Control Theory and Technology (2021) 19:529–537
https://doi.org/10.1007/s11768-021-00069-5

RESEARCH ARTICLE

Sequencing of multi-robot behaviors using reinforcement learning

Pietro Pierpaoli[1] · Thinh T. Doan[2] · Justin Romberg[1] · Magnus Egerstedt[3]

Received: 29 May 2021 / Revised: 19 August 2021 / Accepted: 31 August 2021 / Published online: 7 December 2021

Abstract
Given a collection of parameterized multi-robot controllers associated with individual behaviors designed for particular tasks, this paper considers the problem of how to sequence and instantiate the behaviors for the purpose of completing a more complex, overarching mission. In addition, uncertainties about the environment or even the mission specifications may require the robots to learn, in a cooperative manner, how best to sequence the behaviors. In this paper, we approach this problem by using reinforcement learning to approximate the solution to the computationally intractable sequencing problem, combined with an online gradient descent approach to selecting the individual behavior parameters, while the transitions among behaviors are triggered automatically when the behaviors have reached a desired performance level relative to a task performance cost. To illustrate the effectiveness of the proposed method, it is implemented on a team of differential-drive robots for solving two different missions, namely, convoy protection and object manipulation.

Keywords Multi-robot systems · Reinforcement learning · Distributed control

1 Introduction

In the multi-robot literature, significant contributions have been made towards the design of distributed controllers that achieve particular, targeted objectives or tasks in a coordinated manner, such as meeting at a common location, assembling a particular geometry, covering an area, or patrolling a perimeter, e.g., [1–4]. However, more complex tasks typically require the robots to be able to execute coordinated motions that go beyond what any one such targeted controller, or behavior, could achieve, as was illustrated in [5] for a complex scenario whereby a team of robots were asked to search for and subsequently secure a building. To add to the complication, oftentimes not everything about the environment (or even the task itself) is fully known a priori. For example, consider a team of robots tasked with moving a box between two points without previous knowledge of the box's physical properties, such as its mass distribution, geometry, or ground friction characteristics. Despite that, it is expected that the robots should be able to push or lift the box towards its destination through a potentially unknown, dynamic, and cluttered environment, e.g., [6].

In behavior-based robotics, the idea is to let individual, dedicated behaviors be responsible for achieving particular tasks and by combining these behaviors through some arbitration mechanism, more complex missions can be executed, [7]. Following this general approach, with the arbitration mechanism being a winner-takes-all policy, [8], where a single behavior is active at any given time, one can sequentially combine multi-robot task-centric controllers (or *behaviors*); see for example [9, 10] for examples such sequences of multi-robot behaviors. In addition, to operate successfully in dynamic and unknown environments, one can envision these sequences being subject to machine learning in general, and reinforcement learning in particular, e.g., [11, 12], whereby the system interacts with the environment in a structured and

✉ Magnus Egerstedt
magnus@uci.edu

Pietro Pierpaoli
pietro.pierpaoli@gateech.edu

Thinh T. Doan
thinhdoan@vt.edu

Justin Romberg
jrom@ece.gateech.edu

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

[2] Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

[3] Samueli School of Engineering, University of California, Irvine, CA 92697, USA

adaptive manner in order to improve the behavior sequencing over time.

The primary focus of this paper is the problem of selecting suitable sequences of coordinated behaviors together with their parametric instantiations in order to carry out missions which could not be handled by any individual behavior. In addition, when complete information is unavailable, the robots must cooperatively learn which behaviors to use in any given circumstance through their interactions with the environment. The main contribution of the paper is the introduction of a novel reinforcement learning-based method which combines Q-learning with online gradient descent to select both the appropriate behavior sequences and the optimal parameters associated with the constituent behaviors. To illustrate the effectiveness of the proposed approach, it is implemented on a team of differential-drive robots for solving two different, canonical multi-robot tasks, namely, convoy protection and simplified object manipulation.

The paper is organized as follows. A general class of weighted-consensus coordinated behaviors is described in Sect. 2, followed by a discussion of how to formalize multi-robot behavior sequences in Sect. 3. The behavior selection problem is formulated in that section as well, while the proposed method for solving this problem is presented in Sect. 4. The paper is concluded through the application of the method to two canonical multi-robot problems in Sect. 5.

2 Background

In this section, we provide the preliminaries needed to formulate the problem under consideration, including discussions of related work and how to produce distributed controllers by flowing against the gradient to a pairwise performance cost.

2.1 Related work

The key idea behind behavior-based robotics, e.g., [7], is that complex behaviors can be produced through combinations of other behaviors dedicated to solving particular tasks, such as moving to a goal location or avoiding obstacles. A large number of different types of "arbitration mechanisms", or rules for how to combine behaviors, have been proposed—ranging from elaborate voting schemes, [8], through schema-based vector summation, [13], to the idea that more critical behaviors subsumes or dominates less critical ones, [14].

By following the idea of letting a single behavior be active at any one point in time, the result is a sequence of behaviors. Typically, the production of such a sequence is unproblematic for single robot systems, but not quite straightforward for multiple robots where individual behaviors may require certain types of interactions that may or may not be supported by the current multi-robot configuration. Proposed techniques for overcoming such problems through the composition of coordinated controllers include formal methods [15], path planning [9], finite state machines [16], Petri nets [17], and behavior trees [18].

Additionally, once an appropriate sequence of controllers has been chosen by some mechanism, the transitions between individual controllers must be feasible as well in that the information needed to make the transition must be available to the individual agents. Solutions to this problem include the use of motion description languages [19], graph process specifications [20] and control barrier functions [5, 10].

Finally, reinforcement learning offers a paradigm for learning optimal policies in stochastic control problems based on simulation [11, 12]. In this context, a robot seeks to find an optimal policy through interacting with the unknown environment with the goal of optimizing its long-term future reward. Motivated by the broad applications of multi-agent systems, for example, mobile sensor networks [21, 22] and power networks [23], there is a growing interest in studying multi-agent reinforcement learning; see for example [24–26] and the references therein.

The goal in this paper is to see how reinforcement learning can be used to get at the problem of how to select and sequence behaviors for teams of mobile robots. However, before we can start that discussion, some background must be provided about how to actually generate the individual multi-robot behaviors themselves.

2.2 Multi-robot systems

One common approach when designing multi-robot controllers for performing particular tasks is to define local control rules through the use of a performance cost, as discussed in [2]. If this cost structurally respects the information flow in the network, its gradient inherits the same structural properties. As such, a negative gradient flow solves the task at hand (provided that the performance cost has been appropriately selected). Weighted consensus protocols can be generated in this manner. See for example [2] and references therein. One advantage of formulating the multi-agent control problem in terms of such task-specific controllers is that provable performance guarantees can be established in a systematic manner, [27].

To see how this construction works, consider a team of N robots operating in a 2-dimensional domain, where we denote by $x_i \in \mathbb{R}^2$ the state of robot i, for $i = 1, \ldots, N$. In addition, the dynamics of the robots are given by single integrators,

$$\dot{x}_i = u_i, \tag{1}$$

where u_i is the controller of robot i, which may be a function of x_i and the states of the robots interacting with robot i. The pattern of interactions between the robots is presented

by an undirected graph $G = (V, E)$, where $V = \{1, \dots, N\}$ and $E = (V \times V)$ are the index set and the set of pairwise interactions between the robots, respectively. Moreover, let $N_i = \{j \in V \mid (i,j) \in E\}$ be the neighborhood set of robot i.

For the purpose of this paper, and following the general construction from [2], we let the controller u_i for robot i be composed of two components; one that only depends on the robot's own state and one that captures its interactions with neighboring robots. In particular, the controller $u_i : \mathbb{R}^{2+2|N_i|} \mapsto \mathbb{R}^2$ in (1) is given as

$$u_i = -\sum_{j \in N_i} \Big(w(x_i, x_j, \theta)(x_i - x_j) \Big) + v(x_i, \phi), \tag{2}$$

where $w : \mathbb{R}^2 \times \mathbb{R}^2 \times \Theta \to \mathbb{R}$, often referred to as an *edge weight function* [28], depends on the states of robot i and its neighbors, and on the parameter $\theta \in \Theta$. Additionally, $v : \mathbb{R}^2 \times \Phi \to \mathbb{R}^2$ is the state-feedback term for robot i, which depends only on the individual state x_i and a parameter $\phi \in \Phi$, representing what robot i would be doing in the absence of any other robots, which we informally refer to as its "preference". Here, Θ and Φ are the feasible sets of the parameters θ and ϕ, respectively, belonging to some linear vector spaces. A concrete example of such a controller together with the associated parameters will be given in the next section.

As discussed in [2], one can define an appropriate energy function $\mathcal{E} : \mathbb{R}^{2N} \mapsto \mathbb{R}_{\geqslant 0}$ with respect to the graph G, where the controller in (2) can be described as the negative gradient of $\mathcal{E}$, i.e.,

$$u_i = -\frac{\partial \mathcal{E}}{\partial x_i}. \tag{3}$$

This particular observation will prove useful for subsequent developments.

3 Coordinated behaviors

3.1 Behavior sequences

Given a collection of behaviors, the main problem under consideration in this paper is that of optimally selecting the sequence of such behaviors that best complete a given mission.

Definition 1 A coordinated behavior $\mathcal{B}$ is defined by the 5-tuple

$$\mathcal{B} = (w, \Theta, v, \Phi, G), \tag{4}$$

where Θ and Φ are feasible sets for the parameters of the controller in (2). Moreover, G is the graph representing the interaction structure between the robots.

Given M distinct behaviors we compactly represent them as a library of behaviors, $\mathcal{L}$

$$\mathcal{L} = \{\mathcal{B}_1, \dots, \mathcal{B}_M\}, \tag{5}$$

where each behavior $\mathcal{B}_k$ is defined as in (4), i.e.,

$$\mathcal{B}_k = (w_k, \Theta_k, v_k, \Phi_k, G_k), \quad k = 1, \dots, M. \tag{6}$$

Here, note that the feasible sets Θ and Φ, and the graph G may be different for different behaviors, that is, in switching between different behaviors the communication graphs of the robots may be time-varying. Moreover, based on Definition 1, it is important to note the difference between *behavior* and *controller*. The controller (2) executed by the robots for a given behavior is obtained by selecting a proper pair of parameters (θ, ϕ) from the sets Θ and Φ. Indeed, consider a behavior $\mathcal{B}$ and let $x_t = \left[x_{1,t}^{\mathrm{T}}, \dots, x_{N,t}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{2N}$ be the ensemble states of the robots at time t. In addition, let $u_{\mathcal{B}}(x_t, \theta, \phi)$, where $u_{\mathcal{B}} = \left[u_1^{\mathrm{T}}, \dots, u_N^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{2N}$, be the controllers of the robots defined in (2) for a feasible pair of parameters (θ, ϕ). The ensemble dynamic of the robots associated with $\mathcal{B}$ is then given as

$$\dot{x}_t = u_{\mathcal{B}}(x_t, \theta, \phi). \tag{7}$$

To further illustrate the difference between a behavior and its associated controller, we consider the following formation control example.

Example 1 Consider the formation control problem over a network of 4 robots moving in a plane, as illustrated in Fig. 1, where the desired inter-robot distances are given by a vector $\theta = \{\theta_1, \dots, \theta_5\}$, with $\theta_i \in \mathbb{R}_+$. Here, robot 1 acts as a leader and moves toward the goal $\phi \in \mathbb{R}^2$. It should be noted that the desired inter-robot distances also imply something about the interaction structure between robots (graph G) in that sufficient edges must be present in the graph for the formation to be possible, e.g., [28].

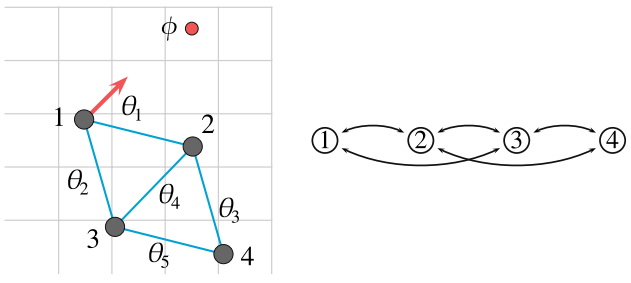

Fig. 1 Example formation for a team of 4 robots and one leader with a goal location, ϕ (left). The interaction graph G needed to support the formation (right)

As the goal of the robots is to maintain the desired formation while moving to the goal location, one possible choice of the edge-weight function of the controller (2) is

$$w = \|x_i - x_j\| - \theta_k, \quad \forall\, e_k = (i,j) \in E, \tag{8}$$

while the individual robot term is given by $v_i = 0,\ i = 2,3,4$, while the leader term is given by

$$v_1 = \phi - x_1. \tag{9}$$

In this example, $\boldsymbol{\Phi}$ is simply a subset of $\mathbb{R}^2$ while $\boldsymbol{\Theta}$ is a set of geometrically feasible distances. Thus, given the formation control behavior $\mathcal{B} = (w, \boldsymbol{\Theta}, v, \boldsymbol{\Phi}, G)$, the controllers $u_{\mathcal{B}}(x, \theta, \phi)$ can be directly derived from (2).

We conclude this section with some additional comments about the formation control problem described in the previous example, where one can choose a single behavior $\mathcal{B} \in \mathcal{L}$ together with a pair of parameters (θ, ϕ) for solving the problem, e.g., [28]. This controller, however, is designed under the assumption that the environment is static and known, i.e., the target ϕ in Example 1 is fixed and known by the robots. Such an assumption is less practical since in many applications, the robots are often operating in dynamically evolving and potentially unknown environments; for example, ϕ is time-varying and unknown. On the other hand, while the formation control problem can be solved using a single behavior, many practical, complex tasks require the robots to support more than one behavior [9, 10]. Our interest, therefore, is to consider the problem of selecting a sequence of the behaviors in $\mathcal{L}$, while allowing for the state of the environment to be unknown and possibly time-varying.

3.2 Optimal behavior selection problems

In this section, we present the problem of optimal behavior selection over a network of robots, through the lens of reinforcement learning. In particular, consider a team of N robots cooperatively operating in an unknown environment and their goal is to complete a given mission over a time interval $[0, t_f]$.

Let x_t and e_t be the states of robots and environment at time $t \in [0, t_f]$, respectively. At any time t, the robots first observe the state of the environment e_t, select a behavior $\mathcal{B}_t$ chosen from the library $\mathcal{L}$, compute the pair of parameters (θ_t, ϕ_t) associated with $\mathcal{B}_t$, and execute the resulting controller $u_{\mathcal{B}}(x_t, \theta_t, \phi_t)$. As a result of the robot actions, as well as the possibly dynamic nature of the environment, its state updates to a new value e_t' over a short sample time, and the robots get a reward returned by the environment based on the selected behavior and tuning parameters.

We here assume that these rewards are appropriately designed in that they encode the given mission, which is motivated by the usual consideration in the literature of reinforcement learning [12]. That is, solving the task is equivalent to maximizing the total accumulated rewards received by the robots. In Sect. 5, we provide a few examples of how to design such reward functions for particular applications. It is worth pointing out that designing these reward functions is itself challenging and requires sufficient knowledge about the underlying problem, as observed in [12].

One could try to solve the optimal behavior selection problem using existing reinforcement learning techniques. However, this problem is in general intractable since the dimension of state space is infinite, i.e., x_t and e_t are continuous variables. Moreover, due to the physical constraint of the robots, it is infeasible (and certainly impractical) for the robots to switch to a new behavior at every discrete time instant. That is, the robots require a finite amount of time to implement the controller of the selected behavior. Thus, to circumvent these issues we next consider an alternate version of the behavior selection problem.

Inspired by the work in [29], we introduce an interrupt condition $\xi : \mathcal{E} \mapsto \{0, 1\}$, where $\mathcal{E}$ is the "energy" in the network, which in turn is a measure of how well the individual task (not the complex mission) is being performed, as was the case in (3) when a negative gradient controller was produced. If $\mathcal{E}_t$ is the value of $\mathcal{E}$ at time t, then the interrupt condition is given by

$$\xi(\mathcal{E}_t) = 1 \ \text{ if } \ \mathcal{E}_t \leqslant \epsilon \tag{10}$$

and $\xi(\mathcal{E}_t) = 0$ otherwise. Here ϵ is a small positive threshold. In other words, $\xi(\mathcal{E}_t)$ represents a binary trigger with value 1 whenever the network energy for a certain behavior at time t is smaller than a threshold. Or, phrased slightly differently, the interrupt condition triggers when the individual task for which the controller was designed has nearly been completed. Thus, it is reasonable to insist that the robots should not switch to a new behavior at time t unless $\xi(\mathcal{E}_t) = 1$ for a given ϵ.

Based on this observation, given a desired threshold ϵ, let τ_k be the switching time associated with behavior $\mathcal{B}$, defined as

$$\tau_k(\mathcal{B}, \epsilon, t_0) = \min\{t \geqslant t_0 \,|\, \mathcal{E}_t \leqslant \epsilon\}. \tag{11}$$

Consequently, the mission time interval $[0, t_f]$ is partitioned into K switching times $\tau_0, \ldots, \tau_K$ satisfying

$$0 = \tau_0 \leqslant \tau_1 \leqslant \cdots \leqslant \tau_K = t_f, \tag{12}$$

where each switching time, except τ_0 and τ_K, is defined as in (11). Note that the number of switching times, K, depends

on the accuracy ϵ. In this paper, we do not consider the problem of how to select the appropriate threshold, ϵ.

We are now ready to describe, at a high level, how the behavior selection mechanism should operate. At each switching time τ_i, the robots choose a behavior $\mathcal{B}_i \in \mathcal{L}$ based on their current states, x_{τ_i}, and the environment state, e_{τ_i}. Next, they decide on a pair of parameters (θ_i, ϕ_i) and implement the underlying controller $u_{\mathcal{B}_i}(x_t, \theta_i, \phi_i)$ for $t \in [\tau_i, \tau_{i+1})$. Based on the selected behaviors and parameters, the robots receive an instantaneous reward $\mathcal{R}(x_{\tau_i}, e_{\tau_i}, \mathcal{B}_i, \theta_i, \phi_i)$ returned by the environment as a function of the selection.

Let J be the accumulative reward received by the robots at the switching times in $[0, t_f]$,

$$J = \sum_{i=0}^{K} \mathcal{R}(x_{\tau_i}, e_{\tau_i}, \mathcal{B}_i, \theta_{\tau_i}, \phi_{\tau_i}). \tag{13}$$

As mentioned previously, the optimal behavior selection problem is cast as the problem of finding a sequence of behaviors $\{\mathcal{B}_i\}$ from $\mathcal{L}$ at $\{\tau_i\}$, and the associated parameters $\{(\theta_i, \phi_i)\} \in \{\Theta_i \times \Phi_i\}$ so that the accumulative reward J is maximized. This optimization problem can be formulated as follows:

$$\begin{aligned} \underset{\mathcal{B}_i, \theta_i, \phi_i}{\text{maximize}} \quad & \sum_{i=0}^{K} \mathcal{R}(x_{\tau_i}, e_{\tau_i}, \mathcal{B}_i, \theta_i, \phi_i) \\ \text{such that} \quad & \mathcal{B}_i \in \mathcal{L}_b,\ (\theta_i, \phi_i) \in \Theta_i \times \Phi_i, \\ & e_{t+1} = f_e(x_t, e_t), \\ & \dot{x} = u_{\mathcal{B}_i}(x_t, \theta_i, \phi_i),\ t \in [\tau_i, \tau_{i+1}), \end{aligned} \tag{14}$$

where $f_e : \mathbb{R}^{2N} \times \mathbb{R}^2 \mapsto \mathbb{R}^2$ is the unknown dynamic of the environment. Since f_e is unknown, one cannot use dynamic programming to solve this problem. Thus, in the next section we propose a novel method for solving (14), which is a combination of Q-learning and online gradient descent. Moreover, by introducing the switching times τ_i, computing the optimal sequence of behaviors using Q-learning is now a tractable problem.

4 A Q-learning approach to behavior selection

In this section, we propose a novel reinforcement learning based method for solving problem (14). The method is composed of Q-learning and online gradient descent methods to find an optimal sequence of behaviors $\{\mathcal{B}_i^*\}$ and the associated parameters $\{(\theta_i^*, \phi_i^*)\}$, respectively. In particular, we maintain a Q-table, whose (i, j) entry is the state-behavior value estimating the performance of behavior $\mathcal{B}_j \in \mathcal{L}$ given the environment state $j \in \mathcal{S}$, where we, for notational simplicity, have assumed that states of the environment at the switching times belong to a finite set $\mathcal{S}$, i.e., $e_{\tau_i} \in \mathcal{S}$ for all i. Thus, one can view the Q-table as a matrix $Q \in \mathbb{R}^{S \times M}$, where S is the size of $\mathcal{S}$ and M is the number of behaviors in $\mathcal{L}$. The entries of the Q-table are updated using Q-learning while the controller parameters are updated using z continuous-time online gradient method. These updates are formally presented in the following algorithm.

Algorithm 1 Q-learning algorithm for optimal behavior selection and tuning. The notation $\sim \mathcal{U}(\mathcal{O})$ is used to represent variables uniformly selected from a set $\mathcal{O}$

$x_0 \sim \mathcal{U}(\mathbb{R}^{2N})$;
$\mathcal{L}_b = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$;
$Q(s, m) \sim \mathcal{U}(\mathbb{R}),\ \forall m = 1, \ldots, M,\ s \in \mathcal{S}$;
for $m \in 1, \ldots, M$ **do**
  $\theta_m \sim \mathcal{U}(\Theta_m), \phi_m \sim \mathcal{U}(\Phi_m)$;
end
$s \in \mathcal{S} \leftarrow$ Observe e_{τ_i};
for $i = 0$ **to** K **do**
  Select $m = \underset{\ell=1,\ldots,M}{\arg\max}\ Q(s, \ell)$;
  $(\bar{\theta}_t, \bar{\phi}_t) \leftarrow (\theta_m, \phi_m)$;
  while $\mathcal{E} \geqslant \varepsilon$ **do**
    $\dot{x} = f_{\mathcal{B}_m}(x_t, \bar{\theta}_t, \bar{\phi}_t)$;
    $\dot{\bar{\theta}} = -\nabla_\theta C_t(x_t, e_{\tau_i}, \bar{\theta}_t, \bar{\phi}_t)$;
    $\dot{\bar{\phi}} = -\nabla_\phi C_t(x_t, e_{\tau_i}, \bar{\theta}_t, \bar{\phi}_t)$;
  end
  $(\theta_m, \phi_m) \leftarrow (\bar{\theta}_t, \bar{\phi}_t)$;
  $r_i \leftarrow$ from environment;
  $s' \in \mathcal{S} \leftarrow$ Observe $e_{\tau_{i+1}}$;
  $Q(s, m) = Q(s, m) + \epsilon\big(r_i + \max_j Q(s', j) - Q(s, m)\big)$;
  $s \leftarrow s'$
end

In the proposed algorithm, at each switching time τ_i, the robots first observe the environment state $e_{\tau_i} = s \in \mathcal{S}$, and then select a behavior $\mathcal{B}_m$ with respect to the maximum entry in the sth row of the Q-table with ties being broken arbitrarily. Next, the robots implement the distributed controller $f_{\mathcal{B}_m}$ and use online gradient descent to find the best parameters (θ_m, ϕ_m) associated with $\mathcal{B}_m$. In order for such a construction to be meaningful, a cost must be established against which the gradient can be taken. To that end, we use the function C_t which can be thought of as a representation of the cost of implementing the controller at time t. This cost can either be chosen in advance (e.g., by equating it to $\mathcal{E}$ in (3)), or let it be returned by the environment.

Based on the selected behavior and the associated controller, the robots receive an instantaneous reward, r_i, while the environment transitions to a new state $s' \in \mathcal{S}$.

Finally, the robots update the (s, m) entry of the Q-table using the update law associated with the Q-learning method, [11, 12]. It is worth noting that the Q-learning step is done in a centralized manner (either by the robots or a supervisory coordinator) since it depends on the state of the environment. Similarly, depending on the structure of the cost functions C_t, the online gradient descent updates can be implemented either in a distributed or in a centralized manner.

5 Example applications

In this section we describe two applications of the proposed behavior selection technique. In both examples, we consider a team of 5 robots and a library of 5 behaviors given by

1. Static formation:

$$u_i = \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - (\theta\, \delta_{ij})^2)(x_j - x_i),$$

where δ_{ij} is the desired separation between robots i and j, while $\theta \in \mathbb{R}$ is a shape scaling factor.

2. Formation with leader:

$$u_i = \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - \theta_{ij}^2)(x_j - x_i),$$
$$u_\ell = \sum_{j \in \mathcal{N}_\ell} ((\|x_\ell - x_j\|^2 - \theta_{\ell j}^2)(x_j - x_\ell)) + (\phi - x_\ell),$$

where δ_{ij} and θ are defined as in the previous controller, while $\phi \in \boldsymbol{\Phi} \subseteq \mathbb{R}^2$ is the leader's goal. The subscript ℓ denotes the leader agent's index.

3. Cyclic pursuit:

$$u_i = \sum_{j \in \mathcal{N}_i} R(\theta)\,(x_j - x_i) + (\phi - x_i),$$

where $\theta = 2r \sin \frac{\pi}{N}$, r is the radius of the cycle formed by the robots, and $R(\theta) \in \mathrm{SO}(2)$. The point $\phi \in \boldsymbol{\Phi} \subseteq \mathbb{R}^2$ is the center of the cycle.

4. Leader-follower:

$$u_i = \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - \theta^2)(x_j - x_i),$$
$$u_\ell = \sum_{j \in \mathcal{N}_\ell} ((\|x_\ell - x_j\|^2 - \theta^2)(x_j - x_\ell)) + (\phi - x_\ell),$$

where θ is the separation between the agents and $\phi \in \boldsymbol{\Phi} \subseteq \mathbb{R}^2$ is the leader's goal.

5) Triangulation coverage:

$$u_i = \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - \theta^2)(x_j - x_i), \tag{15}$$

where θ is the separation between the agents in the triangulation.

For all the behaviors considered above, we assume the following parameter spaces $\boldsymbol{\Theta} = [0.05, 1.1]$ and $\boldsymbol{\Phi} = [-1, 1]$.

In both examples, we construct the state-action value function by implementing the proposed method using Algorithm 1, on the Robotarium simulator, as described in [30]. What these simulations show is that the proposed Q-learning method not only outperforms random parameter and behavior sequence selections, which is expected, but also closely matches the performance of ad-hoc algorithms designed explicitly for the purpose of solving the particular problem. As such, simulations show that online learning approaches, such as reinforcement learning, provide a promising avenue for optimal behavior sequencing based on real-time collected data in multi-robot systems to achieve complex tasks in unknown and time-varying environments.

5.1 Convoy protection

First, we consider a *convoy protection* problem, where a team of robots must surround a moving target and maintain a single robot-to-target distance equal to a constant Δ at all times. Although this problem can be solved by executing a single behavior (e.g., cyclic-pursuit), it allows us to compare the performance of the proposed framework against an ideal solution. The position of the target is denoted by z_t and it is given by the following dynamics:

$$\dot{z}_t = v_z + \sigma, \tag{16}$$

where v_z is a constant velocity and σ is a zero-mean Gaussian disturbance. In this case, the state of the environment at time step t is considered to be the separation between the robots' centroid $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and the target, i.e.,

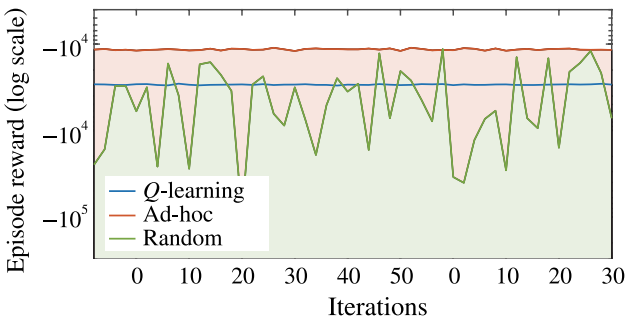

Fig. 2 Comparison between accrued reward for 50 different episodes of the convoy protection example. Rewards from trained model are compared against a tuned and targeted solution and random behaviors/parameters selection. As expected, the trained model outperforms the random solution but not the ad-hoc solution explicitly designed to solve the convoy protection problem

$$e_t = \|\bar{x}_t - z_t\|, \tag{17}$$

where $\|\cdot\|$ denotes the Euclidean norm. The reward provided by the environment at time t is

$$r_t = -\|e_t - \bar{x}_t\|^2 - \frac{1}{N}\sum_{1}^{N}(\|x_{i,t} - e_t\| - \Delta)^2, \tag{18}$$

where the first term represents the proximity between centroid and target, while the second term weights the individual robot-to-target distance. Training is executed over 1000 episodes, with an exponentially decaying greedy policy.

The plot in Fig. 2 shows the collected rewards over a trial of 50 episodes. The results from the trained model are compared against an ad-hoc ideal solution, where the CYCLIC-PURSUIT behavior is recursively executed with parameters θ and ϕ being selected so that the resulting cycle has radius Δ and is centered on the target's position. Finally, the figure also shows the rewards collected when the behaviors and parameters are selected uniformly at random.

5.2 Object manipulation

In the second example in Fig. 3, we consider a team of robots tasked with moving an object from two points. Let e_t represent the position of the object at time t. In order not to complicate the focus of the experiment, we assume somewhat simplified manipulation dynamics. In particular, the box maintains its position if the closest robot is further than a certain threshold (i.e., object is not detected by the robots), otherwise it moves as $\bar{x}$. This manipulation dynamics guarantee that the task cannot be solved with a single, fixed behavior and parameters since it is unknown to the robots. In this context, the robots get the following reward:

$$r_t = -(\kappa + \|e_t - \bar{e}\|), \tag{19}$$

where κ is a constant used to weight the running time until completion of an episode and $\|e_t - \bar{e}\|$ is the distance between the box and its final destination $\bar{e}$.

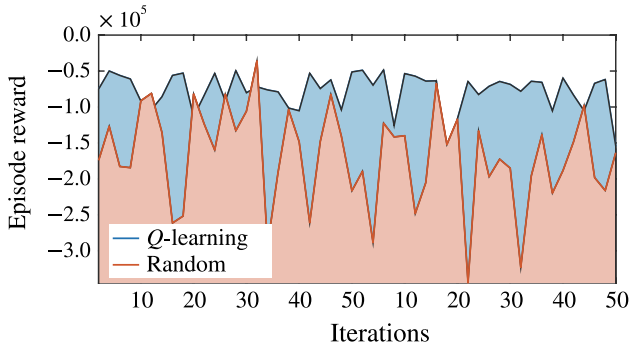

Fig. 4 Comparison between accrued reward over 50 episodes of the object manipulation example. Rewards from trained model are compared with random behaviors/parameters selection

In Fig. 3, the robots are executing the learned policy on the object manipulation task. As can be seen, the robots first execute a leader-follower behavior in order to explore the environment and bring the target within detection distance. After the object is collected, the robots arrange themselves into a formation and collectively transport the object to the desired destination (circle) by executing a cyclic pursuit. Finally, in Fig. 4, we display a comparison between the cumulative reward accrued by the robots over 50 executions of the object collection task when following two distinct policies. In the first case, the robots are following the learned strategy while, in the second case, the behaviors and corresponding parameters are selected from a uniform distribution at interval of times selected from a Poisson distribution. As we can be seen, the learned policy is around two times as effective as this policy.

6 Conclusions

In this paper, we present a reinforcement learning based approach for solving the optimal behavior selection problem, where the robots interact with an unknown environment. Given a finite library of behaviors, the proposed technique

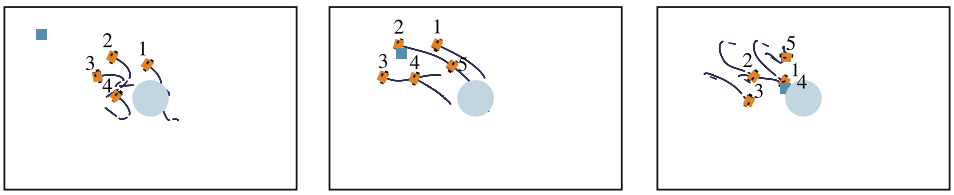

Fig. 3 Screen shoot from Robotarium simulator during execution of the simplified object manipulation scenario at three different times. Executed behaviors are LEADER-FOLLOWER (left), FORMATION WITH LEADER (center), and CYCLIC-PURSUIT (right). The square represents the object which is collectively transported towards the goal (circle)

exploits rewards collected through interaction with the environment to solve a given task that could not be solved by any single behavior. We furthermore provide numerical experiments on a network of robots to illustrate the effectiveness of the proposed method.

Acknowledgements This work was supported by the Army Research Lab (No. DCIST CRA W911NF-17-2-0181).

References

1. Antonelli, G. (2013). Interconnected dynamic systems: An overview on distributed control. *IEEE Control Systems Magazine, 33*(1), 76–88.
2. Cortés, J., & Egerstedt, M. (2017). Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration, 10*(6), 495–503.
3. Oh, K. K., Park, M. C., & Ahn, H. S. (2015). A survey of multi-agent formation control. *Automatica, 53,* 424–440.
4. Schwager, M., Rus, D., & Slotine, J. J. (2011). Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *International Journal of Robotics Research, 30*(3), 371–383.
5. Li, A., Wang, L., Pierpaoli, P., & Egerstedt, M. (2018). Formally correct composition of coordinated behaviors using control barrier certificates. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3723–3729. Madrid, Spain.
6. Berman, S., Lindsey, Q., Sakar, M. S., Kumar, V., & Pratt, S. C. (2011). Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proceedings of the IEEE, 99*(9), 1470–1481.
7. Arkin, R. C. (1998). *Behavior-based robotics*. Cambridge: MIT Press.
8. Rosenblatt, J. K. (1997). Damn: A distributed architecture for mobile navigation. *Journal of Experimental & Theoretical Artificial Intelligence, 9*(2–3), 339–360.
9. Nagavalli, S., Chakraborty, N., & Sycara, K. (2017). Automated sequencing of swarm behaviors for supervisory control of robotic swarms. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2674–2681. Singapore.
10. Pierpaoli, P., Li, A., Srinivasan, M., Cai, X., Coogan, S., & Egerstedt, M. (2019). A sequential composition framework for coordinating multi-robot behaviors. *arXiv preprint*. arXiv:1907.07718.
11. Bertsekas, D. P. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific.
12. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge: MIT Press.
13. Arbib, M. A. (1992). Schema theory. *The Encyclopedia of Artificial Intelligence, 2,* 1427–1443.
14. Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence, 47*(1–3), 139–159.
15. Kress-Gazit, H., Lahijanian, M., & Raman, V. (2018). Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems, 1,* 211–236.
16. Marino, A., Parker, L., Antonelli, G., & Caccavale, F. (2009). Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In *IEEE International Conference on Robotics and Automation*, pp. 831–836. Kobe, Japan.
17. Klavins, E., & Koditschek, D. E. (2000). A formalism for the composition of concurrent robot behaviors. In *IEEE International Conference on Robotics and Automation*, pp. 3395–3402. San Francisco, CA, USA.
18. Colledanchise, M., & Ögren, P. (2014). How behavior trees modularize robustness and safety in hybrid systems. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1482–1488. Chicago, IL, USA.
19. Martin, P., & Egerstedt, M. B. (2012). Hybrid systems tools for compiling controllers for cyber-physical systems. *Discrete Event Dynamic Systems, 22*(1), 101–119.
20. Twu, P., Martin, P., & Egerstedt, M. (2010). Graph process specifications for hybrid networked systems. *IFAC Proceedings Volumes, 43*(12), 65–70.
21. Cortes, J., Martinez, S., Karatas, T., & Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation, 20*(2), 243–255.
22. Ogren, P., Fiorelli, E., & Leonard, N. E. (2004). Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control, 49*(8), 1292–1302.
23. Kar, S., Moura, J. M. F., & Poor, H. V. (2013). *QD*-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations. *IEEE Transactions on Signal Processing, 61,* 1848–1862.
24. Doan, T. T., Maguluri, S. T., & Romberg, J. (2019). Finite-time analysis of distributed TD(0) with linear function approximation for multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1626–1635. Long Beach, CA, USA.
25. Wai, H. T., Yang, Z., Wang, Z., & Hong, M. (2018). Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Annual Conference on Neural Information Processing Systems*. Montreal, Canada.
26. Zhang, K., Yang, Z., & Basar, T. (2018). Networked multi-agent reinforcement learning in continuous spaces. In *IEEE Conference on Decision and Control (CDC)*, pp. 2771–2776. Miami Beach, FL, USA.
27. Zelazo, D., Mesbahi, M., & Belabbas, M. A. (2018). Graph theory in systems and controls. In *IEEE Conference on Decision and Control (CDC)*, pp. 6168–6179. Miami Beach, FL, USA.
28. Mesbahi, M., & Egerstedt, M. (2010). *Graph theoretic methods in multiagent networks*. vol. 33. Princeton University Press.
29. Mehta, T. R., & Egerstedt, M. (2006). An optimal control approach to mode generation in hybrid systems. *Nonlinear Analysis: Theory, Methods & Applications, 65*(5), 963–983.
30. Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., & Egerstedt, M. (2017). The robotarium: A remotely accessible swarm robotics research testbed. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1699–1706. Singapore.

Pietro Pierpaoli is a Robotics Engineer at Quiet Machines LLC in Pasadena, CA, USA. He was a postdoctoral fellow at the Georgia Institute of Technology, GA, USA, in the School of Electrical and Computer Engineer. He received the B.Sc. and M.Sc. degrees in Aerospace and Aeronautical Engineering respectively from Politecnico di Milano, Italy, and Ph.D. degree in Mechanical Engineering from the University of Miami, FL, USA. Pietro was a visiting scholar in the School of Electrical and Computer Engineering at the Georgia Institute of Technology in 2015, and in the William E. Boing department of Aeronautics and Astronautics at the University of Washington in 2016.

Thinh T. Doan is an Assistant Professor in the Department of Electrical and Computer Engineering at Virginia Tech. He obtained his Ph.D. degree at the University of Illinois, Urbana-Champaign, his master degree at the University of Oklahoma, and his Bachelor's degree at Hanoi University of Science and Technology, Vietnam, all in Electrical Engineering His research interests span on the intersection of control theory, optimization, machine learning, reinforcement learning, and applied probability theory.

Justin Romberg is the Schlumberger Professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology, where he has been on the faculty since 2006. Dr. Romberg received the B.Sc., M.Sc., and Ph.D. degrees in Electrical and Computer Engineering from Rice University, and was a Postdoctoral Scholar in Applied and Computational Mathematics at the California Institute of Technology. His current research interesting lie at the intersection of statistical signal processing, machine learning, optimization, and applied probability.

Magnus Egerstedt is the Dean of Engineering in the Samueli School of Engineering at the University of California, Irvine, where he is also a Professor in the Department of Electrical Engineering and Computer Science. Egerstedt received the M.Sc. degree in Engineering Physics and the Ph.D. degree in Applied Mathematics from the Royal Institute of Technology, Stockholm, Sweden, the B.A. degree in Philosophy from Stockholm University, and spent a large part of his career as a faculty member at the Georgia Institute of Technology. Dr. Egerstedt conducts research in the areas of control theory and robotics, with particular focus on control and coordination of complex networks, such as multi-robot and cyber-physical systems. Egerstedt is a Fellow of IEEE and IFAC, and is a Foreign member of the Royal Swedish Academy of Engineering Science. His teaching and research awards include the Ragazzini Award, the O. Hugo Schuck Best Paper Award, and the Alumni of the Year Award from the Royal Institute of Technology.