2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Implicit Adaptive Multi-robot Coordination in Dynamic Environments

Mitchell Colby, Jen Jen Chung and Kagan Tumer

Abstract— **Multi-robot teams offer key advantages over single robots in exploration missions by increasing efficiency (explore larger areas), reducing risk (partial mission failure with robot failures), and enabling new data collection modes (multi-modal observations). However, coordinating multiple robots to achieve a system-level task is difficult, particularly if the task may change during the mission. In this work, we demonstrate how multiagent cooperative coevolutionary algorithms can develop successful control policies for dynamic and stochastic multi-robot exploration missions. We find that agents using difference evaluation functions (a technique that quantifies each individual agent's contribution to the team) provides superior system performance (up to 15%) compared to global evaluation functions and a hand-coded algorithm.**

I. Introduction

Well-coordinated multi-robot teams can accomplish complex tasks in dynamic and stochastic environments. However, many current coordination strategies require these tasks to be well-defined *ahead of time* so that they can be decomposed and allocated to specific agents.

In exploration missions, though, only high level descriptions of the mission can be given prior to execution since the exact locations of points of interest (POIs) may be unknown *a priori* and new POIs are often detected on-the-fly. In circumstances where explicit instruction from a human operator is unavailable, coordination during operation is necessary for the detection and evaluation of potential POIs as well as when deciding how to re-allocate robots and team resources to adapt to the changing mission environment.

The survey in [1] outlined the difficulties of learning coordinating behaviors for multi-robot teams. Many of the noted issues arise directly from the fact that multi-robot problems can grow rapidly in complexity and dimensionality, so much so that it becomes intractable to obtain sufficient learning data to develop coherent team strategies. Market-based approaches, such as in [2] and [3], attempt to reduce complexity by limiting robot interaction to bidding for tasks based on individual preferences, while the behavior-based approaches developed in [4] and [5] remove the need for inter-robot communication by computing internal motivators based on the mission requirements to determine a robot's fitness for a task. In both of these scenarios, brevity is obtained by condensing prior knowledge of mission requirements into individual robot preferences.

In cases where this mapping between tasks to preferences does not exist due to incomplete prior knowledge of the mission, external feedback on team performance throughout the mission can be used to guide policy refinement. Methods such as value and policy iteration in reinforcement learning and fitness evaluation in evolutionary algorithms are able to use feedback signals to successively improve control policies.

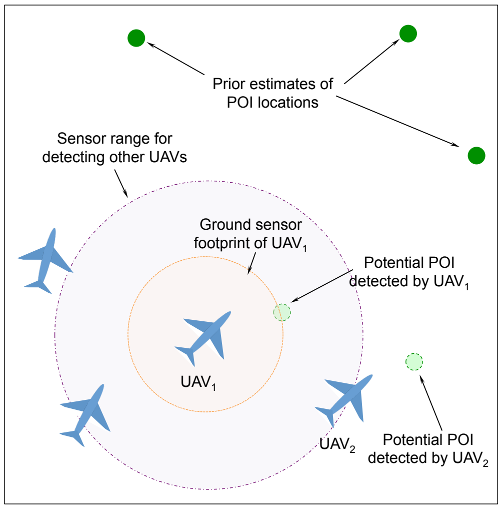
Prior estimates of POI locations
Sensor range for detecting other UAVs
Ground sensor footprint of UAV$_1$
Potential POI detected by UAV$_1$
UAV$_1$
UAV$_2$
Potential POI detected by UAV$_2$

Fig. 1: Multi-robot coordination problem with limited prior information. UAVs begin the mission with prior estimates of POI locations and can detect potential POIs as well as other UAVs within the respective sensing radii. UAVs within the team must coordinate to allocate sensing tasks.

One of the persistent challenges in learning multi-robot team behavior is in devising an appropriate decentralized agent feedback framework that can be translated across different mission specifications and adapt to new feedback from external operators. This is in addition to the difficulty of designing a decentralized feedback signal that clearly maps to the actions taken by the robot while also faithfully representing the global team performance. These two properties are termed the "sensitivity" and "alignment" of a feedback function, respectively [6]. A highly sensitive feedback signal changes mainly due to the actions of the learning robot and is less affected by the noise of other robots' actions, while a feedback signal with strong alignment implies that a robot learning to improve this signal will also learn to improve the global system performance. In many existing agent-feedback functions, these two properties must be traded-off against one another; the challenge is in designing a utility that incorporates both of these properties.

In this paper we use the difference evaluation function [6] from the multiagent literature to learn new robot team behaviors and adapt to changing mission environments without pre-arrangement or explicit communication between team members. We also show how external feedback can be folded

Mitchell Colby, Jen Jen Chung and Kagan Tumer are with the Autonomous Agents and Distributed Intelligence Lab, Faculty of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR, 97330, USA `colbym@engr.oregonstate.edu`, `{jenjen.chung, kagan.tumer}@oregonstate.edu`

into the learning algorithm via global system evaluations.

We demonstrate our algorithm on an environmental survey mission across an unknown region with only high level mission requirements provided prior to execution. Our results show that the team of unmanned aerial vehicles (UAVs) are able to explore for POIs in the region and dynamically reallocate UAVs as new tasks arise. We also show that the team is able to adapt its behavior given performance feedback provided by human operators throughout the mission.

II. Problem Description

The problem that we consider in this paper is one of multi-UAV coordination for environmental monitoring in an unknown region, Fig. 1 illustrates the basic problem setup. A team of UAVs is tasked to discover and observe POIs in a previously unexplored environment. Prior knowledge of the number of POIs, their locations, and the utility of observing any particular POI is uncertain and so the UAV team must perform task assessment and assignment on-the-fly as new POI information is collected. POIs, as defined in this problem setup, can extend beyond static features to also encompass temporal events such as the appearance of hotspots in a spatio-temporal field as well as dynamic targets that move in and out of sensor view.

With a single UAV agent or a small team of UAVs, it may be possible for an external operator to assess the performance of each team member throughout execution and provide individual feedback for policy updating. However in environmental surveillance missions, it is more common for data from the team to be received in batches, thus timely assessment of individual performance becomes intractable particularly with many agents. A single assessment of the entire team performance is relatively simple to compute and can feasibly be returned by an external operator, however, the issue then becomes one of credit assignment, that is, how does each UAV determine its contribution to the team performance? This contribution can then be used by each UAV agent to assess and improve its individual policy.

In the following experiments, each UAV in the team executes a policy described by a neural network controller. Neural networks are capable of approximating any function to arbitrary accuracy and so have the representational ability to encode continuous state-action control policies while only requiring a coarse representation of the system state [7], [8]. Training a neural network involves setting network weights to encode a control policy which accepts the current state as an input and returns the action that maximizes a particular utility function. The performance of the learned controller is strongly dependent on the utility function used to evaluate policies. In this work, we train neural network controllers with cooperative coevolutionary algorithms using both difference and global evaluation functions to assign fitness. Coevolution is detailed in the following section.

III. Evolutionary Algorithms for Learning Multi-Robot Control Policies

A. Evolutionary Algorithms

Evolutionary Algorithms (EAs) are stochastic search algorithms that often outperform classical optimization algorithms, particularly in domains where the gradient of the system evaluation function is unknown [9]. An EA typically contains three basic mechanisms: solution generation, mutation, and selection. These mechanisms act on an initial set of candidate solutions (the population) to generate new solutions and retain solutions that show improvement in system performance. While simple EAs can be applied to a variety of learning tasks, they need to be extended in order to perform effectively in large cooperative multi-robot problems. One such modification is *coevolution*, where multiple populations evolve simultaneously in order to develop policies for interacting agents.

B. Cooperative Coevolutionary Algorithms

Cooperative Coevolutionary Algorithms (CCEAs) are an extension of EAs and have been shown to perform well in cooperative multiagent domains [10]. Multiple populations evolve in parallel, with each population developing a policy for one of the robots in the team. Policies from each population are drawn to form the a team of robots, and the overall performance of this team is evaluated using a fitness function $F(z)$, where z is the joint state of all robots in the team. This fitness is then assigned to each policy in the team. A general CCEA algorithm is shown in Algorithm 1. There are N coevolving populations of neural network controllers, one for each robot in the team. Each coevolving population has a population size of k. For mutation of the neural networks, 10% of the network weights are chosen at random and values drawn from a normal distribution (zero mean and unit variance) are added to the weights.

The key difference between a CCEA and an EA is the assignment of fitness to each evolving agent. In a traditional EA, the fitness of a policy is simply the system performance attained by that policy. However, in a CCEA, all the agents in the team affect the overall system performance; this means that the fitness of a robot's policy is based on its interactions with all other robots it collaborates with, resulting in context-dependent and subjective fitness assignment [11]. For example, a robot executing a severely suboptimal policy collaborating with a team of robots with near-optimal policies may still lead to good system performance. In this case, it is difficult to determine that the suboptimal policy should actually receive a low fitness assignment. This phenomena is often seen in multiagent systems and is known as the *credit assignment problem*: given a team of agents and the overall system performance, it is difficult to determine the relative usefulness of each individual agent. In order to address the credit assignment problem, we focus on shaping fitness functions with the difference evaluation function.

Algorithm 1 Standard CCEA

```
1:  Initialize $N$ populations of $k$ neural networks
2:  for each Generation do
3:      for each Population do
4:          produce $k$ successor solutions
5:          mutate successor solutions
6:      end for
7:      for $i = 1 \rightarrow 2k$ do
8:          randomly select one policy from each population
9:          create team $T_i$ of agents with selected policies
10:         simulate $T_i$ in domain
11:         assign fitness to each agent in $T_i$ using $F(z)$
12:     end for
13:     for each Population do
14:         select $k$ networks using binary tournament
15:     end for
16: end for
```

IV. Difference Evaluation Function

The agent-specific *difference evaluation function* is defined as in [12]:

$$D_i \equiv G(z) - G(z_{-i} + c_i), \tag{1}$$

where $G(z)$ is the total system evaluation, z_{-i} are all the states on which agent i has no effect, and c_i is the *counterfactual* term, which is a fixed vector to replace the effects of agent i. Intuitively, the second term in (1) evaluates the fitness of the system without the effects of agent i, so the difference evaluation gives agent i's contribution to the system evaluation [12]. Note that:

$$\frac{\partial D_i(z)}{\partial a_i} = \frac{\partial G(z)}{\partial a_i}, \tag{2}$$

where a_i is the action of agent i. This means that an agent acting to increase the difference evaluation function also increases the overall system performance; this property is termed *alignment*. Further, the second term in (1) removes portions of $G(z)$ that are not related to agent i, resulting in an improved signal-to-noise ratio; this property is termed *sensitivity*. It was shown in the authors' prior work [12], [13] that the alignment and sensitivity properties of the difference evaluation function results in agent-specific feedback, which allows for superior policies to be learned.

Difference evaluations may be used to define $F(z)$ in the fitness assignment stage of Algorithm 1; however, in (1), we see that computing these evaluation functions requires knowledge of the state of all UAVs in the system z_{-i}, as well as the functional form of $G(z)$. In practice, such information is never known to each agent. Computing the difference evaluation function thus requires approximation.

A. Local Approximation

The algorithm to evaluate the difference evaluation function was first presented in [13], and is presented in Algorithm 2. We assume that a cooperative multi-robot system aims to coordinate in order to maximize some system-level objective function $G(s, a)$, where we have decomposed the system state vector z into a system state vector s and a system action vector a. We assume that the value of $G(s, a)$ is broadcast to each agent, and that each agent i has access to its local state s_i and action a_i. At each timestep, each agent takes an action, and $G(s, a)$ is broadcast to each agent. Each agent i maintains a private neural network approximation for $G(s, a)$, denoted $\hat{H}_i(s_i, a_i)$. The agent's local state s_i, local action a_i, and value of the system objective function $G(s, a)$ are used to update the $\hat{H}(s_i, a_i)$ approximation using backpropagation. Each agent's approximation of $G(s, a)$ is a mapping from that agent's local state and action to the system objective function. This approximation is initially very noisy, because the state and action information is limited to only one agent's state and action. However, as the policies of each agent begin to converge, approximating $G(s, a)$ using only one agent's state and action becomes less noisy, because the variance in the joint action for a particular system-level state is reduced. Given an agent's approximation $\hat{H}_i(s_i, a_i)$, the difference evaluation is estimated as:

$$\hat{D}_i(s, a) = G(s, a) - \hat{H}_i(c_{s,i}, c_{a,i}), \tag{3}$$

where $\hat{D}_i(s, a)$ is agent i's approximation of the difference evaluation function. In order to evaluate $\hat{H}_i(c_{s,i}, c_{a,i})$, a default state $c_{s,i}$ and default action $c_{a,i}$ are chosen. In other words, the approximation of the difference evaluation function determines *the difference between the system objective function and the approximated system objective function if agent i took a default action.*

It is worth noting here that there is no requirement for $\hat{H}$ to approximate $G(z_{-i} + c_i)$. $\hat{H}$ simply needs to represent a function that is aligned with G and does not depend on z_i. Computing $\hat{H}$ using the method outlined above is a sensible choice given these requirements and the available information, however other choices of $\hat{H}$ may also be possible.

V. Experiment Setup

We compared the performance of the learning algorithm using the approximated difference evaluation function signal to one using the global evaluation signal on the UAV exploration problem described in Section II. We also show the best case performance of the algorithm using the exact difference evaluation function signal.

A. UAV Parameters

We investigate the performance and scalability of our learning algorithm over two team sizes, a smaller team with 15 UAVs and a large team of 50 UAVs. Each UAV is equipped with a downward-facing sensor to observe POIs on the ground and an in-plane bearing sensor to detect other UAVs within sensor range. The range of the in-plane bearing sensor, r_{planar} is twice the radius of the downward-facing sensor footprint, r_{ground}. Both sensors take measurements with additive white noise drawn from $\mathcal{N}(0, 0.1)$.

Sensor detections of POIs and other UAVs are discretized into four quadrants, {*north-east*, *north-west*, *south-west*, *south-east*}, with respect to the UAV body frame. The detections within each quadrant are used to find the state

Algorithm 2 CCEA using $\hat{D}_i(s,a)$

```
 1: Initialize $N$ populations of $k$ neural networks
 2: for each Population $i$ do
 3:     for each Individual $j$ do
 4:         Initialize function approximator $\hat{H}_{i,j}(s_i, a_i)$
 5:     end for
 6: end for
 7: for each Generation do
 8:     for each Population do
 9:         produce $k$ successor solutions
10:         mutate successor solutions
11:     end for
12:     for $i = 1 \to 2k$ do
13:         randomly select one policy from each population
14:         create team $T_i$ of agents with selected policies
15:         for each Simulation timestep do
16:             each agent $j$ in team $T_i$ finds local state $s_i$
17:             each agent $j$ in team $T_i$ chooses action $a_i$
18:             $G(s,a)$ is broadcast to each agent
19:             update $\hat{H}_{i,j}(s_i, a_i)$ based on $s_i$, $a_i$, $G(s,a)$
20:             $\hat{D}_{i,j}(s,a) = G(s,a) - \hat{H}_{i,j}(c_{s,i}, c_{s,a})$
21:             each agent $j$ increments fitness by $\hat{D}_{i,j}(s,a)$
22:         end for
23:     end for
24:     for each Population do
25:         select $k$ networks using binary tournament
26:     end for
27: end for
```

inputs of the neural network controller. More specifically, the state variable representing UAV detections in quadrant q of UAV i is defined as:

$$s_{UAV_{i,q}} = \sum_{i' \in N_q} \frac{1}{L_{i'}}, \tag{4}$$

where N_q is the set of observable UAVs in quadrant q, and $L_{i'}$ is the relative distance from UAV i to UAV i'. The state variable representing POI detections in quadrant q of UAV i is defined as:

$$s_{POI_{i,q}} = \sum_{j \in I_q} \frac{V_j}{L_j}, \tag{5}$$

where I_q is the set of observable POIs in quadrant q, V_j is the value associated with POI j, and L_j is relative distance from UAV i to POI j.

At each decision instance, each UAV queries its learned policy to compute an airspeed and a heading change to execute in the next step. The learned policies are encoded by 8-input, 10-hidden unit, 2-output fully connected feedforward neural networks with sigmoidal activation functions. The neural network output provides two values that lie in $[0, 1]$ and each of these are mapped to one of the two controls. The airspeed control is a linear mapping to $[v_{a_{min}}, v_{a_{max}}]$, the range of airspeeds between the minimum and maximum allowable airspeed, while the heading changes are also mapped linearly from $[-90°, 90°]$. In the following experiments, the survey region is 50×50 units in size; the minimum and maximum allowable airspeeds are defined as $v_{a_{min}} = 0.25$ units/timestep and $v_{a_{max}} = 1$ unit/timestep.

B. POIs

The survey region contains 25 POIs that are initialized according to a uniform random distribution across the search space. During each mission, UAVs can only sense POIs that are active and within sensor range. At initial execution, there are n POIs active in the entire survey region; as the mission progresses, more POIs become active at a rate of n POIs per m timesteps, where n and m are chosen based on the total number of POIs and overall simulation timesteps such that all POIs are active after 90% of the mission time. For the following experiments, $n = 5$ and $m = 9$.

Each POI is associated with a radius, $r_{POI} \ll r_{ground}$, such that a POI is considered fully observable to a UAV within this radius. For the following experiments $r_{POI} = 1$.

C. Utility Assignment

A single value is assigned for each POI within sensing range of a UAV, the value of observing POI i is computed as

$$w_i = \begin{cases} V_i & \text{if } d_{near} \le r_{POI}; \\ V_i/d_{near} & \text{if } r_{POI} < d_{near} \le r_{ground}; \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In (6), d_{near} is the lateral distance from the POI to the nearest UAV. The global system performance is evaluated as the sum of the observation values over all POIs,

$$G(z) = \sum_{i=1}^{k} w_i. \tag{7}$$

In the following experiments, the optimal global performance is achieved when all POIs are perfectly observed within one epoch (50 timesteps). The maximum global evaluation in this case is set to 50, thus with all POIs of equal observation value, $V_i = 50/k$.

The design of the utility function is such that there is no benefit to having multiple UAVs observe the same POI. In the following experiments we analyze the ability of the difference evaluation function signal to express this aspect of the mission goal and generate cooperative team behaviors without requiring explicit instruction for UAVs to monitor unobserved POIs.

D. Hand-Coded Patrolling Agents

In addition to comparing the learning performance obtained using the approximated difference evaluation versus the global evaluation, we also compare the performance of the learned policies against that achieved by a basic patrolling algorithm. In the patrolling algorithm for n UAVs, the world is discretized into n equal width strips. Each UAV occupies one strip, and simply patrols back and forth along that strip at a constant airspeed. This patrol algorithm guarantees full coverage of the physical area of the domain.

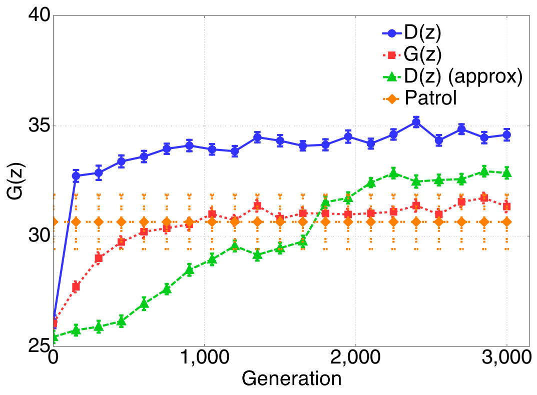

Fig. 2: POI surveillance performance of 15 UAVs with 10% sensor noise and full observability of the region. Mean global evaluations and standard error in the mean are plotted for the four tested algorithms. Policies trained on the approximated and analytical difference evaluations outperformed those trained on global evaluations and the hand-coded patrolling algorithm.

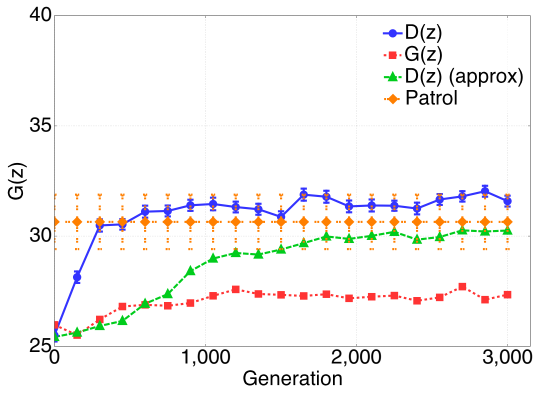

Fig. 3: POI surveillance performance of 15 UAVs with 10% sensor noise and limited sensing range. Policies trained on difference evaluations again provided the best system performance, those trained on approximated difference evaluations perform slightly worse than the patrolling algorithm but have far less variance in performance.

VI. Results

Three sets of experiments were conducted: the first set of experiments tested a team of 15 UAVs each with infinite sensor range, the results of these experiments demonstrated the performance baseline of the team when all members had full system observability. The second set of experiments involved a team of 15 UAVs equipped with sensors that had limited observation ranges. In these experiments, agents could detect other UAVs up to 20 units away, and could detect POIs up to 10 units away. In order to analyze the scalability of the proposed algorithm, the final set of experiments tested a team of 50 UAVs with the same limited-range sensor specifications. The following results are generated from 100 statistical trials of each set of experiments.

A. 15 UAVs, Full Observability

Averaged results of the global team evaluation with error bars reporting the standard error in the mean are given in Fig. 2. These results show that agents learning with the approximated or analytically computed difference evaluations outperformed agents using global evaluations or the hand-coded patrol algorithm. By providing agent-specific feedback, difference evaluations resulted in superior learned policies than agents using global evaluation functions as fitness assignment operators. These results suggest that in cases where there is insufficient information to directly compute difference evaluations, they can be approximated using the method described in Section IV-A and still provide superior performance over directly using the global evaluations.

Two observations may be made regarding the hand-coded patrol policy. First, it has fairly low performance even though it guarantees full domain coverage. This is due to the fact that full domain coverage is less important than POI coverage. In the patrol algorithm, when new POIs are discovered, nothing directs the UAVs to approach and observe these POIs. For example, if a UAV is patrolling along its assigned strip and a POI is discovered behind the UAV, the POI will not be observed until the UAV completes a patrol cycle and returns to the POI location. In cases with limited mission time, this POI may never be observed. The second observation regarding the patrol policy is the high variance in performance. In cases where POIs are active as a patrolling UAV is within its observation radius, the patrolling policy performs quite well. In cases where POIs become active far from patrolling UAVs, system performance suffers. Thus, although the patrolling policy performs well in some cases, it does not generalize to arbitrary POI configurations, and is thus limited in its applicability.

B. 15 UAVs, Limited-Range Sensors

As seen in Fig. 3, limiting the observation radius of UAV sensors decreases system performance by ~13% when using difference evaluations. In the limited observability case, agents using difference evaluation functions outperformed all other approaches analyzed. The patrolling policy resulted in slightly superior mean performance than agents using approximate difference evaluations. However the performance of the patrolling policy was far more variable for the reasons discussed in the previous subsection. Notably, the worst case patrolling performance achieved a global evaluation of 24.73 compared to 29.49 received by the poorest-performing policy trained on the approximated difference evaluation. This is again due to the stochastic nature of the environment and the ability of the learning algorithm to adaptively direct UAVs in the team to observe newly discovered POIs and improve coordination as learning progresses.

C. 50 UAVs, Limited-Range Senors

To investigate the scalability of the proposed algorithm, the number of UAVs in the system was increased from 15 to 50, and the number of POIs was increased from 25 to 85. Averaged results over 100 trials are shown in Fig. 4.

In the 50 agent case, policies trained on both the difference and approximated difference evaluations outperform those trained on the global evaluations and the patrol algorithm. As in the 15 agent case, the hand-coded patrolling algorithm exhibited a high variance in performance. Approximated difference evaluations produced policies that performed slightly better than the best case patrol algorithm, and performed far better than the worst case patrol algorithm.

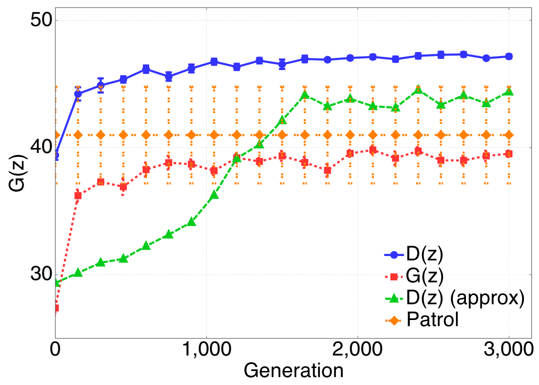

Fig. 4: POI surveillance performance of 50 UAVs with 10% sensor noise and limited sensing range. Both approximate and analytical difference evaluations outperform global evaluations and the basic patrolling algorithm.

It is of note that in all three sets of experiments, no UAVs came into conflict (assuming a conflict radius of 1). As the agents were learning to explore the domain, the learned control policies implicitly kept the UAVs "spread out". It is possible to add a penalty to discourage UAV conflicts in the fitness function, but was unnecessary in this work.

D. Alignment Analysis

Recall from (2) that the difference evaluation function and global evaluation function are aligned, meaning that an agent acting to increase its difference evaluation also increases system performance. However, this guarantee of alignment is lost when difference evaluations are approximated. We investigate how the degree of alignment of the approximate difference evaluation impacts performance.

We randomly sampled state-action pairs to evaluate the degree of alignment of the approximate difference evaluation function. We chose a random system state s and a random pair of joint actions $\{a, a'\}$, then computed G and $\hat{D}$ for each state-action tuple. If $G(s,a) \geq G(s,a')$ and $\hat{D}(s,a) \geq \hat{D}(s,a')$ or vice versa, the approximation is aligned with the system evaluation function for this state-action pair. We repeated this process for $500,000$ randomly generated state-action pairs and the results are shown in Table I.

Note that although approximation alignment increased from 15 to 50 agents, the relative performance of the approximation decreased. This is due to the increased complexity of the larger system. There are two interesting results in this alignment study. First, a strong correlation is seen between the degree of alignment and the relative performance of approximated and directly computed difference evaluations. Second, the approximation having high alignment is sufficient for high system performance. In a learning algorithm, the accuracy of the evaluation metric is less important than the relative ordering of state action pairs; as long as an approximator can determine the preference between two actions, then the accuracy of the estimated utilities of these actions are of minimal importance.

TABLE I
D̂ ALIGNMENT ANALYSIS

# of UAVs	$\hat{D}$ vs. G Alignment	$\hat{D}/D$ Performance
15	$92.19 \pm 0.81\%$	$96.84 \pm 0.12\%$
50	$94.44 \pm 0.05\%$	$92.47 \pm 0.41\%$

VII. CONCLUSIONS

In this work, we developed control and coordination policies for a team of UAVs in an environmental surveillance domain by coevolving neural network controllers and assigning fitness with difference evaluation functions. As the information required to compute difference evaluations is often unavailable in real world scenarios, each agent approximated the difference evaluation function using locally available information. Policies trained on the exact and also the approximated difference evaluations resulted in superior performance compared to those trained on global evaluations by up to 15%. The presented learning algorithms scale well to large UAV teams and are also able to adapt to the changing environment to provide far less variance in mission performance compared to a hand-coded patrolling algorithm.

ACKNOWLEDGMENT

This work was partially supported under NASA grant number NNX14AI10G.

REFERENCES

[1] L. Parker, "Multiple mobile robot systems," *Springer Handbook of Robotics*, pp. 921–941, 2008.

[2] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[3] L. Liu and D. A. Shell, "An anytime assignment algorithm: From local task swapping to global optimality," *Autonomous Robots*, vol. 35, no. 4, pp. 271–286, 2013.

[4] L. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[5] L. Parker, "Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance," *Autonomous Robots*, vol. 8, no. 3, pp. 239–267, 2000.

[6] A. K. Agogino and K. Tumer, "Analyzing and visualizing multiagent rewards in dynamic and stochastic environments," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 2, pp. 320–338, 2008.

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 3, 1989.

[8] P. Huang, J. Lehman, A. Mok, R. Miikkulainen, and L. Sentis, "Grasping novel objects with a dexterous robotic hand through neuroevolution," in *IEEE Symposium on Computational Intelligence in Control and Automation*, Orlando, FL, 2014, pp. 1–8.

[9] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.

[10] S. G. Ficici, O. Melnik, and J. B. Pollack, "A game-theoretic and dynamical-systems analysis of selection methods in coevolution," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 580–602, 2005.

[11] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[12] A. Agogino and K. Tumer, "Efficient evaluation functions for evolving coordination," *Evolutionary Computation*, vol. 16, no. 2, pp. 257–288, 2008.

[13] M. Colby and K. Tumer, "Approximating difference evaluations with local information," in *Proceedings of the Fourteenth International Joint Conference on Autonomous Agents and Multiagent Systems*, Istanbul, Turkey, 2015.