Efficient Reward Functions for Adaptive Multi-Rover Systems

Kagan Tumer[1] and Adrian Agogino[2]

[1] NASA Ames Research Center
Mail Stop 269-4
Moffet Field, CA 94035
Phone:650-604-4940
Fax:650-604-4036
ktumer@mail.arc.nasa.gov

[2] UC Santa Cruz, NASA Ames Research Center
Mail Stop 269-3
Moffet Field, CA 94035
Phone:650-604-5985
Fax:650-604-4036
adrian@email.arc.nasa.gov

Abstract. This paper addresses how efficient reward methods can be applied to multiple agents co-evolving in noisy and changing environments, under communication limitations. This problem is approached by "factoring" a global reward over all agents into agent-specific rewards that have two key properties: 1) agents maximizing their agent-specific rewards will tend to maximize the global reward, 2) an agent's action has a large influence over its agent-specific reward allowing it to evolve quickly. Agents using these agent-specific rewards are tested in episodic and non-episodic, continuous-space multi-rover environment where rovers evolve to maximize a global reward function over all rovers. The environments are dynamic (i.e. changes over time) and can be noisy and can restrict communication between agents . We show that a control policy evolved using these agent-specific rewards outperforms global reward methods by up to 400%. More notably, in the presence of a larger number of rovers or rovers with noisy and communication limited sensors, the proposed method outperforms global reward by a higher percentage than in noise-free conditions with a small number of rovers.

1 Introduction

Using machine learning techniques to control a large set of distributed agents is a difficult problem, especially when the environment is noisy and changing. Using a single-agent machine learning algorithm by treating the collection of agents as a single agent is highly problematic due to exponentially increasing state spaces. In addition many problems are fundamentally distributed, especially ones where agents have communication limitations. This paper shows a

method where successful single-agent machine learning algorithms can be extended to a "collective," a distributed control task where a large set of entities that collectively strive to maximize a global reward function [20, 16, 17]. In this paper we focus on a collective of data-gathering rovers whose task is to maximize the aggregate information collected by the full collective. In order to distinguish the members of the collective from the individuals in the population of an evolutionary algorithm, we will use "rovers" exclusively to refer to the members of a collective through this paper[3].

Our approach is based on giving each rover in the collective its own reward function, which it tries to maximize using a learning algorithm. The key issue in such an approach is to ensure that the rover reward function possesses the following two properties: (i) it is aligned with the global reward, ensuring that the rovers that maximize their own reward do not hinder one another and hurt the reward of the collective; and (ii) it is sensitive to the actions of the rover, ensuring that it provides the right selective pressure on the rover (i.e., it limits the impact of other rovers in the reward function).

Our domain has a number of properties that make it particularly difficult for learning algorithms:

1. The environment is dynamic, meaning that the conditions under which the rovers evolve change with time. The rovers need to evolve general control policies, rather than specific policies tuned to their current environment.
2. The rovers have restrictions on their sensing abilities, meaning that the information they have access to is limited. The rovers need to formulate policies that satisfy the global reward function based on limited, local information.
3. The number of rovers in the system can be larger. The rovers need to decouple the impact of other rovers from their reward functions.

This paper provides methods to learn control policies in dynamic environments for large collectives of rovers with limited communication capabilities. In Sections 2 and 3 we discuss the properties needed in a collective, how to evolve rovers using reward functions possessing such properties along with a discussion of related work. In Section 4 we present the "Rover Problem" where planetary rovers in a collective use neural networks to determine their movements based on a continuous-valued array of sensor inputs. Section 5 presents the performance of the rover collective evolved using rover reward functions in dynamic and communication limited domains. The results show the the effectiveness of the rovers in gathering information is 400% higher with properly derived rover reward functions than in rovers using a global reward function. Finally Section 6 we discuss the implication of these results and their applicability to different domains.

[3] Note, one can have individuals in a population of rovers or in a population of collectives, depending on where the evolutionary operators are applied.

2 Rover Reward Function Properties

Let us now derive effective rover reward functions based on the theory of collectives described in [20]. Let the **global reward function** be given by $G(z)$, where z is the state of the full system (e.g., the position of all the rovers in the system, along with their relevant internal parameters and the state of the environment). Let the **rover reward function** for rover i be given by $g_i(z)$. First we want the private reward functions of each agent to have high *factoredness* with respect to G, intuitively meaning that an action taken by an agent that improves its private reward function also improves the global reward function (i.e. G and g_η are aligned). Formally, the degree of factoredness between g_i and G is given by:

$$\mathcal{F}_{g_i} = \frac{\int_z \int_{z'} u[(g_i(z) - g_i(z'))\ (G(z) - G(z'))]dz'dz}{\int_z \int_{z'} dz'dz} \tag{1}$$

where z' is a state which only differs from z in the state of rover i, and $u[x]$ is the unit step function, equal to 1 when $x > 0$. Intuitively, a high degree of factoredness between g_i and G means that a rover evolved to maximize g_i will also maximize G.

Second, the rover reward function must be more sensitive to changes in that rover's actions than to changes in the actions of other rovers in the collective. Formally we can quantify the *rover-sensitivity* of reward function g_i, at z as:

$$\lambda_{i,g_i}(z) = E_{z'}\left[\frac{\|g_i(z) - g_i(z - z_i + z'_i)\|}{\|g_i(z) - g_i(z' - z'_i + z_i)\|}\right] \tag{2}$$

where $E_{z'}[\cdot]$ provides the expected value over possible values of z', and $(z - z_i + z'_i)$ notation specifies the state vector where the components of rover i have been removed from state z and replaced by the components of rover i from state z'. So at a given state z, the higher the rover-sensitivity, the more $g_i(z)$ depends on changes to the state of rover i, i.e., the better the associated signal-to-noise ratio for i is. Intuitively then, higher rover-sensitivity means there is "cleaner" (e.g., less noisy) selective pressure on rover i.

As an example, consider the case where the rover reward function of each rover is set to the global reward function, meaning that each rover is evaluated based on the actions of the full collective. Such a system will be factored by the definition of Equation 1. However, the rover reward functions will have low rover-sensitivity (the reward of each rover depends on the actions of all other rovers).

3 Difference Reward Functions

Let us now focus on improving the rover-sensitivity of the reward functions. To that end, consider **difference** reward functions [20], which are of the form:

$$D_i(z) \equiv G(z) - G(z_{-i} + c_i) \tag{3}$$

where z_{-i} contains all the states on which rover i has no effect, and c_i is a fixed vector. In other words, all the components of z that are affected by rover i are replaced with the fixed vector c_i. Such difference reward functions are fully factored no matter what the choice of c_i, because the second term does not depend on i's states [20] (e.g., D and G will have the same derivative with respect to z_i). Furthermore, they usually have far better rover-sensitivity than does a global reward function, because the second term of D removes some of the effect of other rovers (i.e., noise) from i's reward function. In many situations it is possible to use a c_i that is equivalent to taking rover i out of the system. Intuitively this causes the second term of the difference reward function to evaluate the value of the system without i and therefore D evaluates the rover's contribution to the global reward.

Though for linear reward functions D_i simply cancels out the effect of other rovers in computing rover i's reward function, its applicability is not restricted to such functions. In fact, it can be applied to any linear or non-linear global utility function. However, its effectiveness is dependent on the domain and the interaction among the rover reward functions. At best, it fully cancels the effect of all other rovers. At worst, it reduces to the global reward function, unable to remove any terms (e.g., when z_{-i} is empty, meaning that rover i effects all states). In most real world applications, it falls somewhere in between, and has been successfully used in many domains including rover coordination, satellite control, data routing, job scheduling and congestion games [3, 18, 20]. Also note that the computation of D_i is a "virtual" operation in that rover i computes the impact of its not being in the system. There is no need to re-evolve the system for each rover to compute its D_i, and computationally it is often easier to compute than the global reward function [18]. Indeed in the problem presented in this paper, for rover i, D_i is easier to compute than G is (see details in Section 5).

4 Continuous Rover Problem

In this section, we show how evolutionary computation with the difference reward function can be used effectively in the Rover Problem[4]. In this problem, there is a collective of rovers on a two dimensional plane, which is trying to observe points of interests (POIs). Each POI has a value associated with it and each observation of a POI yields an observation value inversely related to the distance the rover is from the POI. In this paper the distance metric will be the squared Euclidean norm, bounded by a minimum observation distance, δ_{min}:[5]

$$\delta(x, y) = \max\{\|x - y\|^2, \delta_{min}^2\} \; . \tag{4}$$

[4] This problem was first presented in [3].

[5] The square Euclidean norm is appropriate for many natural phenomenon, such as light and signal attenuation. However any other type of distance metric could also be used as required by the problem domain. The minimum distance is included to prevent singularities when a rover is very close to a POI.

The global reward function is given by:

$$G = \sum_t \sum_j \frac{V_j}{\min_i \delta(L_j, L_{i,t})} , \tag{5}$$

where V_j is the value of POI j, L_j is the location of POI j and $L_{i,t}$ is the location of rover i at time t. Intuitively, while any rover can observe any POI, as far as the global reward function is concerned, only the closest observation matters[6].

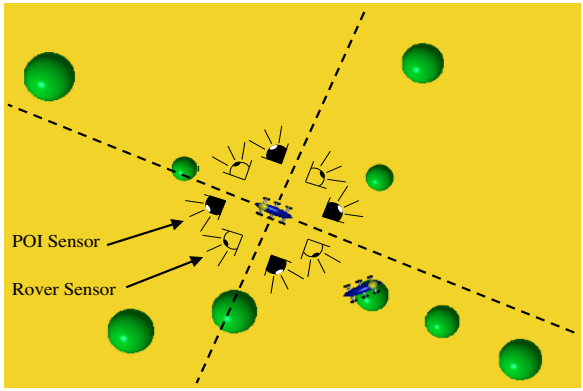

Fig. 1. Diagram of a Rover's Sensor Inputs. The world is broken up into four quadrants relative to rover's position. In each quadrant one sensor senses points of interest, while the other sensor senses other rovers.

4.1 Rover Capabilities

At every time step, the rovers sense the world through eight continuous sensors. From a rover's point of view, the world is divided up into four quadrants relative to the rover's orientation, with two sensors per quadrant (see Figure 1). For each quadrant, the first sensor returns a function of the POIs in the quadrant at time t. Specifically the first sensor for quadrant q returns the sum of the values of the POIs in its quadrant divided by their squared distance to the rover and scaled by the angle between the POI and the center of the quadrant:

$$s_{1,q,j,t} = \sum_{j \in J_q} \frac{V_j}{\delta(L_j, L_{i,t})} \left(1 - \frac{|\theta_{j,q}|}{90}\right) \tag{6}$$

where J_q is the set of observable POIs in quadrant q and $|\theta_{j,q}|$ is the magnitude of the angle between POI j and the center of the quadrant. The second sensor

[6] Similar reward functions could also be made where there are many different levels of information gain depending on the position of the rover. For example 3-D imaging may utilize different images of the same object, taken by two different rovers.

returns the sum of inverse square distances from a rover to all the other rovers in the quadrant at time t scaled by the angle:

$$s_{2,q,i,t} = \sum_{i' \in N_q} \frac{1}{\delta(L_{i'}, L_{i,t})} \left(1 - \frac{|\theta_{i',q}|}{90}\right) \tag{7}$$

where N_q is the set of rovers in quadrant q and $|\theta_{i',q}|$ is the magnitude of the angle between rover i' and the center of the quadrant.

The sensor space is broken down into four regions to facilitate the input-output mapping. There is a trade-off between the granularity of the regions and the dimensionality of the input space. In some domains the tradeoffs may be such that it is preferable to have more or fewer than four sensor regions. Also, even though this paper assumes that there are actually two sensors present in each region at all times, in real problems there may be only two sensors on the rover, and they do a sensor sweep at 90 degree increments at the beginning of every time step.

4.2 Rover Control Strategies

With four quadrants and two sensors per quadrant, there are a total of eight continuous inputs. This eight dimensional sensor vector constitutes the state space for a rover. At each time step the rover uses its state to compute a two dimensional output. This output represents the x, y movement relative to the rover's location and orientation.

The mapping from rover state to rover output is done through a Multi Layer Perceptron (MLP), with eight input units, ten hidden units and two output units. The MLP uses a sigmoid activation function, therefore the outputs are limited to the range $(0, 1)$. The actual rover motions dx and dy, are determined by normalizing and scaling the MLP output by the maximum distance the rover can move in one time step. More precisely, we have:

$$dx = d_{max}(o_1 - 0.5)$$
$$dy = d_{max}(o_2 - 0.5)$$

where d_{max} is the maximum distance the rover can move in one time step, o_1 is the value of the first output unit, and o_2 is the value of the second output unit.

4.3 Rover Policy Selection

The MLP for a rover is selected using an evolutionary algorithm. In this case, each rover has a population of MLPs. At each N time steps (N set to 15 in these experiments), the rover uses ϵ-greedy selection ($\epsilon = 0.1$) to determine which MLP it will use (e.g., it it selects the best MLP from its population with 90% probability and a random MLP from its population with 10% probability). The selected MLP is then mutated by adding a value sampled from the Cauchy Distribution (with scale parameter equal to 0.3) to each weight, and is used for

those N steps. At the end of those N steps, the MLP's performance is evaluated by the rover's reward function and re-inserted into its population of MLPs, at which time, the poorest performing member of the population is deleted. Both the global reward for system performance and rover reward for MLP selection is computed using an N-step window, meaning that the rovers only receive an reward after N steps.

While this is not a sophisticated evolutionary algorithm, it is ideal in this work since our purpose is to demonstrate the impact of principled reward functions selection on the performance of a collective. Even so, this algorithm has shown to be effective if the reward function used by the rovers is factored with G and has high rover-sensitivity. We expect more advanced algorithms from evolutionary computation, used in conjunction with these same reward functions, to improve the perform collective further.

4.4 Learning Control Strategies in a Collective

The key to success in this approach is to determine the correct rover reward functions. In this work we test three different reward function for rover selection. The first reward function is the global reward function (G), which when implemented results in approach two discussed in Section 2:

$$G = \sum_t \sum_j \frac{V_j}{\min_i \delta(L_j, L_{i,t})} \tag{8}$$

The second reward function is the "perfectly rover-sensitive" reward function (P):

$$P_i = \sum_t \sum_j \frac{V_j}{\delta(L_j, L_{i,t})} \tag{9}$$

The P reward function is equivalent to the global reward function in the single rover problem. In a collective of rover setting, it has infinite rover-sensitivity (in the way rover sensitivity is defined in Section 2). This is because the P reward function for a rover is not affected by the states of the other rovers, and thus the denominator of Equation 2 is zero. However the P reward function is not factored. Intuitively P and G offer opposite benefits, since G is by definition factored, but has poor rover-sensitivity. The final reward function is the difference reward function. It does not have as high rover-sensitivity as P, but is still factored like G. For the rover problem, the difference reward function, D, becomes:

$$\begin{aligned} D_i &= \sum_t \left[\sum_j \frac{V_j}{\min_{i'} \delta(L_j, L_{i',t})} - \sum_j \frac{V_j}{\min_{i' \neq i} \delta(L_j, L_{i',t})} \right] \\ &= \sum_t \sum_j I_{j,i,t}(z) \left(\frac{V_j}{\delta(L_j, L_{i,t})} - \frac{V_j}{\min_{i' \neq i} \delta(L_j, L_{i',t})} \right) \end{aligned}$$

where $I_{j,i,t}(z)$ is an indicator function, returning one if and only if rover i is the closest rover to POI j at time t. The second term of the D is equal to the value

of all the information collected if rover i were not in the system. Note that for all time steps where i is not the closest rover to any POI, the subtraction leaves zero. As mentioned in Section 3, the difference reward in this case is easier to compute as long as rover i knows the position and distance of the closest rover to each POI it can see (and the second closest when rover i is the closest). In that regard it requires knowledge about the position of fewer rovers than if it were to use the global reward function. In the simplified form, this is a very intuitive reward function yet it was generated mechanically from the general form if the difference reward function [20]. In this simplified domain we could expect a hand-crafted reward function to be similar. However the difference reward function can still be used in more complex domains with a less tractable form of the global reward, even when it is difficult to generate and evaluate hand-crafted solution. Even in domains where an intuitive feel is lacking, the difference reward function will be provably factored and rover-sensitive.

In the presence of communication limitations, it is not always possible for a rover to compute its exact D_i, nor is it possible for it to compute G. In such cases, D_i can be compute based on local information with minor modifications, such as limiting the radius of observing other rovers in the system. This has the net effect or reducing the factoredness of the reward function while increasing its rover-sensitivity.

5 Results

We performed extensive simulation to test the effectiveness of the different rover reward function under a wide variety of environmental conditions and rover capabilities. In these experiments, each rover had a population of MLPs of size 10. The world was 75 units long and 75 units wide. All of the rovers started the experiment at the center of the world. Unless otherwise stated as in the scaling experiments, there were 30 rovers in the simulations. The maximum distance the rovers could move in one direction during a time step, d_{max}, was set to 3. The rovers could not move beyond the bounds of the world. The minimum observation distance, δ_{min}, was equal to 5.

In these experiments the environment was dynamic, meaning that the POI locations and values changed with time. There were as many POIs as rovers, and the value of each POI was set to between three and five using a uniformly random distribution. In these experiments, each POI disappeared with probability 2.5%, and another one appeared with the same probability at 15 time step intervals. Because the experiments were run for 3000 time steps, the initial and final environments had little similarity.

All of the experiments (except the last one) were done in a non-episodic domain, where the environment continually changed with time. The dynamic environment experiments reported here explore how rover control policies can be generalized from one set of POIs to another, regardless of how significantly the environment changes. Figure 2 shows an instance of change in the environment throughout a simulation. The final POI set is not particularly close to the initial

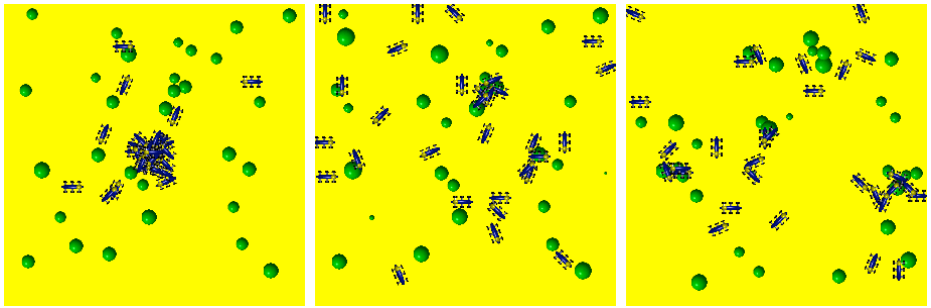

Fig. 2. Sample POI Placement. Left: Environment at time = 15. Middle: Environment at time = 150. Right: Environment at time = 1500.

POI set and the rovers are forced to focus on the sensor input-output mappings rather than focus on regions in the (x, y) plane.

5.1 Learning Rates

The first set of experiments tested the performance of the three reward functions in a dynamic environment for 30 rovers. Figure 3 shows the performance of each reward function. In all cases, performance is measured by the same global reward function, regardless of the reward function used to evolve the system. All three reward functions performed adequately in this instance, though the difference reward D_i outperformed both the perfectly rovers sensitive reward, P and the global reward G.

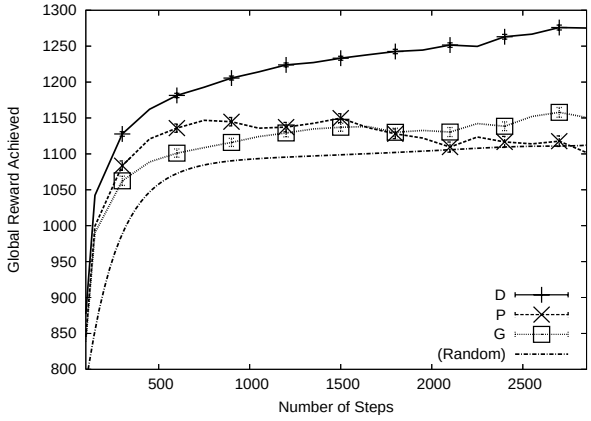

Fig. 3. Performance of a 30-rover collective for all three reward functions in noise-free environment. Difference reward function provides the best collective performance because it is both factored and rover-sensitive. (Random curve smoothed).

The evolution of this system also demonstrates the different properties of the rover reward functions. Rovers using all rewards improve initially, even rovers using random utilities (utilities returning uniformly random values between 0.0 and 1.0). This improvement happens since the domain is non-episodic, and the initial rover locations are very poor, so even rovers using random neural networks spread out to superior locations. After initial improvements, the system with the G reward function improves slowly. This is because the G reward function has low rover-sensitivity. Because the reward of each rover depends on the state of all other rovers, the noise in the system overwhelms the reward function. P on the other hand has a different problem: After an initial improvement, the performance of systems with this reward function decline. This is because though P has high rover-selectivity, it is not fully factored with the global reward function. This means that rovers selected to improve P do not necessarily improve G. D on the other hand is both factored and has high rover-sensitivity. As a consequence, it continues to improve well into the simulation as the reward signal the rovers receive is not swamped by the states of other rovers in the system. This simulation highlights the need for having reward function that are both factored with the global reward function and have high rover-sensitivity. Having one or the other is not sufficient.

5.2 Scaling Properties

The second set of experiments focused on the scaling properties of the three reward functions in a dynamic environment. Figure 4 shows the performance of each reward function at t=3000 for a collective of 10 to 70 rovers. For each different collective size, the results are qualitatively similar to those reported above, except where there are only 5 rovers, in which case P performs as well as G. This is not surprising since with so few rovers, there are almost no interactions among the rovers, and in as large a space as the one used here, the 5 rovers act almost independently.

As the size of the collective increases though, an interesting pattern emerges: The performance of both P and G drop at a faster rate than that of D. Again, this is because G has low rover-sensitivity and thus the problem becomes more pronounced as the number of rovers increases. Similarly, as the number of rovers increases, P becomes less and less factored. D on the other hand handles the increasing number of rovers quite effectively. Because the second term in Equation 3 removes the impact of other rovers from rover i, increasing the number of rovers does very little to limit the effectiveness of this rover reward functions. This is a powerful results suggesting that D is well suited to learning in large collectives in this and similar domains where the interaction among the rovers prevents both G and P from performing well. This result also supports the intuition expressed in Section 2 that Approach Two (i.e., evolving rovers based on the actions of the full collective) is ill-suited to evolving effective collectives in all but the smallest examples.

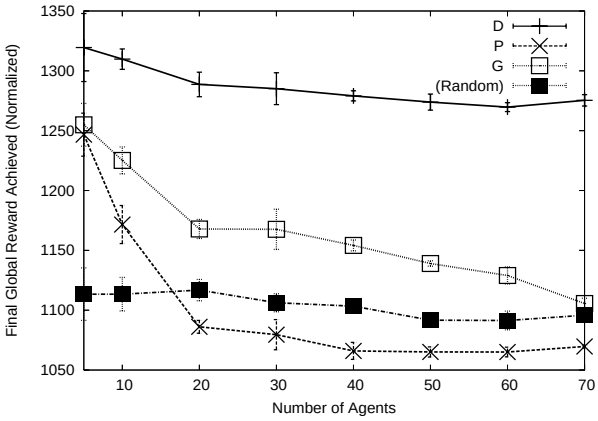

Fig. 4. Scaling properties of the three reward functions. The D reward function not only outperforms the alternatives, but the margin by which it outperforms them increases as the size of the collective goes up.

5.3 Learning with Communication Limitations

The third set of experiments tested the performance of the three reward functions in a dynamic environment where not only the rover sensors were noisy, but the rovers were subject to communication limitations. The communications limitations reduce the amount of information that is available to compute a rover's utility, by reducing the number of other rovers that a rover can observe. Figure 5 shows the performance of all three reward function when the rovers were only aware of other rovers when they were within a radius of 4 units from their current location. This amounts to the rovers being able to communicate with only 1% of the grid. (Because P is not affected by communication limitations, its performance is the same as that of Figure 3.)

The performance of D is almost identical to that of full communication D. G on the other hand suffers significantly. The most important observation is that communication limited G is no longer factored with respect to the global reward function. Though the rover-sensitivity of G goes up in this case, the drop in factoredness is more significant and as a consequence collectives evolved using G cannot handle the limited communication domain.

Figure 6 expands on this issue by showing the dependence of all three reward function on the communication radius for the rovers (P is flat since rovers using P ignore all other rovers). Using D provides better performance across the board and the performance of D does not degrade until the communication radius is dropped to 2 units. This is a severe restriction that practically cuts off the rover from other rovers in the system. G on the other hand needs a rather large communication radius (over 20) to outperform the collectives evolved using P. This result is significant in that it shows that D can be effectively used in many

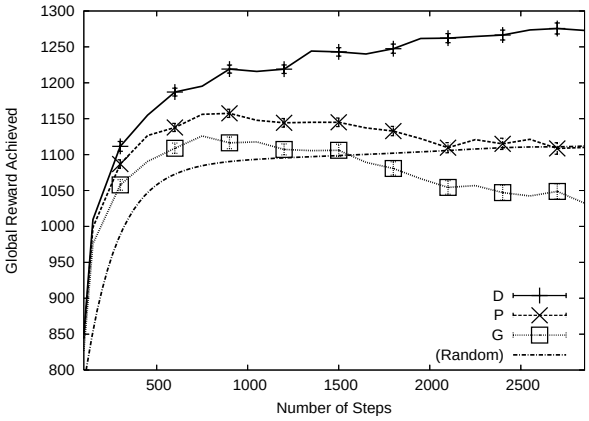

Fig. 5. Results for noisy domain under communication limitations. Rovers can only see of rovers covering an area of 1% of the domain. Difference reward is superior since it is both factored and rover-sensitive. (Random curve smoothed).

practical information-poor domains where neither G nor "full" D as given in Equation 3 can be accurately computed.

Another interesting phenomena appears in the results presented in Figure 6, where there is a dip in the performance of the collective when the communication radius is at 10 units for both D and G (the "bowl" is wider for G than D, but it is the same effect). This phenomenon is caused by the interaction between the degree of factoredness of the reward functions and their sensitivity. At the maximum communication radius (no limitations) D is highly factored and has high rover-sensitivity. Reducing the communication radius starts to reduce the factoredness, while increasing the rover-sensitivity. However, the rate at which these two properties change is not identical. At a communication radius of 10, the drop in factoredness has outpaced the gains in rover-sensitivity and the performance of the collective suffers. When the communication radius drops to 5, the increase in rover-sensitivity compensates for the drop in factoredness. This interaction among the rover-sensitivity and factoredness is domain dependent and has also been observed in previous applications of collectives [15, 17].

5.4 Leaning in Changing Episodic Environment

All the previous experiments were non-episodic, meaning that the environment is not reset over the course of a trial. In episodic domains, learning is performed over a series of episodes with the environment being reset at the beginning of each episode. This section shows results for the episodic version of the Rover Problem. In this problem the rovers move for fifteen time-steps to complete an episode and at the end of the episode the rovers are returned to their original position. In most episodic domains, the environment is also reset to its starting configuration at the beginning of the episode. However, to make this problem

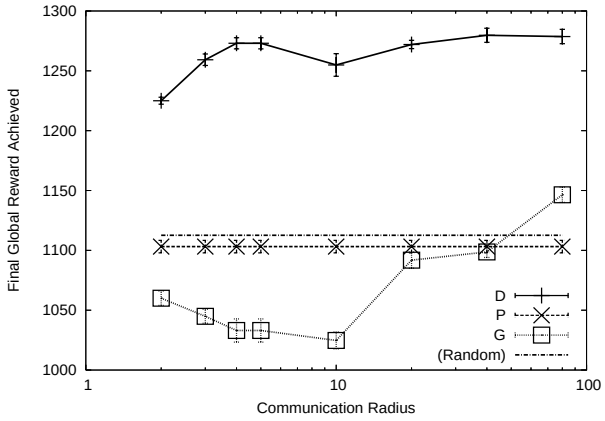

Fig. 6. Sensitivity of the three reward functions to the degree of communication limitations. Difference reward is not affected by communication limitations by as much as global reward.

more difficult, we reset the POIs to new random positions at the beginning of each trial. By placing the POIs this way, the rovers have to learn a general policy for efficiently navigating using sensors, and cannot form a specific policy to a single environmental configuration.

Figure 7 shows that rovers using D performed best in this scenario. Rovers using D were effective in generalizing the knowledge gained from exploring previous POI configurations and applying that knowledge to new POI configurations. In contrast, rovers using the P rewards were especially ineffective in this scenario. We attribute this to the congested nature of the problem, where the rovers competed rather than cooperating with each other. Since a rover's P rewards only returns the value of what that rover observes, a rover using the P rewards tends to move towards the highest valued POI in its area. However all the other rovers in that vicinity are also moving towards the same high-valued POI, and thus many other POIs are not properly observed.

6 Discussion

This paper presented a method for providing rover specific reward functions to directly evolve individual rovers in a collective. The fundamental issue in this approach is in determining the rover specific reward functions that are both aligned with the global reward function and are as sensitive as possible to changes in the reward of each member.

In dynamic environments rovers using the difference reward function D, derived from the theory of collectives, were able to achieve high levels of performance because the reward function was both factored and highly rover-sensitive. These rovers performed better than rovers using the non-factored perfectly rover-

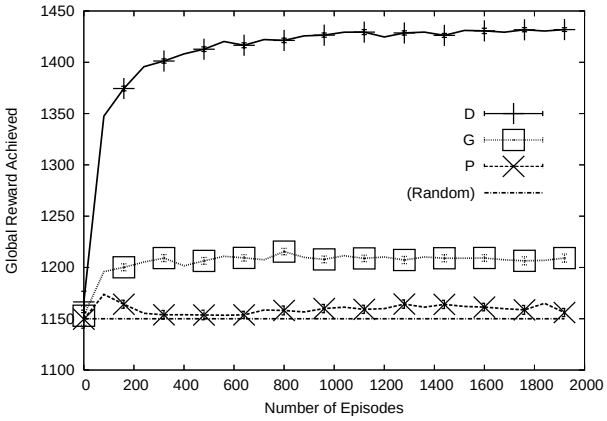

Fig. 7. Learning in an Episodic Domain with Changing Environment. Even in an episodic domain where the environment changes at the end of every episode, agents using the difference reward, D, are still able to achieve high performance.

sensitive reward and more than 400% better (over random rovers) than rovers using the hard to learn global rewards.

We then extended these results to rovers with limited communication capabilities and larger collectives. In each instance the collectives evolved using D performed better than alternatives and in most cases (e.g., larger collectives, communication limited rovers) the gains due to D increase as the conditions worsened. These results show the power of using factored and rover-sensitive reward functions, which allow evolutionary computation methods to be successfully applied to large distributed systems in real world applications where communication among the rovers cannot be maintained.

References

1. A. Agah and G. A. Bekey. A genetic algorithm-based controller for decentralized multi-agent robotic systems. In *In Proc. of the IEEE International Conference of Evolutionary Computing*, Nagoya, Japan, 1996.
2. A. Agogino, K. Stanley, and R. Miikkulainen. Online interactive neuro-evolution. *Neural Processing Letters*, 11:29–38, 2000.
3. A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.
4. T. Balch. Behavioral diversity as multiagent cooperation. In *Proc. of SPIE '99 Workshop on Multiagent Systems*, Boston, MA, 1999.
5. G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behavior. *Artificial Life*, pages 9: 255–267, 2003.
6. M. Dorigo and L. M. Gambardella. Ant colony systems: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

7. S. Farritor and S. Dubowsky. Planning methodology for planetary robotic exploration. In *ASME Journal of Dynamic Systems, Measurement and Control*, volume 124, pages 4: 698–701, 2002.
8. 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.
9. 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.
10. 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.
11. E. Lamma, F. Riguzzi, and L. Pereira. Belief revision by multi-agent genetic search. In *In Proc. of the 2nd International Workshop on Computational Logic for Multi-Agent Systems*, Paphos, Cyprus, December 2001.
12. A. Martinoli, A. J. Ijspeert, and F. Mondala. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29:51–63, 1999.
13. M. J. Mataric. Coordination and learning in multi-robot systems. In *IEEE Intelligent Systems*, pages 6–8, March 1998.
14. K. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, San Francisco, CA, 2002.
15. K. Tumer and A. Agogino. Overcoming communication restrictions in collectives. In *Proceedings of the International Joint Conference on Neural Networks*, Budapest, Hungary, July 2004.
16. K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.
17. K. Tumer and D. Wolpert. A survey of collectives. In *Collectives and the Design of Complex Systems*, pages 1,42. Springer, 2004.
18. 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.
19. D. Whitley, F. Gruau, and L. Pyeatt. Cellular encoding applied to neurocontrol. In *International Conference on Genetic Algorithms*, 1995.
20. D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.