Model-Free Policy Gradients for Multi-Agent Shape Formation

Elizabeth Boroson, Fei Sha, and Nora Ayanian

Abstract— **Multi-agent tasks that require tight coordination and adaptation to changing environments, such as in first response scenarios, are challenging to address with distributed solutions, due to the coordination and information exchange required between agents. In this work, we discuss the problem of shape formation, where groups of anonymous agents must arrange themselves into a prescribed shape without communication and with only a limited field of view. To solve this problem, we develop a model-free technique based on gradient ascent for multi-agent systems to learn distributed policies. While shape formation provides a working example, the technique is general and can be applied to any problem requiring cooperation between agents and scaled to groups of any size. We present results from simulations using this method on groups of agents of several sizes. Compared to other model-free methods, our technique is more consistent and adapts better to changes in the environment, and compared to commonly-used model-based methods, it performs significantly better and can handle larger groups.**

I. INTRODUCTION

Distributed solutions for tasks that require tight coordination between multiple robots present a significant challenge. First, agents must interact with other agents in order to maintain enough knowledge about them and their state. Second, since agents are likely to have a limited view of their environment, they must model their own sensors and any uncertainty about their state. Finally, while communication between agents may be helpful, managing the volume of data that must be exchanged with large numbers of agents becomes challenging.

For example, consider a group of small robots performing search and rescue after a natural disaster, such as in a collapsed building after an earthquake. Initially, the robots may need to explore the building as any previous floorplan is unlikely to be accurate. They may separate and later rendezvous to exchange collected information, but their ability to coordinate would be extremely limited. No robot would have a complete map of the building, and even with some ability to communicate, wireless signals may be unreliable. Finally, visibility may be limited by the sensors available, dust, debris, or rubble, so the robots may not see each other from a distance. The robots will be aware when they have all found each other, but may not receive any intermediate feedback before reaching that goal. This scenario is representative of the type of task and extremely strict conditions that may be faced by a group of robots that need to coordinate.

Authors are with the Department of Computer Science, University of Southern California, USA {boroson, feisha, ayanian}@usc.edu

In this work, we explore coordination under such severely restricted conditions. Specifically, we study the task of shape formation, where a group of agents in a 2-D space must arrange themselves into a desired shape. For example, they may need to provide a redundant wireless network or collaboratively move large items (such as rubble). Agents can move within the space with limited velocity and observe other agents nearby, but cannot uniquely identify or explicitly communicate with them. The shape may be located anywhere in the space, as only agents' relative positions are important for task completion. This problem relates to the search-and-rescue scenario described and to many other tasks requiring a group of robots to assemble in a formation or maintain tight positioning constraints, such as network deployment [1], surveillance [2], [3], and entertainment [4]. In those tasks, while communication may be available, the bandwidth required makes communication across the entire team infeasible.

Shape formation is a task that can only be accomplished with tight coupling between agents. With more agents, accuracy may improve but coordination becomes more difficult. Previous work on shape formation has used potential functions to move agents to a previously-defined shape [3], [5], [6], and assigned each agent to a specific point in the goal shape [7]. In these works, though control is decentralized, agents must be aware of their position relative to the goal shape and only coordinate with other agents by maintaining a fixed distance. It may be unrealistic for agents to know their exact position with limited sensing, and predefining a goal shape requires centralized decision-making that may not be available in a distributed system. Also, if the environment changes, as in the collapsed building, the predefined location may be unreachable.

With a centralized planner and full observability of the space, shape formation is a deterministic Markov decision process (MDP). However, when each agent has a limited view of its environment and no communication with other agents, the task becomes a decentralized partially observable MDP (Dec-POMDP) [8]. The task requires tight coordination among all agents: no agent can solve the problem alone, and the group of agents cannot succeed if one agent does not participate. Due to the interaction required, the problem difficulty scales with the number of agents. In a group of two, each agent only needs to interact with at most one other agent, but in a larger group, agents must be able to interact with many others concurrently.

This paper's contribution is the development of a model-free approach for a multi-agent system to learn distributed policies that perform well in both the training environment

and new environments. Using a policy parameterization that allows for gradient methods, a group of agents uses gradient ascent to jointly reach a set of policies with which they complete the shape formation task efficiently in the environment they trained in and in unknown environments. While the shape formation problem provides a working example, the technique is much more general and can be applied to any problem requiring cooperation between agents and scaled to any size group.

Other model-free machine learning approaches for multi-agent coordination do not apply to shape formation or cannot overcome changes to the environment after training. Model-based Dec-POMDP solvers cannot handle models of the size required for more than a few agents, because each agent must also model the behavior of the other agents. With those methods, scaling up to hundreds of agents would be impossible. Our method, on the other hand, scales linearly in the number of agents. With parallelization of training, it could be applied to any number of agents.

In this paper, we discuss how the shape formation problem relates to other cooperative tasks that have been studied (Section II). We formally define the problem (Section III) and discuss our method of model-free learning (Section IV). Finally, we show results from simulations using this method on groups of agents with different sizes and goal shapes and compare results from our method to those from commonly-used Q-learning and model-based methods (Section V).

II. RELATED WORK

A. Model-Based Approaches

The shape formation problem is an example of a Dec-POMDP [9]. Most previous work on Dec-POMDPs has focused on model-based approaches using search trees to find optimal policies or near-optimal approximations. These algorithms require explicit models of the state space, transitions, and observations in the problem, and often fail if the state space or number of agents is too large. Some algorithms can take advantage of structured problems like factored Dec-POMDPs, but shape formation does not have this structure because agents must interact.

Recent work has extended solutions to larger Dec-POMDPs using macro-actions, where individual actions are replaced by controllers for complex actions [10], [11]. This allows agents to solve less complex problems. However, it also requires them to maintain detailed models of the environment and controllers for those actions, which may be limiting for small agents with limited capabilities.

B. Model-Free Approaches

Using model-free approaches, some work has been done on distributed reinforcement learning in multi-agent systems, such as Distributed Q-Learning (DQL) for a fully-observable cooperative MDP [12]. Several algorithms based on Q-Learning exist for non-cooperative or fully competitive games (cf. Busoniu *et al.* [13] for related references). However, these algorithms are not typically evaluated on larger groups or large state spaces due to computational requirements. Additionally, they consider tasks where a single agent can learn alone but cooperation is helpful, such as foraging [14], [15]. Many solutions require agents' actions to be abstracted to behaviors (e.g., *look for food*, *take food to base* in foraging). This allows agents to perform more complex tasks with reduced algorithmic complexity, but restricts them to those predetermined behaviors even if more efficient behaviors exist.

In Hysteretic Q-Learning (HQL) [16], [17], agents learn a Q-value function that is not overly optimistic, as DQL is, but also accounts for other agents learning better policies. Our experiments with HQL demonstrate that though agents using it learn good policies for shape formation, the policies they learn have large variance in quality and have difficulty overcoming changes in the workspace, which is critical in realistic multi-robot applications, since environmental conditions may not always be known or controllable. Recent work on learning the hysteretic Q-value function with deep neural networks and distillation has shown success in multi-task learning, but still requires training and execution in the same environments [18].

Policy gradient algorithms, in which an agent follows a parameterized policy and uses gradient ascent to reach parameters that maximize the reward, are sometimes used for POMDPs [19], [20]. They are common in single-agent learning, but due to computational requirements, have rarely been applied to distributed multi-agent systems. One example is the box-pushing work of Buffet *et al.* [21], where two agents with limited observations learn to push boxes collaboratively. Using policy gradients and reward shaping, the agents start from a simple task and move to more complex environments. However, the problem is inherently limited in the number of agents that interact: even with more present, it requires coordination between only two agents.

Other recent work has trained distributed agents using methods that allow centralized training, but distributed execution, such as in Foerster *et al.*'s work on learning communication [22]. Another approach is actor-critic methods, where a critic that can observe the whole state estimates gradients or Q-functions to improve the agents' policies [23], [24]. These methods expect that the environment will not change between training and execution, which does not apply for robots operating in previously unknown environments. Our algorithm does not make any assumptions about the environment, so does not depend on it remaining the same.

III. PROBLEM FORMULATION

A. Preliminary Definitions

A Dec-POMDP is defined by a tuple $\langle A, X, \{U_i\}, P, R, \{Y_i\}, O, h\rangle$. Here, A is a finite set of n agents ($|A| = n$). X is a finite set of states. At every time step, the system is in state $x \in X$. U_i is a finite set of actions agent $a_i \in A$ can take, with the set of joint actions U. At every time step, a_i takes action $u_i \in U_i$, and the joint action of all agents is $u \in U$. $P : X \times U \times X \to [0, 1]$ gives the transition probabilities for each initial state, action, and final state, with $P(x, u, x') = \Pr(x'|x, u)$. $R : X \times U \to \mathbb{R}$

is the reward function, where $R(x, u)$ is the reward for being in state x and taking action u; all agents receive the same reward. Y_i is the set of observations a_i can make, with the set of joint observations Y. At each time step, a_i makes observation $y_i \in Y_i$, and the joint observation is $y \in Y$. $O : Y \times U \times X \to [0, 1]$ is the observation probability, where $O(y, u, x') = \Pr(y|u, x')$. Finally, $h \in \mathbb{N} \cup \{\infty\}$ is the horizon (i.e., number of steps until termination).

The shape formation problem is a Dec-POMDP with deterministic state transitions and observations, and observations and rewards depending only on the state (and not the action to reach that state). The underlying multi-agent MDP $\langle A, X, \{U_i\}, P, R, h\rangle$ is deterministic: given an initial state and joint action, the final state is known. Similarly, given a state, the joint observation is known. However, the reverse is not true: the state is not known from the observation.

When solving a Dec-POMDP, the goal is to find a policy $\pi : Y \to U$ identifying the joint action to be taken given a joint observation. This is the action for each agent that will result in the highest expected reward for the group. The underlying MDP state transitions depend on the joint actions. However, each agent's action depends only on its own observation, so action selection is distributed.

Dec-POMDPs have a fixed or infinite horizon, h. The optimal policy is the policy with which the agents accumulate the maximum expected reward in h steps. In shape formation, the agents must reach a goal state in the fewest steps. We model this with a horizon that is finite but larger than the number of steps required. The agents receive a reward for each time step they are in a goal state, so they maximize their reward by reaching a goal state as quickly as possible.

B. Problem Definition

Consider a group of n agents in a closed, bounded, connected workspace $\mathcal{W} \subset \mathbb{R}^2$. At time step t, agent a_i has position $\mathbf{x}_i^t = (x_i, y_i)^t$ and takes action $\mathbf{u}_i^t = (u_i, v_i)^t$, where $\|\mathbf{u}_i^t\|_2 < v_{max}$, and

$$\begin{aligned} \mathbf{x}_i^{t+1} &= \mathbf{x}_i^t + \mathbf{u}_i^t, i = 1, \ldots, n \\ \mathbf{x}^t &\equiv [\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_n^t] \in X \\ \mathbf{u}^t &\equiv [\mathbf{u}_1^t, \mathbf{u}_2^t, \ldots, \mathbf{u}_n^t] \in U. \end{aligned} \tag{1}$$

Here, $X \equiv \mathcal{W}^n \subset \mathbb{R}^{2n}$ and $U \subset \mathbb{R}^{2n}$. Agent a_i observes agents within radius r,

$$\mathbf{y}_i^t = \left\{\mathbf{x}_j^t - \mathbf{x}_i^t \;\middle|\; \|\mathbf{x}_j^t - \mathbf{x}_i^t\|_\rho < r,\ j \in \{1, \ldots, n\}\right\}, \tag{2}$$

with $\rho = 2$. The joint observation is $\mathbf{y}^t \in Y$. The agents have a set of *goal states* $X_G \subset X$ that is closed under translation and permutation. Specifically, $\forall\, \mathbf{x} \in X_G$, consider the translation of $\mathbf{x}$:

$$\mathbf{x}' \equiv [\mathbf{x}_1 + (x, y), \mathbf{x}_2 + (x, y), \ldots, \mathbf{x}_n + (x, y)]. \tag{3}$$

If $\mathbf{x}' \in X$, then $\mathbf{x}' \in X_G$. The permutation of $\mathbf{x}$ is

$$\mathbf{x}'' \equiv [\mathbf{x}_{\sigma(1)}, \mathbf{x}_{\sigma(2)}, \ldots, \mathbf{x}_{\sigma(n)}] \in X_G, \tag{4}$$

where σ is any permutation of $\{1, 2, \ldots, n\}$. In other words, the goal shape can appear anywhere in $\mathcal{W}$ and the agents can appear in any order within the goal shape. For example, if the goal shape is a square, any agent can be in a corner.

Problem 3.1 (Continuous Shape Formation): For any initial state $\mathbf{x}^0 \in X$, consider the system (1) with workspace $\mathcal{W} \subset \mathbb{R}^2$, goal set $X_G \subset X \subset \mathbb{R}^{2n}$ and observations (2), with $\rho = 2$. Find a joint policy $\pi : Y \to U$ such that:

1) $\forall\, \mathbf{x}^0$, $\exists\, t_G$ with $\mathbf{x}^t \in X_G$ for $t \geq t_G$;
2) t_G is minimized; and
3) $\mathbf{u}_i^t$ depends only on $\mathbf{y}_i^t$.

The continuous shape formation problem has infinite state and action spaces, making the problem intractable. Furthermore, defining a goal in continuous space can be challenging, since it may be unclear which states should be in the goal set. For example, a square with aspect ratio of 1.1 may or may not be close enough to the desired square. Thus, we reduce the problem to discrete space, where the state and action spaces are finite and goal sets can be more easily defined.

Consider a group of n agents in a finite, connected workspace $\bar{\mathcal{W}} \subset \mathbb{N}^2$. Henceforth we will refer only to the discrete problem, and thus allow agent a_i to have position $\mathbf{x}_i^t \in \mathbb{N}^2$ and actions $\mathbf{u}_i^t \in \mathbb{N}^2$ such that $\|\mathbf{u}_i^t\|_1 \in \{0, 1\}$. The system inherits dynamics (1) and observations (2) from the continuous problem, with X and U finite and discrete, $\rho = \infty$, and joint observation $\mathbf{y}^t \in Y$. The system has goal states $X_G \subset X$, with properties (3) and (4).

Problem 3.2 (Discrete Shape Formation): For any initial state $\mathbf{x}^0 \in X$, consider the discrete system (1) in workspace $\bar{\mathcal{W}} \subset \mathbb{N}^2$, goal set $X_G \subset X \subset \mathbb{N}^{2n}$ and discrete observations (2) with $\rho = \infty$. Find a joint policy $\pi : Y \to U$ such that:

1) $\forall\, \mathbf{x}^0$, $\exists\, t_G$ with $\mathbf{x}^t \in X_G$ for $t \geq t_G$;
2) t_G is minimized; and
3) $\mathbf{u}_i^t$ depends only on $\mathbf{y}_i^t$.

Due to the applications we are interested in (e.g., search and rescue), we additionally desire a policy that is robust to changes in $\mathcal{W}$ or $\bar{\mathcal{W}}$. In other words, in a different finite, connected workspace $\bar{\mathcal{W}}' \subset \mathbb{N}^2$ of the same size as $\bar{\mathcal{W}}$, agents following policy π should not require more steps to reach a goal state than agents following a random policy.

IV. METHODS

A. Discrete Simulation

We simulate the shape formation task using a rectangular grid world where agents move in discrete time steps. Agents occupy one grid cell and can move horizontally or vertically to an adjacent empty cell. Each agent observes the surrounding 8 grid cells, and has no other information about the grid world. Agents cannot communicate or uniquely identify other agents that they observe. An example is shown in Fig. 1.

All agents are initialized randomly on the grid. At each time step, each agent independently selects an action according to its policy, as described in the next section. Each agent has action space: {up, right, down, left, wait}. All agents make observations and select actions synchronously, but actions are executed in a fixed sequence. When an agent takes an action, it moves one cell in the selected direction, unless that cell is occupied by another agent or a wall. If the cell is occupied, the agent does not move. An agent might select an action but be unable to move if the cell it would

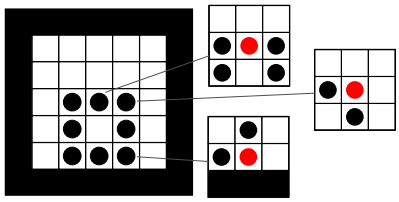


Fig. 1. Example of 8 agents forming a square in the discrete grid world, with examples of some agents' states on the right.

move to were previously empty but an agent earlier in the sequence had already occupied it, or if the cell were occupied but the agent still selected that action (for example, if it should not move, as there is no penalty for trying to move to an occupied cell).

The grid world is initialized with goal set X_G. For the example in Fig. 1, X_G contains the state shown and also the states where the agents form a square in each corner, along each side, and in the center of the grid. When the group reaches a state in X_G, an oracle rewards all agents:

$$R(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 & \mathbf{x} \in X_G \\ 0 & \mathbf{x} \notin X_G \end{cases}.$$

B. Policy Parameterization

The shape formation task is fully distributed, with no explicit communication between agents. Though the underlying MDP transitions are based on joint actions, each agent maintains its own policy and independently selects its own action. Each agent uses a parameterized stochastic policy giving the probability of taking each action from a given state observation. Agent i's policy is

$$\mu_{u_i}(\theta, y_i) = \frac{e^{\theta_{u_i y_i}}}{\sum_{u_i' \in U_i} e^{\theta_{u_i' y_i}}},$$

where $\theta_{u_i y_i}$ is the parameter related to action u_i and observation y_i, and θ is the matrix of $\theta_{u_i y_i}$ for all $u_i \in U_i$ and $y_i \in Y_i$. $\mu_{u_i}(\theta, y_i)$ is the probability of agent i taking action u_i from observation y_i. This policy representation lends itself to the use of gradient ascent techniques, which are used to find values of θ for a policy with maximum expected reward.

With this policy representation, the number of rows in θ is equal to the number of possible observations an agent can make. For several agents in a large state space, this is very large, and manipulation of θ can be a challenge. However, the policy can be computed independently for each observation.

C. Gradient Ascent

For each action-observation pair (u_i, y_i), $\mu_{u_i}(\theta, y_i)$ is continuous and differentiable $\forall\, \theta_{u_i y_i} \in \mathbb{R}$. The policy gradient can be computed analytically and is nonzero for only a few elements. Our approximation of the reward gradient uses the ratio

$$\frac{\frac{\partial \mu_{u_i}(\theta, y_i)}{\partial \theta_{u_i' y_i'}}}{\mu_{u_i}(\theta, y_i)} = \begin{cases} 1 - \mu_{u_i}(\theta, y_i) & u_i' = u_i, y_i' = y_i \\ -\mu_{u_i'}(\theta, y_i) & u_i' \neq u_i, y_i' = y_i \\ 0 & y_i' \neq y_i \end{cases}. \quad (5)$$

Since $\mu_{u_i}(\theta, y_i) \in (0, 1)\, \forall\, u_i, y_i, \theta$, this gradient is positive along the parameter for the action that was taken and negative along the parameters for actions that were not taken.

The policy gradient implementation follows Baxter and Bartlett's GPOMDP algorithm [19]. Our goal is to find the parameters θ that maximize the expected reward $\eta(\theta)$. The gradient of the expected reward can be approximated:

$$\nabla \eta_\beta(\theta) = \frac{1}{T} \sum_{t=1}^{T} R(X^t) \left[\beta^{T-t} \frac{\nabla \mu_{U_i^t}(\theta, Y_i^t)}{\mu_{U_i^t}(\theta, Y_i^t)} \right], \quad (6)$$

where T is the number of time steps over which the gradient is computed, $\frac{\nabla \mu}{\mu}$ as computed in (5) is a matrix evaluated for $u_i \in U_i$ and $y_i \in Y_i$, and $\beta \in [0, 1)$ is a discount factor.

Computing the gradient at a parameter value is done by running a simulation; the gradient (6) is computed iteratively by each agent as it moves around the grid world. Intuitively, $\nabla \eta_\beta(\theta)$ is positive for parameters that contribute to the agent receiving rewards and negative for parameters that cause it to receive fewer rewards. β decreases the gradient approximation with respect to parameters for observations seen early in the simulation. When a small value of β is used, $\nabla \eta_\beta(\theta)$ has low variance, but large bias. If a large value of β is used, $\nabla \eta_\beta(\theta)$ is less biased but extremely noisy.

The gradient ascent step follows Baxter *et al.*'s CONJPOMDP algorithm [25], a conjugate gradient algorithm based on the approximation computed in (6). It uses line search to find the point where the gradient is approximately zero in the search direction. When used for a single agent, the line search is repeated until the gradient is near zero, indicating that a local maximum has been reached.

For multiple agents, the best choice of parameters for one agent depends on all other agents' parameters. Thus, we iterate through the set of agents, and each agent completes one step of conjugate gradient ascent. At the end of each cycle of all agents taking one step toward an improved reward, we stop only if all agents have reached parameters with gradients near zero. This procedure is shown in Algorithm 1.

Algorithm 1 Multi-Agent Gradient Ascent

```
1:  procedure MAGRADASCENT(agents A, grid W, step size s_0, n)
2:      for all a_i in A do
3:          θ(a_i) ← ones
4:      end for
5:      converged ← false
6:      while not converged do
7:          for all a_i in A do
8:              θ(a_i) ← FINDGRADIENTZERO(A, W, a_i, s_0, n)
9:          end for
10:         ESTIMATEGRADIENT(A, W, n)
11:         if All gradients < ε then
12:             converged ← true
13:         end if
14:     end while
15: end procedure
```

Our algorithm MAGRADASCENT (Algorithm 1) calls two additional functions. FINDGRADIENTZERO follows Baxter *et al.*'s GSEARCH algorithm [25] to find the parameters where a_i's gradient crosses zero in the search direction. Our implementation follows the processing in GSEARCH exactly, but the processing is centralized instead of being done by a_i because all agents are needed to compute the gradient.

To compute the gradient in both MAGRADASCENT and FINDGRADIENTZERO, we call our function ESTI-

TABLE I
TRAINING TIME AND RESULTS FOR GROUPS OF DIFFERENT SIZES

Group size	Shape	Steps before training	Steps after training	Training time
2	Line	27	8	1 hour
4	Square	9290	36	24 hours
4	Line	9361	35	24 hours
5	Plus sign	75,896	111	24 hours
8	Square	129,504	4689	500 hours

MATEGRADIENT, which implements Baxter and Bartlett's GPOMDP algorithm [19] for all agents simultaneously. For T timesteps, all agents step around the grid according to their policies and each agent updates its own calculation of $\nabla\eta_\beta(\theta)$ as it makes observations and receives rewards. Even though one agent optimizes its parameters at a time, all agents must step through the grid together.

V. EXPERIMENTAL RESULTS

A. Simulations

To evaluate our solution to Problem 3.2, we performed a series of experiments in the grid world simulation. In each experiment, a group of agents trained to reach a set of goal states, then the group's performance was evaluated. All agents were initialized with uniform policies selecting between all actions with equal probability. During training, the gradient ascent algorithm was iterated until the gradient was near zero or the group reached a set of policies that performed significantly better than random exploration. On each iteration, one step of gradient ascent was computed for each agent. This step required the reward gradient to be computed at several values of the policy parameters.

Agents initialize randomly in the grid and move according to their policy. Every time a goal state is reached, the group is given a reward and reset to a random initialization. Since the gradient approximation depends on the number of time steps used, it is computed after a fixed number of time steps that allows the group to complete the task several times. This is especially necessary early in training, when the agents' uniform policies cause significant variations in task completion times. The number of steps used is determined empirically and modified for each group size.

No group reached a true reward maximum, with all agents' gradients near zero. The random initial policies had nearly flat gradients, so groups took time to reach a policy that performed well. Simulations were stopped when good policies were reached (a subjective measure). While groups continued to improve with further training, as shown in Fig. 2, the rate of improvement after reaching a good policy was very slow.

All groups reached policies significantly better than random exploration. Table I gives the time required to train groups of different sizes and the average number of steps to reach a goal state before and after training. Training was performed on an Intel Xeon 3.5 GHz processor with 32 GB of RAM, in a single-threaded simulation written in Python.

All learned policies led all agents to converge in a corner of the grid. In moving to a corner, the agents implicitly agree on a landmark as a meeting point, which is much more efficient than searching the space to find each other. The choice of corner is related to randomly selected behaviors

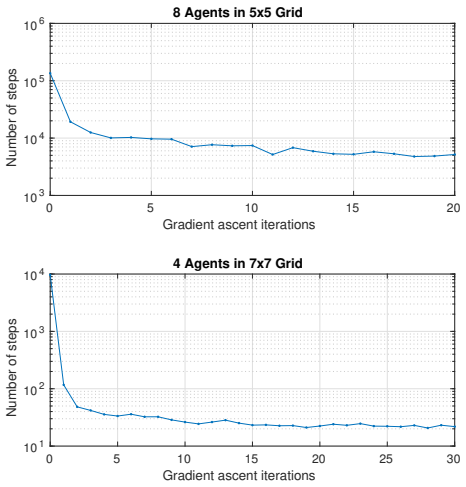


Fig. 2. Policy improvement over several iterations of gradient ascent for groups of 8 agents forming a square and 4 agents forming a line. Initially, all agents have random policies. In each iteration, all agents take a single step in their parameter space.

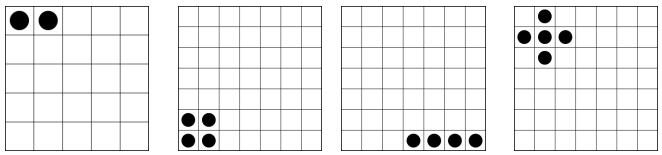


Fig. 3. Final positions of groups of 2, 4, and 5 agents after training. All groups reached policies completing the goal shape in a corner of the grid.

during training, since policies converging at any corner are equally good and equidistant from the initial uniform policies. This leads different groups to converge to different corners (c.f. Fig. 3). Figure 4 shows the behavior of all agents during one evaluation of shape formation. All agents prioritize moving to the bottom left corner, where the group meets and then moves into a square.

While different groups converged in different corners, the policies of agents that learned to form the same shape in the same corner were similar. For example, groups that formed a square in the lower left corner had similar performance whether the group contained agents that had trained together, agents that had trained separately, or copies of the same agent. Despite learning to move to a particular corner, agents did not learn a specific role within the group; they were equally likely to end up in any position in the goal state.

Different group and grid sizes were evaluated. For 2 agents in a 5×5 grid with a goal shape of two agents in horizontally adjacent cells, groups reached a good policy in about an hour.

We performed a parameter analysis for 4-agent groups in a 7×7 grid, with both square and horizontal line goal shapes. Good policies for both were reached in approximately 24 hours. These policies were used to evaluate the effects of the discount factor β and the goal shape. Results are shown in Table II and described further below.

When an agent uses a larger value for discount factor β, each state transition it encounters contributes more to the gradient approximation because its contribution does not decay as fast. As a result, the approximation made with a larger β has lower bias but higher variance. Similarly, with a smaller β, the approximation is more biased, but has lower variance. Groups of 4 agents were evaluated after being

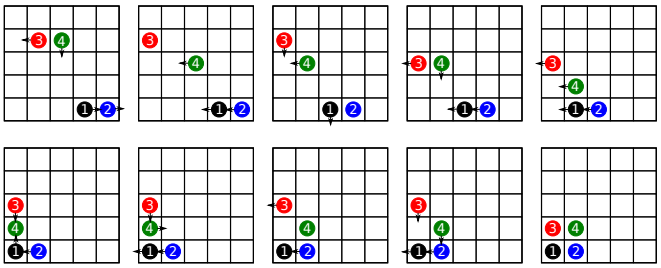


Fig. 4. A policy learned by 4 agents to form a square, beginning with random initialization in the top left and continuing from left to right. At each step, agents choose the actions indicated or no action and take those actions in order. All agents have learned to move toward the lower left corner first, then form the final shape. Further training would reduce the frequency of non-optimal actions, but they would not disappear entirely because the agents' policies have nonzero probability for every action.

TABLE II
AVERAGE NUMBER OF STEPS TO COMPLETE GAME AFTER TRAINING FOR 4 AGENTS FORMING A SQUARE OR A LINE

Shape	$\beta = 0.5$	$\beta = 0.9$
Square	44	100
Line	37	82

trained with $\beta = 0.5$ or 0.9. All groups reached significantly better policies with $\beta = 0.5$ than with $\beta = 0.9$. This is because initially the agents move randomly until they find a goal state. With larger β, this reward contributes more to the gradient for parameters related to observations seen earlier, which have little to do with the actions that led to the reward. As a result, the policies learned with larger β select more nonoptimal actions and take longer to converge.

Goal shapes like a square with 4 agents are simpler than the general problem because all agents can observe all other agents and therefore know when a goal state is reached. In general, any individual agent can only observe part of the shape, as in Fig. 1. For 4 agents forming a line, agents may incorrectly believe they are in a goal state, such as when the agents are in two groups of two. However, even in this problem, the groups learn a good policy. Feedback from the oracle forces the agents to develop indirect coordination, so all agents move to the same landmark even without knowing how many others they must coordinate with.

We also evaluated larger groups of 5 agents in a 7×7 grid forming a plus sign, and 8 agents in a 5×5 grid forming an empty square, as in Fig. 1. Larger groups required significantly longer training times to reach similar results, as shown in Table I, for several reasons. First, goal states were less frequent during random exploration, so a single gradient computation required more steps. Next, each step required more agents to take an action. Finally, each agent had more possible observations. For example, with two agents, each has 49 possible observations. With 8 agents, each observes up to 7 others and has 6561 possible observations. The policy has one parameter for each state-action pair, so each agent estimates gradients along 32,805 parameters. Despite the size of this computation, the agents reach good policies. With more agents, the number of observations will stop growing and the time for each step will scale only linearly with the number of agents (since each agent must take an action). Each agent observes only 8 grid cells, so in a group of 9 or more agents, each will have only 6562 possible observations.

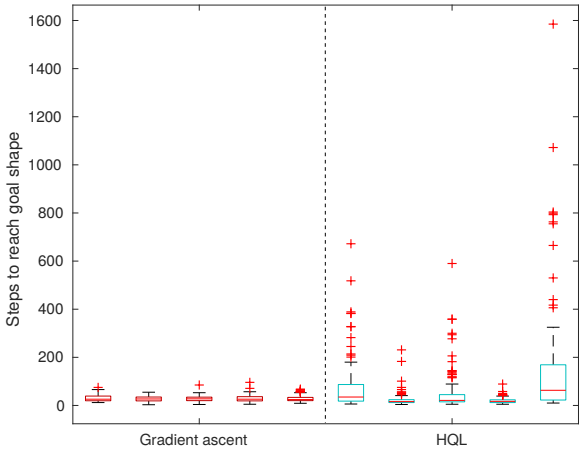


Fig. 5. For each algorithm, 5 different groups were trained and evaluated on 100 random initializations. This plot shows the performance of these groups, with each column showing the results from one group. Though HQL groups have median performance similar to gradient ascent groups, their worst-case performance tends to be much worse.

B. Comparison to Hysteretic Q-Learners

To compare our approach to other model-free methods, we implemented HQL [16] in the grid world simulation. HQL also fulfills our requirements for a solution to Problem 3.2: agents can learn by exploring the state space together, cannot communicate explicitly, and are not required to model either the space or other agents' behavior.

In HQL, each agent maintains a Q-value function, which contains the expected reward for each state-action pair. After each step, each agent a_i updates its expected reward Q_i according to:

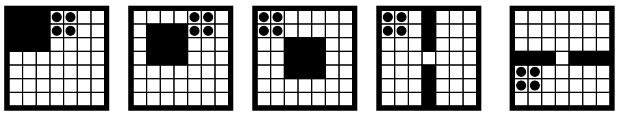
$$\delta \leftarrow R(X^t) + \gamma \max_{u_i \in U_i} Q_i(y_i^{t+1}, u_i)$$

$$Q_i(y_i^t, u_i) \leftarrow \begin{cases} (1-\alpha_H)\, Q_i(y_i^t, u_i) + \alpha_H\, \delta & \delta > Q_i(y_i^t, u_i) \\ (1-\beta_H)\, Q_i(y_i^t, u_i) + \beta_H\, \delta & \text{otherwise} \end{cases},$$

with learning rates $\alpha_H > \beta_H$. Using this update, agents improve their estimates of Q as other agents learn policies, so they can predict state transitions without fully modeling the other agents. By using different learning rates, they learn quickly when coordination that leads to a reward is observed, but react more slowly to other agents' exploratory behavior, which may decrease Q-values.

We evaluated groups of 4 hysteretic Q-learners forming squares and compared them with groups learning with policy gradients. In general, the hysteretic Q-learners learned a policy with similar median performance to gradient ascent agents, but there was much more variance in individual evaluations of the same policy and in the quality of learned policies between different groups, and those policies did not adapt as well to obstacles in the space. It is notable that in policies learned using HQL, agents often converged on a wall rather than in a corner, as they did in policies learned with gradient ascent.

Figure 5 shows the results of training five groups with each algorithm, with each column representing one group's performance. To evaluate this, we used each algorithm to train five groups with no initial policies. The hysteretic Q-learners were trained with $\alpha_H = 0.1$ and $\beta_H = 0.01$,



(a) (b) (c) (d) (e)

Fig. 6. Environments with obstacles, and common goal states that groups found. (a) Block in corner. (b) Block near corner. (c) Block in center. (d) Vertical wall. (e) Horizontal wall.

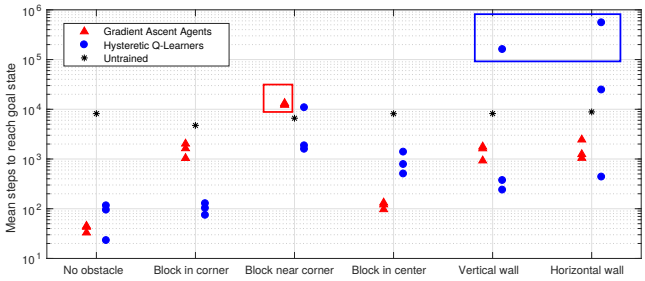


Fig. 7. In spaces with each type of obstacle, 3 groups of agents trained with each algorithm were evaluated on 100 random initializations. This plot shows the mean number of steps for each group to reach a goal state, and for an untrained group to reach a goal state in the same space. When the groups were evaluated with corner obstacles, the obstacles were placed in the corner where that group converged most often. Boxes indicate performance on the most difficult obstacles for each algorithm. Note the logarithmic scale.

as recommended by Matignon *et al.* [16], and used ε-greedy action selection, with $\varepsilon = 0.1$, during both training and evaluations [17]. While empirically HQL took fewer steps to reach a good solution, we trained for 5,000,000 steps to ensure convergence. For the policy gradient agents, each group performed 10 iterations of gradient ascent. Over that time, each agent typically performed about 12,000,000 computations toward gradient estimates and updated its parameters about 40 times. After each group was trained, its performance was evaluated over 100 randomly initialized games, and the average performance of each group was used to compute the values shown. The algorithms have a similar mean number of steps to complete a goal formation, indicating that the average policy learned performs similarly in the training environment. However, the HQL policies have a larger variance and long tails, which shows that the policies learned are less consistent and the performance of a learned policy is less predictable.

As mentioned previously, we are interested in policies that not only complete shape formation well, but that can adapt to changes in the environment. To that end, we created test environments with obstacles that might interfere with the formations that the groups would normally have formed. These obstacles were introduced *after* training the agents; in other words, the agents had not seen these obstacles before. The test environments are shown in Fig. 6, and the performance of groups that had trained with each algorithm in each of the five test environments is shown in Fig. 7.

Comparing worst-case performance of the policies, the hysteretic Q-learners performed extremely poorly in scenarios with an obstacle intersecting the wall they tended to converge on; this interfered most with the learned policy, as agents could get stuck on either side of it. Two of the groups of hysteretic Q-learners averaged over 100,000 steps to reach a goal state in this environment, with one taking over 500,000.

In contrast, the obstacle that interfered most with the gradient ascent agents' policies was a block one grid square away from the corner where the group converged: the group could not form the square behind it, and agents could get stuck behind the obstacle. However, all gradient ascent groups successfully completed the shape formation in about 13,000 steps, which is comparable to the 9290 steps they took before learning any policy. While this is not optimal, it demonstrates that gradient ascent agents are able to overcome foreign obstacles more consistently than the HQL agents.

C. Comparison to Model-Based Methods

Our approach to solving Problem 3.2 is model-free: agents do not have any explicit knowledge of the space in which they are moving, the goal states, the relationship between states, or the policies of other agents. Instead, they explore and directly observe only their observations and actions, with no model of the underlying state.

A common alternative approach to solving Dec-POMDPs is model-based [9]. Agents use an explicit model of the world to predict the most likely state given their observation. From that prediction and their confidence in it, the agent chooses the action with the highest expected reward. With this method, exact solutions require large policy trees that can be time-consuming to search. Approximate solutions can be found by sampling the policy space, but for a large policy space and a long horizon, a good solution may not be found.

We used the MultiAgent Decision Process (MADP) toolbox [26] to solve the Dec-POMDP of two agents in a 3×3 grid, which is small enough that the model and policies can be evaluated by hand. This scenario has 72 states and 25 joint actions, thus 1800 transitions must be specified. An agent can always determine its own position, but there are a few cases where the agents cannot observe each other, making the problem a Dec-POMDP. It is possible for the agents to reach a goal state in 2 steps from any initial state, so for a horizon of 2 steps, the expected value for a good policy is greater than 1. Some algorithms in the MADP toolbox produced approximate solutions. The Generalized Multiagent A* (GMAA) with Alternating Maximization [27] and Direct Cross-Entropy Policy Search (DICEPS) [28] solvers found policies with value above 1. Exact solvers were not able to find solutions, even after several hours of processing on the same computer used for the simulations.

For direct comparison, we used the MADP toolbox to evaluate the shape formation problem with two agents in a 5×5 grid, the smallest problem evaluated with policy gradients. As shown in Section V-A, a good policy was reached using gradient ascent in about an hour, and the best policies would take at most 8 steps to reach a goal state. This problem has 600 states and 25 joint actions, with 15,000 state transitions that must be specified. None of the exact algorithms in the MADP toolbox can process a model of this size. The DICEPS solver was only able to provide approximate solutions up to a horizon of 3 steps. Since an

equally good policy to those found with policy gradients would require a horizon of at least 8 steps, we were not able to compare the policies found with the policy gradient approach to those found with other methods.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have developed a model-free approach for a group of agents to learn distributed policies for coordination based on gradient ascent techniques. Our method is general: it can be applied to any problem requiring coordination and scaled to any group size, and the policies it learns are robust to changes in the environment. We have introduced *shape formation*, a multi-agent problem requiring tight coordination in a large group of agents, to demonstrate its performance. Using this method, we successfully trained groups of 2 agents in one hour and groups of 4 and 5 agents in 24 hours. We demonstrated its use on larger groups, including a group of 8 agents learning to form a square. We also showed that policies learned using this method perform better with new obstacles in the environment than those learned using HQL, another model-free approach.

Our gradient ascent method performed significantly better than model-based methods in finding policies for shape formation. On the very small problem of 2 agents in a 3×3 grid, approximate solvers (in the MADP toolbox) found policies matching the expected performance, but exact solvers were not able to find solutions. This problem was smaller than any that we used to evaluate our method. On a larger problem, a group of 2 agents in a 5×5 grid, our method reached good policies in about an hour, but even the approximate model-based solvers could not find solutions.

In the future, we would like to explore using the policy gradient approach on larger groups. While the training time would grow more slowly above 9 agents, as described in Section V-A, training large groups would still take significant time. More efficient methods for training large groups of agents are necessary; possible approaches include parallelizing the training process or training initially with smaller groups, then combining them. Furthermore, we plan to work toward quantifying the requirements for our policies, which will be necessary to understand how those policies can be used in real applications.

REFERENCES

[1] S. Wang, B. Krishnamachari, and N. Ayanian, "The optimism principle: a unified framework for optimal robotic network deployment in an unknown obstructed environment," in *IEEE/RSJ Intl Conf Intelligent Robots and Systems*, 2015, pp. 2578–2584.

[2] B. Anderson, B. Fidan, C. Yu, and D. Walle, "UAV formation control: Theory and application," *Recent Advances in Learning and Control*, pp. 15–33, 2008.

[3] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized Controllers for Shape Generation with Robotic Swarms," *Robotica*, vol. 26, no. 5, pp. 691–701, 2008.

[4] K. Kaplan. (2016, Nov. 4) 500 drones light night sky to set record. [Online]. Available: https://iq.intel.com/500-drones-light-show-sets-record/

[5] T. M. Cheng and A. V. Savkin, "Decentralized control of multi-agent systems for swarming with a given geometric pattern," *Computers and Mathematics with Applications*, vol. 61, no. 4, pp. 731–744, 2011.

[6] L. Pimenta, G. Pereira, M. Gonçalves, N. Michael, M. Turpin, and V. Kumar, "Decentralized controllers for perimeter surveillance with teams of aerial robots," *Advanced Robotics*, vol. 27, no. 9, pp. 697–709, 2013.

[7] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *Intl J Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[8] D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of markov decision processes," in *Proc Conf on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 32–37.

[9] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[10] C. Amato, G. D. Konidaris, and L. P. Kaelbling, "Planning with macro-actions in decentralized POMDPs," in *Autonomous Agents and Multi-Agent Systems*, 2014, pp. 1273–1280.

[11] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling, "Planning for decentralized control of multiple robots under uncertainty," in *IEEE Intl Conf Robotics and Automation*, 2015, pp. 1241–1248.

[12] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc Intl Conf Machine Learning*, 2000.

[13] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans Systems Man and Cybernetics Part C Applications and Reviews*, vol. 38, no. 2, p. 156, 2008.

[14] M. J. Mataric, "Learning to behave socially," in *Intl Conf Simulation of Adaptive Behavior*, vol. 617, 1994, pp. 453–462.

[15] T. Recchia, J. Chung, and K. Pochiraju, "Improving learning in robot teams through personality assignment," *Biologically Inspired Cognitive Architectures*, vol. 3, pp. 51–63, 2013.

[16] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *IEEE/RSJ Intl Conf Intelligent Robots and Systems*. IEEE, 2007, pp. 64–69.

[17] ——, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[18] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Intl Conf on Machine Learning*, 2017, pp. 2681–2690.

[19] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[20] L. Peshkin, K. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to cooperate via policy search," in *Proc Conf Uncertainty in Artificial Intelligence*, 2000, pp. 489–496.

[21] O. Buffet, A. Dutech, and F. Charpillet, "Shaping multi-agent systems with gradient reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 15, no. 2, pp. 197–220, 2007.

[22] J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[23] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[24] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *arXiv preprint arXiv:1705.08926*, 2017.

[25] J. Baxter, P. L. Bartlett, and L. Weaver, "Experiments with infinite-horizon, policy-gradient estimation," *J Artificial Intelligence Research*, vol. 15, pp. 351–381, 2001.

[26] F. A. Oliehoek, M. T. J. Spaan, P. Robbel, and J. V. Messias, "The MADP toolbox: An open-source library for planning and learning in (multi-)agent systems," in *Sequential Decision Making for Intelligent Agents—Papers from the AAAI 2015 Fall Symposium*, Nov. 2015, pp. 59–62.

[27] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Approximate solutions for partially observable stochastic games with common payoffs," in *Autonomous Agents and Multiagent Systems*, 2004, pp. 136–143.

[28] F. A. Oliehoek, J. F. Kooij, and N. Vlassis, "The cross-entropy method for policy search in decentralized POMDPs," *Informatica*, vol. 32, no. 4, pp. 341–357, 2008.