Autonomous Multiagent Space Exploration with High-Level Human Feedback

Mitchell Colby
Oregon State University
colby.mk@gmail.com

Logan Yliniemi
Oregon State University
logan.yliniemi@engr.orst.edu

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

Abstract

Extraterrestrial exploration missions are extremely difficult due to a number of factors. In particular, there exists high uncertainty in space environments, light-time delays, and high mission costs. Artificial intelligence based multiagent systems can address these concerns by creating autonomous multi-robot teams. Agents which can act autonomously in uncertain environments improve science returns on missions, and minimize the effects of light-time delays causing downtime while agents wait for instruction from Earth. A multi-robot system is also inherently robust, because the failure of a single robot will not cause total system failure; this robustness becomes especially critical as mission costs rise. In this work, we present a novel human in the loop cooperative coevolutionary algorithm to train a multi-robot system exploring an unknown environment. Autonomous agents learn to make low-level control decisions to maximize a science return, while human scientists on Earth can change high-level mission tasks to encourage different behavior. Results demonstrate that our algorithm reduces the number of robots needed for a particular performance level tenfold compared to traditional cooperative coevolutionary algorithms, resulting in significantly lower mission costs. Further, the trained multi-robot system is extremely robust to noise, and 10% sensor and actuator noise does not alter system performance in a statistically significant manner. Finally, the system is extremely robust to agent failures. When 10% of the robots in the system fail, the overall system performance is reduced by less than 10%.

1 Introduction

By their nature, extraterrestrial exploration missions (to planets, moons, comets, or asteroids) push the limits of technology and automation. To date, such exploration missions have been undertaken using a single monolithic explorer

(with the notable exception of *Spirit/Opportunity*, which only shared a fraction of their ground-based human resources [29]) which not only limits the science return, but also introduces a single, and very expensive point of failure. In addition, the vast distances and speed-of-light communications difficulties can lead to laboriously slow operation. Finally, such missions have relied of a high human-to-rover ratio. *Curiosity*, for example, relied on a team of 200 engineers and 400 scientists working on Martian time to perform all the necessary Earth-bound functions to operate at full capacity [42].

Autonomous multi-robot missions offer a compelling alternative to this paradigm, fundamentally addressing many of the difficulties of space exploration. Simply spreading the cost of a mission across multiple rovers removes the single point of full investment loss; a fatal error to one rover only decays the team's performance instead of becoming a complete loss. This allows for more risk taking on the part of individual robots, as well as the construction of each individual robot from cheaper parts. It also provides new capabilities: temporally-synchronized, but spatially-separated measurements can be taken to measure ephemeral phenomena that simply could not be captured by a single monolithic rover [12]. Cooperating surface and orbital vehicles can even observe the same phenomena simultaneously [9]. Multiple rovers can cooperate on complex tasks like construction, surpassing even the most capable single rover. Teams of less sophisticated rovers can even share long-range communications resources [8].

At first glance it might appear that this would simply multiply the problems of the ground-based human team: instead of a single rover for which to plan, optimize and operate, the team must now deal with N separate rovers, each in unique positions with unique needs that must be addressed. This is a difficult task even in ideal situations with perfect communication. Compounded by the speed of light communication delay and restrictions and the task becomes impossible for a human crew to keep up with.

However, these issues can be alleviated by an increase in autonomous capability of the rovers. Autonomy is very good at a specific subset of tasks that are currently subsidized by the Earth based human team of scientists and engineers, but not all of them. Thankfully the types of tasks that autonomy can perform are those that require the quickest feedback (navigation and coordination), and the tasks that uniquely require human expertise are ones that can be performed on a longer timescale (reconfiguring system-wide objectives); this circumvents the bottleneck resulting from speed-of-light communications delay. In this work we propose a novel human-in-the-loop multiagent learning system to address many of the difficulties found in the robotic extraterrestrial exploration. We consider a domain in which sensors are noisy, unreliable, and have a very limited range; actuators are noisy and may fail entirely; and whole rovers may cease functioning. Specifically, we focus on a mission profile that will incorporate a fully autonomous team of rovers made up of light "scouts" and heavier scientific "payload" platforms working simultaneously, which communicate with human decision makers only to modify/change goals on a longer time scale. Based on initial orbital surveillance, an Earth-based human team has identified a preliminary, incomplete set of "Points of Interest" (POIs). Once the team is deployed,

a series of "scout" vehicles (which could be lightweight, efficient rovers or small airborne vehicles like quadrotors) explore the area around the team, and send their initial photographs to a human team standing by on Earth. After some delay, the human team can process this initial low-resolution data and identify additional POIs. Simultaneously, the "payload" rovers use the most up-to-date list of POIs available to them and can analyze the POIs thoroughly with their more capable suite of scientific instruments. The team as a whole seeks to observe as many POIs as possible with the greatest possible detail, while providing graceful failure modes in the face of the harsh and unforgiving environment of space travel. This entire process is carried out in various different environments, so that the rovers are not simply learning a memorized path that they should execute, but their policies instead are broadly applicable to a wide variety of situations in which they may find themselves; thus, a small deviation from a modeled environment does not lead to a catastrophic failure, and the team will perform well even in highly uncertain environments. The contributions of this work are to:

- Provide a novel human-in-the-loop learning algorithm that allows humans to specify changing high-level mission objectives, while delegating low-level control tasks and coordination to the autonomous learning agents.
- Provide a difference-evaluation-based framework for structural credit assignment to encourage implicit coordination between agents.
- Demonstrate agents are robust to sensor and actuator noise and failure of other agents, while providing and reliable performance in a simulated extraterrestrial exploration domain in which difference evaluations increase team-wide performance by up to 33.5% over global evaluation functions.

The remainder of this work is organized as follows: Section 2 provides the necessary background and related work. Section 3 thoroughly describes the experimental setting and domain. Section 4 provides the details of our implemented algorithm. Section 5 provides the results of our experiments, and a discussion of the broader impact of these results. Finally, Section 6 concludes the paper.

2 Background and Related Work

In this section we provide the necessary background information on space exploration (Section 2.1), autonomous multiagent systems (Section 2.2), and the classic continuous rover problem (Section 2.3).

2.1 Space Exploration

Space exploration is fundamentally one of the most challenging domains that humans have ever tried to explore. The cost of transporting systems to another celestial body is high. The distances are vast. The environments are harsh, unforgiving, and difficult to fully observe. Each of these factors combines to makes

space exploration a unique challenge for robotic exploration missions: the high cost requires that a mission be carried out in a robust manner; the harsh environment requires careful observation and planning; and the long distances and speed-of-light communication restrictions makes the use of human intelligence to ensure the safe and reliable operation of these missions prohibitively slow [39].

However, the current state-of-the-art has defaulted to using human intelligence to thoroughly plan the day-to-day operations of the a rover. This allows for safe and robust operation, at the cost of hundreds of humans being required to operate a single rover, sometimes disrupting their lives for months at a time to operate based on the day/night cycle of the rover. While this is a solution that does provide for the ability to operate rovers on the surface of another planet, even with all of these efforts, the exploration accomplished by the rover is still painfully slow. *Spirit*, for example, travelled less than 8 kilometers over 6 years of operation. This all results from the speed-of-light communication delay, which requires that only a short series of commands be sent to the rover at a time.

However, the requirement that robust mission operation requires extremely safe and cautious maneuvering is mostly due to the fact that each mission conducted to date consists of a single, monolithic rover, and any single failure on the rover could end the mission [5]. Consider, for a moment, an alternative strategy: what if, instead of a single rover, a team of simpler rovers was sent, each with lesser capabilities than the single monolithic rover. As a team, they could cooperate to have capabilities that are greater than those of the monolithic rover. Because they could be built with some redundancy, the loss of a single rover would no longer end the mission as a whole, and instead would just decay the capabilities of the team as a whole.

This strategy would then remove the need for a vast team of human experts to plan each rover's individual day-to-day operations, and would delegate the planning on this level to an autonomous controller. This then becomes a multiagent system, with properties that we discuss in the following section.

However, there are certain aspects of this system that humans are still superior at to current autonomous technology. Defining the actual mission parameters as new discoveries are made and mission parameters change is a complex extrapolation problem that is beyond the current capabilities of autonomous systems. This process, however, can be conducted on a much longer time scale that will not be limited as harshly by the speed-of-light communication delay. In this manner, the human experts can still be a part of the overall mission, examining preliminary data and returning new mission parameters to the robotic team, which can incorporate these new priorities into their autonomous planning process.

Furthermore, the team of rovers doesn't necessarily need to be homogeneous. Different rovers with different capabilities could be used for different parts of the task. Initial investigations could be carried out by light rovers or even flying autonomous drones, which could then relay their low-resolution imagery back to Earth for processing. After identifying potentially interesting locations from those images, heavier rovers with a full suite of scientific instruments can then

thoroughly observe those points.

This creates a complex, heterogeneous multiagent system that has to coordinate its activities such that each rover is acting to achieve the overall team's mission. This type of a multiagent system creates many technical issues that must be addressed to achieve good performance, which are discussed in the following section.

2.2 Autonomous Multiagent Systems

With multiple agents (rovers) trying to work in tandem, the coordination of these agents becomes of paramount importance. No longer is a single agent trying to achieve all of the mission goals, but each individual agent is trying to perform tasks that will be lead to mission success. Sometimes it can be difficult to determine how each agent's actions will affect the overall mission's success, for a variety of reasons. First, actions taken "now" can have a strong effect on actions that can or should be taken a period of time later (temporal credit assignment), and second, actions taken by an agent A can affect the actions that agent B should take (structural credit assignment). For example, if agent B performs the exact same actions as agent A, its efforts are all for naught, because the observations made by rover A make the observations made by rover B redundant. This is the case even though if agent B was doing those exact same actions in a different context, they could be quite beneficial. Structural credit assignment is discussed in the following section.

As the number of rovers increases, it becomes less obvious to human intuition exactly what each rover should be doing to forward the goals of the team, making this an ideal situation for autonomy to determine the policies that each agent should follow. Adaptive agents can try out a number of different *policies*, and determine which ones are better suited to meeting the team's needs. There are a wide variety of techniques for doing this. In this work we focus on Evolutionary Algorithms (EAs) which in this team-based setting become Cooperative Coevolutionary Algorithms (CCEAs). These two concepts are discussed below.

An implicit advantage that is gained through the use of a multi-rover system is robustness to failure. The failure of a single component can, in the worst case, completely disable a single rover, while leaving the remainder of the rovers to complete the mission and achieve high team-wide performance. In a monolithic single-rover system, such a failure could lead to mission termination. In this way, simply by conducting a multi-rover mission, robustness to failure can be achieved. In addition, the team can also recover some system performance by continuing to adapt their policies through the CCEA, and in doing so will marginally compensate for the absence of the failed rover.

2.2.1 Structural Credit Assignment

When developing functions to evaluate specific agent performance in a multiagent learning system, there are two critical properties to consider: alignment

and sensitivity. An agent feedback function is said to be aligned with the system evaluation function if an agent acting to increase the feedback function also acts to increase the system evaluation function. An agent feedback function is said to be sensitive if it has a favorable signal to noise ratio based on the agent's actions; in other words, if a feedback function is a strong function of an agent's actions (and a weak function of other agent's actions), then it is sensitive. We now consider three structural credit assignment schemes, and analyze their alignment and sensitivity properties.

One question that quickly stands out is how exactly we can calculate what "better suited to meeting the team's needs" actually means in practice. Does this mean that each rover should try to acquire as much scientific data itself as possible? This is known as a **local evaluation (L_i)** scheme, and can lead to rovers failing to cooperate for the good of the team. In the example above, both rovers A and B would receive high local evaluations, because they are both making high-quality observations. However, the second identical set of observations does not provide any additional scientific value, and does not increase the value of the system evaluation function. In this case, the local evaluations are not aligned with the system evaluation function, but they are highly sensitive to the actions of each agent.

So then, should we simply score the team of rovers as a whole, giving them feedback based on the amount of scientific data that the entire team collected? This is known as a **global evaluation** (G) scheme, and has the drawback of not giving specific, personalized feedback to each rover. In our example above, either rover A or rover B choosing to remain stationary through the entire run would lead to no change in the global evaluation, and the rover that contributed nothing to the team would receive the same evaluation as the rover who was efficiently forwarding the team's efforts. Thus, the global evaluation scheme has low sensitivity to an individual agent's actions. However, it is perfectly aligned.

Ideally, a scheme should provide an individualized evaluation that is both *aligned* with the system-level reward, as well as *sensitive* to the actions of the individual agents. One scheme that is proven to be aligned, and shown to be sensitive is known as the **difference evaluation** (D_i) scheme, which calculates agent i's incremental contribution each agent has toward the system's performance. It is calculated as [2]:

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

where z is the system's state, $D_i(z)$ is the difference evaluation of agent i, $G(z)$ is the global evaluation, and $G(z_{-i} + c_i)$ is the global evaluation without agent i is removed from the system and replaced with a *counterfactual* agent c_i. Note that:

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

where a_i is the action of agent i. Thus, difference evaluations are perfectly aligned, and any action which increases the value of $D_i(z)$ also increases the

value of $G(z)$ [2, 41, 43]. Analyzing the second term in the difference evaluation, we see that it removes all portions of the system evaluation function not dependent on agent i. This dramatically improves the signal to noise ratio of difference evaluations, making them extremely sensitive to an agent's actions [1, 2, 40].

2.2.2 Cooperative Coevolutionary Algorithms

Evolutionary algorithms are a biologically-inspired class of stochastic search algorithms which often outperform classical optimization techniques, particularly, in complex domains where gradient information about the system evaluation function is unavailable [6, 16]. An evolutionary algorithm typically contains three basic mechanisms inspired by biological evolution: solution generation, mutation, and selection. These mechanisms are used on an initial set of candidate solutions (a population) to generate new solutions and to retain solutions which show improvement over time. Evolutionary algorithms have been successful in a wide variety of optimization problems, rocket guidance [18], helicoptor autopilot [21], autonomous robotics [31], market modeling [45], and previous studies on space exploration [38]. Although EAs are excellent tools, they need to be modified for implementation in large multiagent systems. One such modification is *coevolution*, where multiple populations evolve simultaneously to develop policies for interacting agents.

Coevolutionary Algorithms (CEAs) are an extension of evolutionary algorithms and are often well-suited for multiagent domains [14]. In a CEA, the fitness of an individual is based on its interactions with other agents. Thus, assessing the fitness of each agent is context-sensitive and subjective [34]. In *competitive* coevolution, individuals benefit when other agents fail. In *cooperative* coevolution, individuals succeed or fail as a team. This paper is focused on Cooperative Coevolutionary Algorithms (CCEAs), described in the following paragraph.

CCEAs are a natural approach in domains where agents succeed or fail as a team [37]. In CCEAs, distinct populations evolve simultaneously, and agents from these populations collaborate to reach good system solutions. One issue with CCEAs is that they tend to favor stable solutions, rather than optimal solutions [35]. This phenomena occurs because the different evolving populations adapt to each other, rather than adapting to form an optimal policy. Another issue that arises with CCEAs is the problem of credit assignment. Since the agents succeed or fail as a team, it becomes difficult to assign fitness to each individual agent in the system, as it is unclear how one particular agent affected the overall system performance. The credit assignment problem is typically addressed with fitness function shaping, such as use of local, global, or difference evaluation functions described in the previous section.

Cooperative Coevolutionary Algorithms are a multiagent implementation of Evolutionary Algorithms. In the case of CCEAs, each agent maintains a population of policies independent of each of the others, and runs an EA in parallel. The only change is the evaluation step, in which each agent is evaluated

based on how well the team performs. Thus, agents that not only achieve highly themselves, but also encourage other agents to achieve highly receive a high evaluation and are likely to move on to the next generation.

However, it is known that CCEAs tend toward stable, rather than optimal solutions [35]. Because each agent's fitness is a function of the agent with which it is paired, this evaluation is context-dependent. Agents that cooperate reasonably well with a wide variety of agents are highly likely to survive no matter which agents in the other populations they are paired with, encouraging stability. An agent that, with certain collaborators, actually forms the optimal solution to the problem might not survive if its actions were not compatible with a wide variety of other collaborators.

2.2.3 Neural Network Control

Neural networks are valuable tools for storing control policies, for three key reasons. First, neural networks have the ability to approximate any function to arbitrary accuracy, meaning they have the representational ability to encode any control policy [23]. Second, neural networks interpolate between data points well, which is useful in continuous state domains when novel states are encountered [24]. Finally, neural network controllers are capable of encoding effective control policies that only require a coarse representation of the system state [24]. In this work we use neural networks as policies to map inputs to outputs. They are useful for a wide variety of applications including:pattern recognition [3], fingerprint classification[44], robotic control [15], and multiagent systems [7, 22].

A neural network controller typically takes the current state (as measured by sensors) as an input, and returns an action to execute as its output. Training a neural network involves setting network weights to encode a control policy which returns actions that maximize a particular utility function. In this work neural networks are trained using CCEAs, where the fitness functions used to evaluate networks are either difference or global evaluation functions. In general, the specific evaluation function used to assign fitness to coevolving controllers can dramatically impact converged system performance.

2.3 Classic Continuous Rover Domain

In this work we use a modified version of the Continuous Rover Problem (CRP). The original domain [2] consists of a team of rovers that strive to observe a set of previously-defined POIs on a two-dimensional Euclidian plane. A POI p has a defined, static location L_p and a static scalar value V_p. The value of the observed information from a particular POI is a function of the value V_p and the inverse-square Euclidian distance at which it is being observed (δ), which is bounded by a minimum observation distance d_{min}:

$$\delta(x, y) = min(||x^2 - y^2||, d_{min}) \tag{3}$$

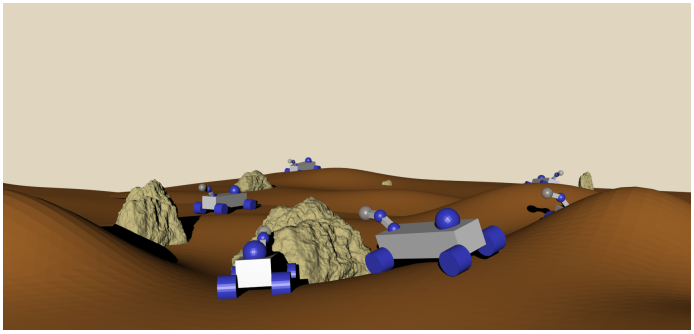

Figure 1: **A team of rovers exploring an unknown extraterrestrial environment.**

where x and y are two locations on the Euclidian plane. The global value that the team of rover seeks to maximize is calculated as the sum of the observations of all POIs:

$$G = \sum_{p} \frac{V_p \mathcal{O}_p}{\min_i \delta(L_p, L_i)} , \tag{4}$$

where V_p is the value of POI p, $\mathcal{O}_p$ returns 1 if any rover is within d_{max} and 0 otherwise, L_p is the location of POI p and L_i is the location of rover i.

Rover Sensor Model The rovers have a total of 8 sensory inputs which they map to two outputs which represent the distance they will move forward, and left/right. These 8 sensory inputs consist of 4 "rover" sensors and 4 "POI" sensors, each of which sense the relative distance-dependent density of objects within a particular quadrant, which rotates with the rover as the rover moves through the space (See Figure 2).

The POI sensor for quadrant q returns the function:

$$s_{1,q,i} = \sum_{p \in I_q} \frac{V_p}{\delta(L_p, L_i)} \tag{5}$$

where I_q is the set of observable POIs in the quadrant. The second sensor returns the sum of square distances from a rover to all the other rovers in the quadrant:

$$s_{2,q,i} = \sum_{i' \in N_q} \frac{1}{\delta(L_{i'}, L_i)} \tag{6}$$

where N_q is the set of rovers in quadrant q. The eight sensors provide the rover with a representation of their world based on the locations of POIs and other rovers. Note that this is an coarse representation of the world, as the location and number of rovers and POIs in each quadrant is reduced to an average number. Analyzing Equations 5 and 6, we see that the state variables are large if either (i) there are many POIs/rovers in a quadrant, or (ii) POIs/rovers in a

quadrant are close to the agent. This coarse representation is chosen in order to maintain a constant dimension for the state space; regardless of the number of POIs or rovers in the system, an agent's state is comprised of 8 values. This allows for the number of POIs or rovers to be easily scaled without affecting the learning algorithm.

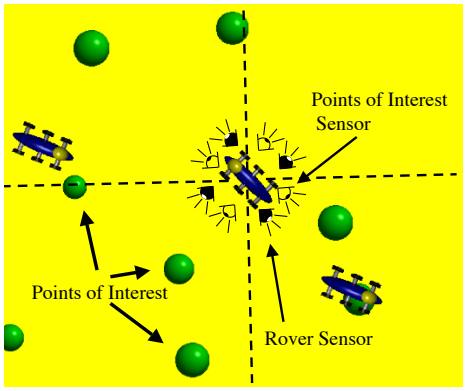

Figure 2: **Diagram of a Rover's Sensor Inputs.** The world is broken up into four quadrants relative to rover's position. In each quadrant one sensor senses points of interests, while the other sensor senses other rovers.

Rover Motion Model The two outputs of the neural network, $\{dx, dy\}$, indicate how far left/right and forward/backward the rover should move (Figure 2). This represents a local waypoint that the rover will navigate toward, which in practice can then be passed to a low-level navigation algorithm that will find a safe route to the waypoint [19].

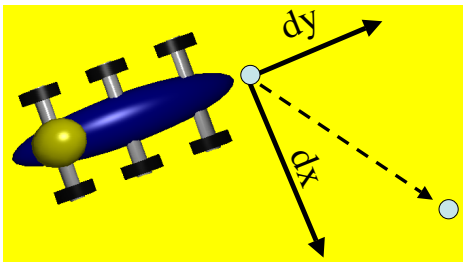

Figure 3: **Diagram of a Rover's Motion Model.** The rovers outputs indicate how far forward and to the side it will move. The rover's orientation for the next time step is in the direction of the vector $\{dx, dy\}$

Rover Evaluation Functions Here, we discuss the incorporation of the credit assignment schemes discussed in Section 2.2.1. Previous work has shown

that the local evaluation leads to extremely poor performance (in both this domain and other domains of various sorts), so in this work we omit consideration of the local evaluation.

To train the team of rovers using the global evaluation, every rover participating on a team directly receives Equation 14 as its evaluation. To train the team of rovers using the difference evaluation (in experiments that are completely independent from those where the team is trained using the global evaluation) we directly apply Equation 1 to Equation 14, resulting in:

$$D_i = \sum_p \frac{V_p \mathcal{O}_p}{min_i \delta(L_p, L_i)} - \frac{V_p \mathcal{O}_p}{min_{i' \neq i}(L_p, L_{i'})} \tag{7}$$

Note that each portion of D_i is only non-zero in the case where rover i collected the closest observation of POI p. The second term of the D_i is equal to the value of all the information collected if rover i were not in the system.

Shortcomings This implementation of the continuous rover domain is very valuable as a testing ground for multiagent coordination: to achieve highly, a team of agents has to either explicitly or implicitly coordinate such that their actions better the team as a whole, rather than some individual concerns. However, as a domain for developing algorithms that could be actually implemented, there are a number of shortcomings. The pattern in these shortcomings is that too much information is available to the rover team.

In actual space exploration, the environment is very uncertain, and communication is very limited [33]. In addition, mission priorities are often not set in stone before the mission begins, resulting in dynamically changing mission objectives. To address these shortcomings, in this work we introduce the novel Human-in-the-loop Limited Continuous Rover Domain (HLCRD).

3 Human-in-the-loop Limited Continuous Rover Domain

The HLCRD functions on the same basic mechanisms and dynamics of the CRD described above, but adds a few mechanics to the system, including:

(i) Sensor noise

(ii) Actuator noise

(iii) Limited sensor distance

(iv) Heterogeneous scout/payload rovers

We discuss each of these in turn. Sensor noise (i) is implemented by adding a random number drawn from a normal distribution to the sensor reading:

$$\forall n \; \forall q \; \forall i : \quad s\{n, q, i\} = \mathcal{N}\left(s\{n, q, i\}, \frac{s_{max}}{10}\right) \tag{8}$$

where $\mathcal{N}(\mu, \sigma)$ returns a random number drawn from a normal distribution with mean μ and standard deviation σ. In this case, sensor noise has a standard deviation equivalent to 10% of the maximum sensing distance s_{max}. In other words, sensor noise has a 68% probability of being 10% or less and a 95% probability of being 20% or less.

Actuator noise (ii) is implemented by adding a random number drawn from a normal distribution to each rover's chosen motion vector:

$$\forall i\{dx_i, dy_i\} = \{dx_i + \mathcal{N}(dx_i, \frac{dx_{max}}{10}), dy_i + \mathcal{N}(dy_i, \frac{dy_{max}}{10})\} \tag{9}$$

Similar to sensor noise, actuator noise is drawn from a normal distribution with a standard deviation equal to 10% of the maximum distance that a rover can travel in one time step.

The limited sensor range (iii) means that at a distance s_{max}, the rovers' sensors fail to register readings. This means that each of the sensors is calculated with an indicator function $S(i, p)$, which indicates whether a payload rover or POI is within sensory range ($S(i, p) \leftarrow 1$) or not ($S(i, p) \leftarrow 0$):

$$s_{1,q,i} = \sum_{p \in I_q} S(i, p) \frac{V_p}{\delta(L_p, L_i)} \tag{10}$$

$$s_{2,q,i} = \sum_{i' \in N_q} S(i, i') \frac{1}{\delta(L_{i'}, L_i)} \tag{11}$$

The heterogeneous scout/payload rover implementation (iv) bears additional explanation. We assume that an initial set of POIs exists that have been identified from initial orbital surveillance. The pre-identified POIs are marked with an $F_{j,t}$ value that is always set as 1. The other POIs are marked with an $F_{j,t}$ value that is set to 0 until the scout rovers pass close to the POI, send that imagery back to Earth, and the team receives a response from the Earth-based human team, at which point the $F_{j,t}$ value for that POI is changed to 1. We assume that communication (inbound or outbound) can happen every $C = 10$ time steps.

To clarify, we consider 3 of POIs, p_a, p_b, and p_c. p_a was pre-identified as a POI, and functions entirely like a POI for the entirety of the simulation, with $F_{b,t \geq 0} = 1$. If the communication time step $C = 4$, and a scout passes within the discovery distance of p_b at time step 3, then the imagery is sent to Earth on time step 4. At time step 8, $F_{b,t \geq 8} = 1$. In parallel, if a scout passes within the discovery distance of p_c at time step 5, the imagery is sent to Earth on time step 8, and subsequently returned at time step 12. Thus, $F_{c,t \geq 12} = 1$. These "found" indicators work into the domain by multiplying each sensor reading and POI value by the $F_{p,t}$ as follows. For the sensors:

$$s_{1,q,i} = \sum_{p \in I_q} F_{p,t} S(i,p) \frac{V_p}{\delta(L_p, L_i)} \tag{12}$$

$$s_{2,q,i} = \sum_{i' \in N_q} F_{p,t} S(i,i') \frac{1}{\delta(L_{i'}, L_i)} \tag{13}$$

And for the global evaluation:

$$G = \sum_p \frac{V_p \mathcal{O}_p F_{p,t}}{\min_i \delta(L_p, L_i)} \ , \tag{14}$$

And in turn the difference evaluation:

$$D_i = \sum_p F_{p,t} \left(\frac{V_p \mathcal{O}_p}{min_i \delta(L_p, L_i)} - \frac{V_p \mathcal{O}_p}{min_{i' \neq i}(L_p, L_{i'})} \right) \tag{15}$$

Note that each simulation is conducted independently, and a POI that is discovered early in one simulation might not be discovered until much alter in another simulation, or may never be explored at all. In this way the payload rovers must be develop very generalizable policies that function highly with the information available to them, and the performance of the scout rovers bounds the performance of the team as a whole.

We now turn our attention to the scout rovers, which do not function based on the sensory information or evaluations outlined to this point, but instead are concerned with exploring as much new terrain as possible. In order to encourage the scouts to explore new terrain, we place a series of fictional "beacons" in an evenly spaced Cartesian grid across the surface. Each of these beacons remains active ($v_{beacon} = 1$) until a scout passes within d_{beacon} distance of it, at which it point it is considered deactivated ($v_{beacon} = 0$). This implies that the scouts do have short-range communication ability and a shared database of which beacons have and have not been deactivated. The process of deactivating beacons after they are closely observed encourages the scouts to constantly seek out unexplored areas of the domain. Scouts discover future POIs by passing within $d_{discovery}$ distance of the POI location (and subsequent sending of the location and preliminary sensory data back to Earth based on the communication cycle described previously). The scouts can sense the beacons from a longer distance ($d_{be\_sensory} = 2 \cdot d_{discovery}$), so that their sensors represent the relative density of active beacons that could be deactivated (new territory that could be explored) by traveling toward each quadrant. The scouts can sense other scout rovers from an even longer distance ($d_{sc\_sensory} = 4 \cdot d_{discovery}$), to reduce the chance of redundant exploration.

The scout's sensor representation is formulated similarly to the payload rover representation, with each quadrant having a beacon sensor and a scout sensor:

$$s_{beacon,q,i} = \sum_{b \in I_q} S(i,b) \frac{v_{beacon}}{\delta(L_b, L_i)} \tag{16}$$

$$s_{scout,q,i} = \sum_{i' \in N_q} S(i,i') \frac{1}{\delta(L_{i'}, L_i)} \tag{17}$$

where the beacon sensor $s_{beacon,q,i}$ identifies all active ($v_{beacon} = 1$) beacons b in quadrant I_q within $d_{sensory}$ distance of rover i (so that $S(i,b) = 1$). The scout sensor, $s_{scout,q,i}$ identifies all scout rovers within sensory range of rover i ($S(i,i') = 1$ iff $\delta(i,i') < d_{sc\_sensory}$) to provide a distance-weighted value per quadrant. It is of note that the state variables of scout and payload rovers are not affected by each other. In other words, when computing the rover value for a particular quadrant on a scout rover, sensed payload rovers are not incorporated into this computation. Similarly, sensed scout rovers are not incorporated into payload rover state variable computations.

The global evaluation that the scout rovers attempt to maximize is the number of beacons that have been deactivated:

$$G_{scout} = \sum_b (1 - v_{beacon,b}) \tag{18}$$

When we apply Equation 1 directly to Equation 18, we arrive at:

$$D_{scout,i} = \sum_b (1 - v_{beacon,b}) - \sum_b (1 - v_{beacon,b,-i}) \tag{19}$$

Where $v_{beacon,b,-i}$ is the value that $v_{beacon,b}$ would take if agent i did not contribute to the system. Note, then, that any beacon which 2 or more scouts would have deactivated gives zero contribution to each agent's difference evaluation, and only the beacons that would have been uniquely deactivated by one agent contribute to that agent's difference evaluation.

4 Proposed Algorithm

We now describe the CCEA algorithm used in this work, including detailing the general algorithm, the population initialization operator, the mutation operator, and the selection operator. In the CCEA algorithm, N coevolving populations of neural networks are utilized to form teams comprised of M agents. For this work, M and N are equal, meaning that each coevolving population is responsible for developing a policy for one of the agents in the system. Each population is initially comprised of k neural networks. The networks are randomly initialized by drawing weights from the normal distribution $\mathcal{N}(0, \sigma_i)$, where $\sigma_i = 1.0$ is the standard deviation used for weight initialization in this work.

For each generation, k successor networks are generated in each population, doubling the population size. For every network in the population, a mutated

Algorithm 1 Standard CCEA

```
Initialize N populations of k neural networks
for each Generation do
    for each Population do
        produce k successor solutions
        mutate successor solutions
    end for
    for i = 1 → 2k do
        randomly select one policy from each population
        create team T_i of agents with selected policies
        simulate T_i in domain
        assign fitness to each agent in T_i using Ψ(z)
    end for
    for each Population do
        select k networks using binary tournament
    end for
end for
```

copy is created by randomly selecting 10% of the weights to be mutated. These selected weights are mutated by adding a value drawn from the normal distribution $\mathcal{N}(0, \sigma_m)$, where $\sigma_m = 0.25$ is the standard deviation used for mutation in this work.

After mutation, $2k$ teams of N agents are formed by randomly selecting agents from each population and placing these agents on a team. Each agent in a population is selected only once. The performance of each team is then evaluated in the domain, and the fitness of every agent on the team is set based on a fitness assignment operator $\Psi(z)$. For the purposes of this work, $\Psi(z)$ is either the global evaluation function or the difference evaluation function. After being assigned fitness values, each agent in the team is placed back into its respective population.

Once each team has been evaluated and every agent in each population has a fitness value, agents are selected to survive to the next generation. Selection is conducted using a binary tournament operator for each population [17]. Two agents are randomly drawn from a population at a time, and the agent with the higher fitness value is chosen to survive, while the agent with the lower fitness value is discarded. After the selection stage, the population are of size k, and the next generation begins by returning to the mutation step. This process is repeated for a set number of generations. The CCEA used in this work is detailed in Algorithm 1.

For the HLCRD, two separate CCEAs are utilized. The first CCEA evolves control policies for the payload rovers. In this case, the fitness values are assigned based on Equation 14. The second CCEA evolves control policies for the scout rovers. In this case, the fitness values are assigned based on Equation 18. These two CCEAs are conducted in parallel and the performance of the payload rovers

is implicitly coupled with the performance of the scout rovers: if potential POIs are never discovered by a scout and subsequently identified as a POI by the human ground team on Earth, the payload rovers will never be able to observe those POIs, which directly leads to degraded system performance. There does not exist such a link in the other direction; the performance of the scout rovers is not affected by the performance of the payload rovers.

5 Experimental Results

We present here the experimental results for the following four experiments:

(i) Scout rovers' coverage of the unknown terrain (§ 5.1);

(ii) Scout/payload performance with 10% domain observability and no sensor/actuator noise (§ 5.2);

(iii) Scout/payload performance with 10% domain observability and 10% sensor/actuator noise (§ 5.3);

(iv) Scout/payload performance with 10% domain observability, 10% sensor/actuator noise, and 10% agent failures (§ 5.4)

For each experiment, we examine teams of 10, 50, and 100 rovers attempting to observe 20, 100, and 200 POIs, respectively. The rovers act in a 100x100 grid, where there are equal numbers of scout and payload rovers; for example, the 10 rover case has a team of 10 scout rovers and 10 payload rovers. The simulation runs for 100 time steps, with Earth-communication available every 10 time steps. In all cases, POIs are assigned equal value, and their total values sum to 50. All rovers (scouts and payload rovers) are initialized near the center of the domain, with their (x, y) locations being drawn from $(\mathcal{N}(50, 10), \mathcal{N}(50, 10))$, assuming the world dimensions are in the bounds $(x \in [0, 100], y \in [0, 100])$. For the scout rovers, beacons are arranged uniformly in the 100x100 grid, being located at the center of each grid cell.

Initially, 10% of the POIs are known to the rovers, and scouts attempt to explore new areas to increase the number of POIs discovered. Every 10 time steps, information from the scout rovers is transmitted to earth, and POI locations found from this data are sent back to the payload rovers. Once these new POIs are discovered, payload rovers must go to these POIs and observe them.

5.1 Scout Rover Results

Figure 4 shows the coverage of the domain (percent of fictitious beacons successfully deactivated by the scouts) generated by the scout rovers as a function of generation. In all results plots to follow, error bars report error in the mean; in many cases, error bars are smaller than the plot markers and cannot be seen. As might be expected, more scout rovers are more capable of exploring

the unknown area, and identifying POIs for the payload rovers more quickly. The scout rovers trained with the difference evaluation perform significantly better than those trained with the global evaluation function, covering at 9% to 16% more of the domain depending on the number of agents in the system. This demonstrates that difference evaluation functions are capable of improving coordination between agents.

For the 100 agent case, scouts trained with the difference evaluation function achieve 97% coverage on average, with 21% of the statistical trials achieving optimal performance. This suggests that multiagent surveillance tasks can be achieved in a fully autonomous manner, requiring minimal input from expert system designers.

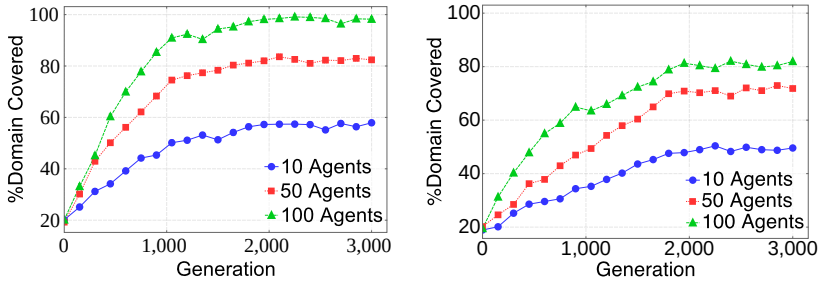

Figure 4: **Scout rover coverage as a function of generation for 10, 50, and 100 rovers; difference evaluations left, global evaluations right**

5.2 Limited Observations

The second experiment involves the ideal case of rovers having limited observability (sensors can see a distance equal to 10% of the world length), but having no sensor or actuator noise. This experiment gives us a performance baseline to analyze how the addition of noise or agent failures affects system performance.

Figure 5 shows the global performance attained by teams of 10, 50, and 100 rovers in turn. It also shows the converged performance as a function of team size. In all three of these cases, agents trained using difference evaluations achieve higher performance than agents using global evaluation functions. This is due to the fact that agent-specific feedback based on difference evaluations during learning allow agents to more easily discern the effects of their actions, allowing them to more easily optimize their local feedback signals in order to improve overall system performance. In the 100 agent case, agents using difference evaluations reach 91% of the theoretically optimal value, while agents using global evaluations only reach 68% of the theoretical optimal value.

Of particular interest is the scaling plot in Figure 5. The performance of 10 agents using difference evaluations is almost equivalent to the performance of 100 agents using global evaluations. Thus, an improved fitness assignment operator reduces the hardware requirements tenfold for similar performance. This is

an important finding, because the cost of sending hardware on extraterrestrial missions is extremely high, meaning that attaining a similar performance level with less hardware results in significant monetary savings.

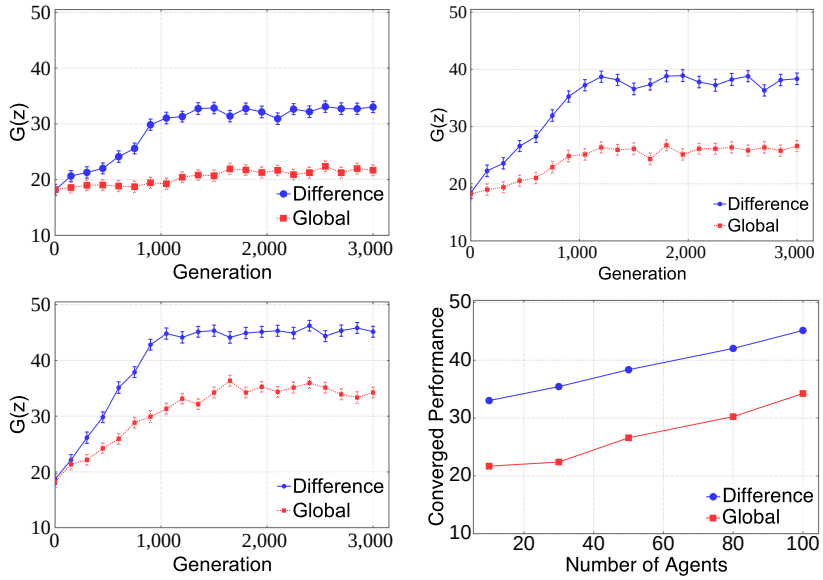

Figure 5: **Global team performance under limited observation radius, as a function of generation for 10 (top left), 50 (top right), and 100 (bottom left) rovers. In all cases teams trained with difference evaluations outperform teams trained with the global evaluation. Bottom right shows converged system performance as a function of team size.**

5.3 Limited Observation, Sensor/Actuator Noise

The third experiment involves the case of rovers having limited observability (equivalent to experiment in § 5.2), but also having 10% sensor and actuator noise. In real-world settings, sensors and actuators always have noise, and it is important to determine the effects of noise on system performance. Figure 6 shows the global performance achieved by teams of rovers in a limited observation environment with sensor and actuator noise, for 10, 50, and 100 agents. It also shows the converged performance as a function of the number of agents in the system.

As in the results from § 5.2, agents using difference evaluations significantly outperform agents using global evaluations. Interestingly, the addition of 10% noise had no statistically significant impact on system performance. This is due to two key reasons. First, the coarse state representation which reduces quadrant information to single values is inherently robust to noise, because of the

loss of resolution when compactly summarizing the state of a quadrant. Second, neural networks are excellent at rejecting noise in input signals, because small changes in inputs do not strongly affect their outputs. This property allows neural networks to encode extremely robust control policies. This is an encouraging result, because noise always exists in sensors and actuators, and could potentially become worse in extreme environments found in extraterrestrial exploration. A control strategy which is robust to this noise is therefore critical.

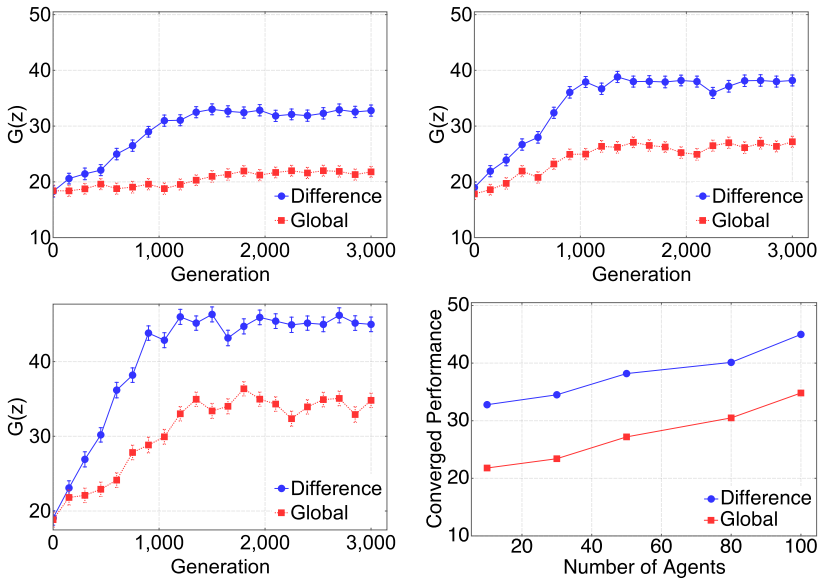

Figure 6: **Global performance in the presence of limited observation, sensor noise, and actuator noise as a function of generation for 10 (top left), 50 (top right), and 100 (bottom left) rovers. Bottom right shows converged performance as a function of the number of agents in the system. The addition of 10% noise does not alter performance in a statistically significant manner compared to the noise-free experiments.**

5.4 Limited Observation, Noise, Rover Failure

The final experiment involves the case with 10% observability, 10% sensor and actuator noise, and agent failures. After 3000 generations, 10% of the agents fail. When a rover fails, it is unable to move, sense, or observe nearby POIs. However, it can still be sensed by other rovers. This requires the remaining rovers to account for the presence of the failed rovers in their policies, and adds to the complexity of the problem: if a rover A senses another rover B in the scenarios without failure, A may be able to assume that B will be observing

POIs close to its current location. Once the failures occur, this assumption is no longer valid and the rovers must adapt their policies accordingly. This experiment demonstrates a real-world scenario, in which sensors have limited sensing ranges, sensors and actuators are noisy, and some amount of hardware will fail.

Results for the 10, 50, and 100 rover cases are shown in Figure 7. This figure also shows the converged performance as a function of the number of agents in the system. The distributed multi-rover setup leads to the team, no matter whether it was trained on the global or difference evaluation, experiencing only a degradation of system performance, instead of the complete end of the mission, which might happen in the case of a monolithic rover. Notice that after the failures, the team does continue adapting to the absence of the failed rovers, and recovers some system performance by reconfiguring (this is shown most clearly in the bottom left figure of the 100 agent team).

Agents using difference evaluations outperform agents using global evaluations, regardless of the system size. As in the previous experiments, the 10 agent system using difference evaluations performs almost identically to the 100 agent system using global evaluations. This demonstrates the mission cost savings attainable by using difference evaluation functions, because 10 times less hardware is needed for similar performance.

A particularly interesting result is the robustness to noise demonstrated by difference evaluations. For the 100 agent case, 10% agent failures resulted in a 5.8% decrease in overall system performance. For agents using global evaluations, 10% agent failures led to a 9.1% reduction in system performance. Thus, difference evaluations not only result in higher overall system performance, but they also are more robust to agent failures within the system.

6 Discussion and Conclusion

Extraterrestrial exploration missions are exceedingly difficult due to the variety of challenges faced in space. In particular, these missions require operations in uncertain and extreme environments. Exploring unknown environments is slowed considerably by light-time delays, as exploring robots often can't make control decisions until they communicate with human scientists on Earth, which can take hours or days. Further, as the monetary cost of such missions is extremely high, reliability and robustness of the exploring system is paramount.

Artificial intelligence techniques can be used to develop autonomous systems which address many of these difficulties. In particular, learning-based multiagent techniques can be used to develop a team of multiple autonomous coordinating agents. Agent-specific control policies can be developed such that agents can autonomously act in uncertain environments, while coordinating with other agents in the system. By adding autonomy to the system, uncertainty in the environment can be addressed without mission downtime being caused by light-time delays and waiting for direction by scientists on Earth. However, full autonomy (free of any human intervention) is not always desirable. For

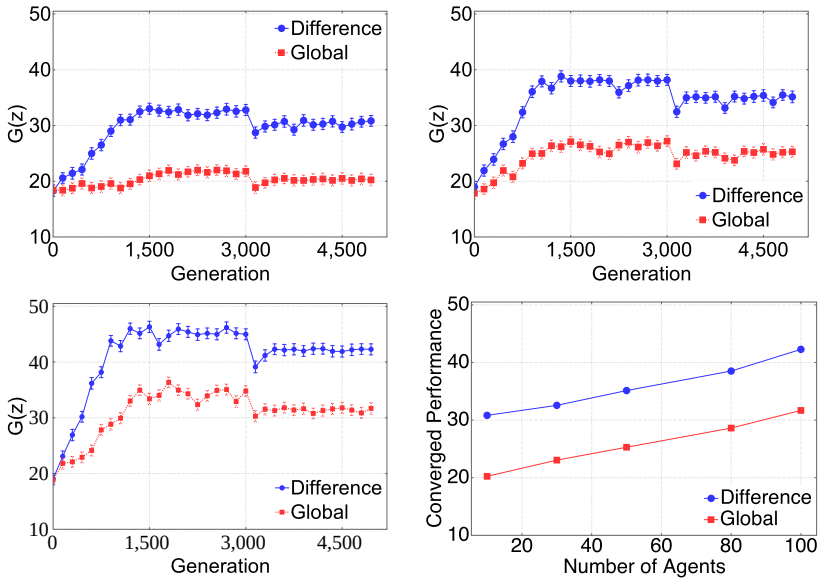

Figure 7: **Global performance in the presence of limited observation, sensor noise, and actuator noise as a function of generation for 10 (top left), 50 (top right), and 100 (bottom left) rovers for teams trained on the global (red) and difference (blue) evaluation. Bottom right plot demonstrates converged performance as a function of the number of agents in the system.**

example, autonomous agents which can make low-level decisions in real time are desirable, but high-level tasks should still be capable of being altered by human scientists on Earth. Further, multiagent exploration missions have no single point of failure. If a single robot in a team of robots fail, the science return is not all lost. This inherent robustness of multiagent systems is extremely desirable for costly space exploration missions.

In this work, we present a novel human-in-the-loop cooperative coevolutionary algorithm for a team of rovers exploring an unknown environment, using agent-specific fitness assignment operators known as difference evaluation functions. Low-level actions such as task distribution between agents and navigation are handled autonomously by agents in the system. However, high-level tasks such as mission goals can be altered by human scientists on Earth. This allows for light-time delays to be minimized, and only affect the system when changes to high-level tasks must specified.

Our results demonstrate that the use of our algorithm dramatically outperforms traditional cooperative coevolutionary algorithms. In fact, 10 agent systems trained with our algorithm perform similarly to 100 agent systems trained with traditional algorithms. This result leads to two possibilities. First, less hardware can be sent on an extraterrestrial mission while achieving the same

performance, dramatically reducing mission costs. Second, the same amount of hardware can be sent on the mission, resulting in superior science return.

In addition to increasing the science return of exploration missions, our algorithm results in control policies which are extremely robust to sensor and actuator noise as well as agent failures, properties which are critical in extreme and uncertain environments. The addition of 10% sensor and actuator noise results in system performance which is not different from a noise free system in a statistically significant manner. When 10% of the agents in the system fail, the science return is only reduced by 5.8%. These results demonstrate the capability of this algorithm to develop autonomous control policies for multiagent extraterrestrial exploration robots which are robust to sensor and actuator noise, can perform low-level tasks autonomously, and require minimal intervention from human scientists on Earth while maximizing the science return of the mission.

References

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pages 1–12, Seattle, WA, 2004.

[2] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *JAAMAS*, 17(2):320–338, 2008.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[4] M. Brambilla, C. Pinciroli, M. Birattari, and M. Dorigo. Property-driven design for swarm robotics. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 139–146, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[5] R. A. Brooks. A robust layered control system for a mobile robot. AI Memo 864, Massachusetts Institute of Technology, September 1985.

[6] A. Bucci and J.B. Pollack. On identifying global optima in cooperative coevolution. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 539–544, New York, NY, USA, 2005. ACM.

[7] L. Cheng, Z-G. Hou, M. Tan, Y. Lin, and W. Zhang. Neural-network-based adaptive leader-following control for multiagent systems with uncertainties. *Neural Networks, IEEE Transactions on*, 21(8):1351–1358, 2010.

[8] S. Chien, A. Barrett, T. Estlin, and G. Rabideau. A comparison of coordinated planning methods for cooperating rovers. *International Conference on Autonomous Agents*, 2000.

[9] S. Chien and Joshua Doubleday et al. Combining space-based and in-situ measurements to track flooding in thailand. *Geoscience and Remote Sensing*, 2011.

[10] M. Colby and K. Tumer. Multiagent reinforcement learning in a distributed sensor network with indirect feedback. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 941–948, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[11] P. Dasgupta, V. Ufimtsev, C. Nelson, and S. Hossain. Dynamic reconfiguration in modular robots using graph partitioning-based coalitions. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 121–128, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[12] T. Estlin and S. Chien et al. Coordinating multiple spacecraft in joint science campaigns. *i-SAIRAS*, August 2010.

[13] F. Fang, A. Jiang, and M. Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 957–964, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[14] S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic and dynamical-systems analysis of selection methods in coevolution, 2005.

[15] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In *Proc. of Conf. on Simulation of Adaptive Behavior*, 1994.

[16] D.B. Fogel. An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3 –14, jan 1994.

[17] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801–2996, 1991.

[18] F. Gomez and R. Miikkulainen. Active guidance for a finless rocket through neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, Illinois, 2003.

[19] K. Z. Haigh and M. M. Veloso. High-level planning and low-level execution: Towards a complete robotic agent. In *First International Conference on Autonomous Agents*, February 1997.

[20] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls. Multi-robot collision avoidance with localization uncertainty. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems -*

Volume 1, AAMAS '12, pages 147–154, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[21] F. Hoffmann, T.-J. Koo, and O. Shakernia. Evolutionary design of a helicopter autopilot. In *Advances in Soft Computing - Engineering Design and Manufacturing, Part 3: Intelligent Control*, pages 201–214, 1999.

[22] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings National Academy of Science*, 79:2554–2558, April 1982.

[23] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(3), 1989.

[24] P. Huang, J. Lehman, A.K. 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*, pages 1–8, Orlando, FL, 2014.

[25] M. Keidar and G. Kaminka. Robot exploration with fast frontier detection: Theory and experiments. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 113–120, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[26] K. Low, J. Chen, J. Dolan, S. Chien, and D. Thompson. Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 105–112, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[27] S. Mamidi, Y. Chang, and R. Maheswaran. Improving building energy efficiency with a network of sensing, learning and prediction agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 45–52, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[28] N. Mathews, A. Stranieri, A. Scheidler, and M. Dorigo. Supervised morphogenesis: Morphology control of ground-based self-assembling robots by aerial robots. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 97–104, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[29] A. Mishkin, D. Limonadi, S. Laubach, and D. Bass. Working the martian night shift-the mer surface operations process. *Robotics & Automation Magazine, IEEE*, 13(2):46–53, 2006.

[30] P. Natarajan, T. Hoang, Kian H. Low, and M. Kankanhalli. Decision-theoretic approach to maximizing observation of multiple targets in multi-camera surveillance. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 155–162, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[31] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Proc. of Artificial Life IV*, pages 190–197, 1994.

[32] Alexandros P., N. Spanoudakis, and M. Lagoudakis. Model-driven behavior specification for robotic teams. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 171–178, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[33] A. D. Panagopoulos and J. D. Kanellopoulos. Prediction of triple-orbital diversity performance in earth-space communication. *International Journal of Satellite Communications*, 20(3):187–200, 2002.

[34] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[35] Liviu Panait, Karl Tuyls, and Sean Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *The Journal of Machine Learning Research*, 9:423–457, 2008.

[36] J. Patrix, A. Mouaddib, S. Le Gloannec, D. Stampouli, and M. Contat. Discrete relative states to learn and recognize goals-based behaviors of groups. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 933–940, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[37] M. A. Potter and K. A. De Jong. Evolving neural networks with collaborative species. *Computer Simulation Conference*, 1995.

[38] T. O. Ting, K. Wan, K. L. Man, and S. Lee. Space exploration of multi-agent robotics via genetic algorithm. In *Network and Parallel Computing*, pages 500–507. Springer, 2012.

[39] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff. Autonomous and autonomic systems: a paradigm for future space exploration missions. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36(3):279–291, 2006.

[40] K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 378–385, Bologna, Italy, July 2002.

[41] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[42] G. Webster. Curiosity team switches back to earth time, November 2012.

[43] D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), March 2000.

[44] Y. Yao, G. Marcialis, M. Pontil, P. Frasconi, and F. Roli. Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines. *Pattern Recognition*, 36(2):397–406, 2003.

[45] Y. C. Zhang. Modeling market mechanism with evolutionary games. *Europhysics Letters*, March/April 1998.