.2	−117.9
	MoBLeS	−228	−236.3	−211.1	−190.9	−177	−161.5	−151
	R-Max + MoBLeS	−227	−189	−162.2	−151.3	−149	−143	−140
	MB	−372.6	−321	−258	−228.2	−210	−186.3	−173.2

The average accumulated rewards for the episode numbers 20, 50, 100, 150, 200, 300, 400 for both one enemy and three enemies are shown

References

Ackley DH, Littman ML (1990) Generalization and scaling in reinforcement learning. In: Advances in neural information processing systems 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 550–557

Angelov P (2014) Outside the box: an alternative data analytics framework. J Autom Mobile Robot Intell Syst 8(2):29–35

Angelov P, Kasabov N (2005) Evolving computational intelligence systems. In: IEEE International Conference on Fuzzy Systems. IEEE, Brisbane, QLD, Australia, pp 76–82

Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) A brief survey of deep reinforcement learning. arXiv:1708.05866

Auer P, Jaksch T, Ortner R (2009) Near-optimal regret bounds for reinforcement learning. Adv Neural Inf Process Syst 21:89–96

Baruah RD, Angelov P (2012) Evolving local means method for clustering of streaming data. In: IEEE International Conference on Fuzzy Systems. IEEE, Brisbane, QLD, Australia, pp 1–8

Baruah RD, Angelov P (2013) Dec: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models. IEEE Trans Cybern 44(9):1619–1631

Bruce J, Sünderhauf N, Mirowski P, Hadsell R, Milford M (2017) One-shot reinforcement learning for robot navigation with interactive replay. arXiv:1711.10137

Daee P, Mirian MS, Ahmadabadi MN (2014) Reward maximization justifies the transition from sensory selection at childhood to sensory integration at adulthood. PLoS One 9(12):e115926

Gu S, Holly E, Lillicrap T, Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp 3389–3396

Gupta A, Devin C, Liu Y, Abbeel P, Levine S (2017) Learning invariant feature spaces to transfer skills with reinforcement learning. arXiv:1703.02949

Hashemzadeh M, Hosseini R, Ahmadabadi MN (2018) Exploiting generalization in the subspaces for faster model-based reinforcement learning. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2018.2869978

Hausknecht M, Stone P (2015) Deep recurrent Q-learning for partially observable MDPs. arXiv:1507.06527

Hemminghaus J, Kopp S (2017) Towards adaptive social behavior generation for assistive robots using reinforcement learning. In: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction. ACM, New York, pp 332–340

Hoeffding W (1963) Probability inequalities for sums of bounded random variables. J Am Stat Assoc 58(301):13–30

Hyde R, Angelov P, MacKenzie AR (2017) Fully online clustering of evolving data streams into arbitrarily shaped clusters. Inf Sci 382:96–114

Kohl N, Stone P (2004) Policy gradient reinforcement learning for fast quadrupedal locomotion. IEEE Int Conf Robot Autom (ICRA) 3:2619–2624

Kulkarni TD, Narasimhan K, Saeedi A, Tenenbaum J (2016) Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In: Advances in neural information processing systems, vol 29. Curran Associates, Inc., New York, pp 3675–3683

Lai M (2015) Giraffe: using deep reinforcement learning to play chess. arXiv:1509.01549

Li Z, Zhao T, Chen F, Yingbai H, Chun-Yi S, Fukuda T (2018) Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator. IEEE/ASME Trans Mech 23(1):121–131

Lonnie C (1992) Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In: Proceedings of the Tenth National Conference on Artificial Intelligence. AAAI Press, San Jose, California, pp 183–188

Michener CD, Sokal RR (1957) A quantitative approach to a problem in classification. Evolution 11(2):130–162

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

Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp 1928–1937

Modares H , Ranatunga I, AlQaudi B, Lewis FL, Popa DO (2017) Intelligent human–robot interaction systems using reinforcement learning and neural networks. In: Trends in control and decision-making for human–robot collaboration systems. Springer, Switzerland, pp 153–176

Ng AY, Coates A, Diel M, Ganapathi V, Schulte J, Tse B, Berger E, Liang E (2006) Autonomous inverted helicopter flight via reinforcement learning. In: Experimental robotics IX. Springer, Berlin, pp 363–372

Oh J, Guo X, Lee H, Lewis RL, Singh S (2015) Action-conditional video prediction using deep networks in atari games. In: Advances in neural information processing systems, vol 28. Curran Associates, Inc., New York, pp 2863–2871

Popov I, Heess N, Lillicrap T, Hafner R, Barth-Maron G, Vecerik M, Lampe T, Tassa Y, Erez T, Riedmiller M (2017) Data-efficient deep reinforcement learning for dexterous manipulation. arXiv:1704.03073

Puterman ML (1994) Markov decision processes: discrete stochastic dynamic programming. Wiley, New York

Rajeswaran A, Kumar V, Gupta A, Schulman J, Todorov E, Levine S (2017) Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. arXiv:1709.10087

Richard SS, Andrew GB (2018) Reinforcement learning: an introduction. MIT press, Cambridge

Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. In: International Conference on Machine Learning. PMLR, France, pp 1889–1897

Shoeleh F, Asadpour M (2017) Graph based skill acquisition and transfer learning for continuous reinforcement learning domains. Pattern Recogn Lett 87:104–116

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489

Singh S, Litman D, Kearns M, Walker M (2002) Optimizing dialogue management with reinforcement learning: experiments with the NJFun system. J Artif Intell Res 16:105–133

Stadie BC, Levine S, Abbeel P (2015) Incentivizing exploration in reinforcement learning with deep predictive models. arXiv:1507.00814

Stone P, Veloso M (2000) Multiagent systems: a survey from a machine learning perspective. Auton Robots 8(3):345–383

Sutton RS (1995) Generalization in reinforcement learning: successful examples using sparse coarse coding. In: Advances in neural information processing systems 8. MIT Press, Cambridge, MA, pp 1038–1044

Synnaeve G, Nardelli N, Auvolat A, Chintala S, Lacroix T, Lin Z, Richoux F, Usunier N (2016) Torchcraft: a library for machine learning research on real-time strategy games. arXiv:1611.00625

Tesauro G (1995) Temporal difference learning and TD-Gammon. Commun ACM 38(3):58–68

Van Seijen H, Sutton RS (2013) Efficient planning in mdps by small backups. In: Proceedings of the international conference on machine learning. JMLR, Atlanta, GA, USA, pp 361–369

Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, Makhzani A, Küttler H, Agapiou J, Schrittwieser J et al (2017) Starcraft II: a new challenge for reinforcement learning. arXiv:1708.04782

Wang W, Hao J, Wang Y, Taylor M (2018) Towards cooperation in sequential prisoner's dilemmas: a deep multiagent reinforcement learning approach. arXiv:1803.00162

Weissman Tsachy, Ordentlich Erik, Seroussi Gadiel, Verdu Sergio, Weinberger Marcelo J (2003) Inequalities for the L_1 deviation of the empirical distribution. Technical report, HP Laboratories Palo Alto

Zhu Y, Mottaghi R, Kolve E, Lim JJ, Gupta A, Fei-Fei L, Farhadi A (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp 3357–3364