Agent Fitness Functions for Evolving Coordinated Sensor Networks

Christian Roth
HS Offenburg
christian.roth@fh-offenburg.de

Matt Knudson
Carnegie Mellon University
matt.knudson@west.cmu.edu

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

ABSTRACT

Distributed sensor networks are an attractive area for research in agent systems. This is due primarily to the level of information available in applications where sensing technology has improved dramatically. These include energy systems and area coverage where it is desirable for sensor networks to have the ability to self-organize and be robust to changes in network structure. The challenges presented when investigating distributed sensor networks for such applications include the need for small sensor packages that are still capable of making good decisions to cover areas where multiple types of information may be present. For example in energy systems, singular areas in power plants may produce several types of valuable information, such as temperature, pressure, or chemical indicators.

The approach of the work presented in this paper provides agent fitness functions for use with a neuro-evolutionary algorithm to address some of these challenges. In particular, we show that for self-organization and robustness to network changes, it is more advantageous to evolve individual policies, rather than a shared policy that all sensor units utilize. Further, we show that using a difference objective approach to the decomposition of system-level fitness functions provides a better target for evolving these individual policies. This is because the difference evaluation for fitness provides a cleaner signal, while maintaining vital information from the system level that implicitly promotes coordination among individual sensor units in the network.

Categories and Subject Descriptors

I.2.6 [**AI**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Distributed Sensor Network, Agent Fitness, Neuro-Evolution

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

In large energy systems (e.g. power plants) an increasing number of sensors is essential to achieve good control behavior and system safety. Sensors in these energy systems might be temperature or pressure sensors, but it is also possible that a highly developed sensing entity is able to sense multiple types of information from the same location. These distributed sensor networks (DSNs) are composed of a large number of highly cooperative sensing devices. Compared to ad-hoc sensor networks there are advantages when dealing with DSNs. While using intelligent sensor placement strategies there might be no need for engineering the network. It is possible that the sensing units are randomly deployed within an unknown or inaccessible environment and are able to change their position autonomously. This implies self-organization capabilities of sensors. In a dynamic environment with different types of information and capabilities of sensors, these intelligent sensor networks are able to adapt their individual capabilities, for example changes in sensing range and efficiency or their control policies to achieve an acceptable system health. A DSN is also characterized by a dynamic structure, meaning adding or removing of sensors after deployment.

Distributed sensor groups which communicate with a centralized controller to increase the system performance have been largely researched. However, this approach is often inefficient because a lot of communication among sensors and the controller is necessary, which affects the communication bandwidth. Instead of using a centralized controller in a large energy system it can be more efficient to control the behavior of sensors with individual, non-centralized policies for intelligent sensor units. This also leads to challenges in dealing with the creation of DSNs. Important issues in these networks are the problems of sensor placement strategies which also include the localization of points of interest and other sensing entities. It is also necessary to handle cases of sensor failures, where other sensors should be able to recover system health autonomously. In a dynamic environment, adaptation strategies are necessary to achieve the desired performance.

In this work, we developed a solution strategy to deal with such challenges in distributed sensor networks. This includes an evolutionary algorithm based on artificial neural networks for individual sensor control policies and different multiagent objective functions for evaluation of the developed policy. With this approach, challenges in the system design process and system health management can be addressed. In a dynamic environment, meaning changes in

number of sensors or sources of information, adaptation of individual control policies is possible. Simulated tests took place under ideal network conditions where the environment consisted of a two dimensional plane area without any obstacles, only surrounded by a border where sources of information and sensor agents were distributed. Successful demonstration of this work leads to reliable and robust sensor networks which are able to self-deploy and reorganize their distribution without the need of a centralized control for system design and system health management.

The rest of the paper is organized as follows. In Section 2, we describe our model of the environment with sources of information and sensor agents. Section 3 covers the sensor agents including environment observation, state to action mapping and different objective functions for evaluation of agent fitness. Section 4 presents the experimental results of this work regarding standard evolution within a non-dynamic environment, and system robustness in case of sensor failures and policy reconfiguration in a dynamic environment if the number of sources of information or sensor agents changes.

1.1 Related Work

Related work in this domain addresses the description of sensor networks [3]. This includes a definition of the term "coverage" in such networks [15, 14, 11]. In [6], methods for an optimal sensor coverage with a minimum number of sensors is provided. A better strategy may be to have an over-coverage (redundancy), to recover system performance in case of sensor failures for a better system robustness. Coordinating mobile sensor networks in an unknown environment, [10] presents a potential-field-based approach to achieve a maximum coverage. A problem in wireless sensor networks is often a spatial localization of points of interest and other sensors of the group. To solve this problem, communication among the sensing entities is often necessary because in most of the cases there is neither an *a priori* knowledge of locations in a static environment nor has a sensing entity a complete observability of the environment [5].

Non-learning approaches to coordination, based on planning, swarms, auctions, and domain specific algorithms have also been investigated. For example, role allocation and plan instantiation have proven successful for large multi-robot systems [20] and nondeterministic planning has been applied to adjust for uncontrollable robots with goals differing from the majority of the system [4]. Several successful applications of robot coordination include search and rescue [23, 24], robotic soccer [21], mobile sensor networks [9], and mine collection [7].

The solution strategy in this work includes ideas from the field of learning in multiagent systems. These ideas provide different techniques for solving problems in DSNs [13, 22]. Thus, the term "agent" will be used as a synonym for a sensing entity meaning an intelligent sensor unit able to sense multiple types of information and behave autonomously by using different control strategies. Particularly for large multiagent systems, new approaches are needed. They include using Markov Decision Processes for online mechanism design [17], developing new reinforcement learning based algorithms [2, 8], or devising agent-specific objective functions [1].

2. SELF-ORGANIZING SENSOR NETWORKS

Developing control policies for a network of sensor units requires an analogous domain in which to evaluate algorithm and agent performance. For self-organization, the agents must have the ability to perceive the environment surrounding them as well as the capability to move to a desirable location. In this domain, we have defined a two dimensional plane area containing information sources that are randomly distributed. Figure 1 shows a sample distribution of sources and agents.

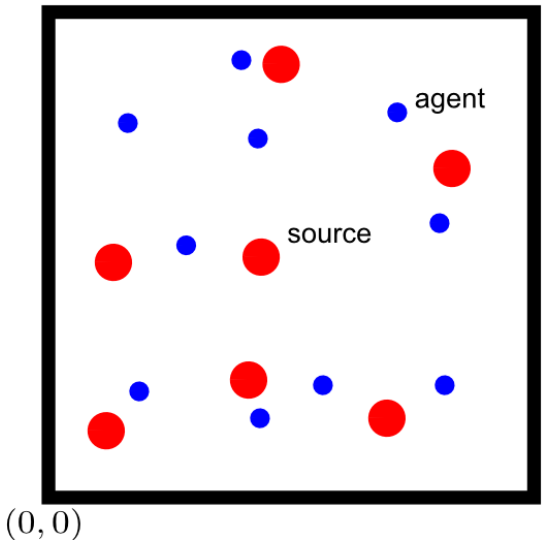

Figure 1: Two dimensional plane area with randomly distributed information sources and sensor agents surrounded by a border.

2.1 Modeling Sources and Agents

The "sources of information" in the world are defined by a *type strength*. For example, these sources may be temperature, pressure, chemical, or a combination of all three. The location of these sources in the environment is in fixed locations, initialized randomly at the beginning of an experiment. The sensor agents in this domain are able to observe the sources and what type(s) of information they offer. How well an agent can see a certain type of information is indicated by the agents' individual *type efficiencies*. As with sources, sensor agents are able to relate to more than one type of information with variable efficiencies. For example, a sensor agent may be capable of sensing both temperature and pressure, but due to damage or age only have half of design capability for temperature. In contrast to the sources, the agents can move in the environment to look for good locations, or in the case of agent failures or environment changes, relocate to recover system performance.

Sources and agents are described by several parameters:

1. Each has a certain position in the world indicated by Cartesian coordinates (x_0, y_0), where position $(0, 0)$ is on the left bottom corner (See Figure 1). The sources are located on fixed positions, where agents have the ability to move on the plane area within the boundaries to change their locations.

2. Sources and agents have individual information type characteristics. Sources can offer different types of information with an individual strength for each type indicated by $\nu_{j,t}$, where j indexes the source numbers and t the information types. Similarly, the agents' information type efficiencies given by $\mu_{i,t}$, where i indexes the agent numbers and t the information types. As a general rule, the source strengths are greater than the agent efficiencies, which requires that multiple agents are needed to "cover" each source.

3. The final parameter, applicable just to agents, is the agent drop off function (See Figure 2). This function indicates how agents cover sources of information. The closer they are to a source of information, the better their influence regarding individual type efficiencies. As an assumption in this work, the drop off function consists of two modified Gaussian curves, summed together (See Equation 1). This function not only promotes good fitness, but approximates the way in which some real world types of information propagate through an environment.

The agent drop off function is given by:

$$f_a(x, y, x_0, y_0) = a\eta e^{-\frac{(x-x_0)^2+(y-y_0)^2}{\lambda_W}} + a(1-\eta)e^{-\frac{(x-x_0)^2+(y-y_0)^2}{\lambda_N}} \quad (1)$$

where $a \in \mathbb{R}_-$ is the amplitude and the factor $\eta \in [0, 1]$ weights the two terms. One term indicates the wide region and the other the nearby region of the intensity distribution, where the range factors λ_W and λ_N specify their respective extent. The current position of the agent is given by x_0, y_0. As an example, Figure 2 shows the drop off function for an agent.

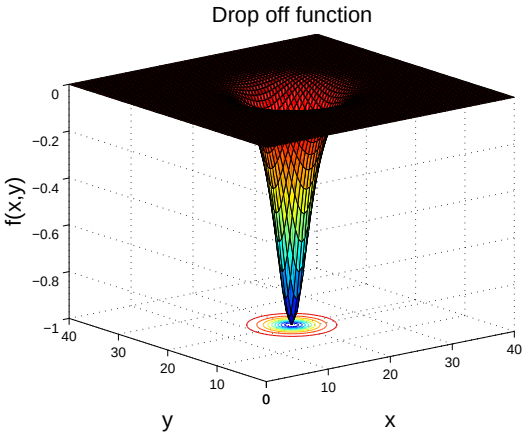

Figure 2: Agent "drop off" function. Indicates how an agents' influence on the environment exponentially diminishes with distance.

2.2 Covering a Source

The coverage of a source for a certain type of information depends on the information type strengths $\nu_{j,t}$ and the relationships between a source and the agents indicated by $\nu'_{j,t} \in \mathbb{R}_+$ (See Equation 2). In this equation, f_a calculates the influence of agent i to source j weighted by the agents' information type efficiency $\mu_{i,t}$. The parameter $\nu'_{j,t}$ decreases its value as more agents are related to this source and type. For coverage calculation, Equation 3 is used. A coverage with a value of 0.0 means that no agent is related to the type j of information the source offers, whereas a value of 100.0 means that an information type is completely covered by agents in the environment.

$$\nu'_{j,t} = \nu_{j,t} + \sum_i (\mu_{i,t} \cdot f_a(x_j, y_j, x_i, y_i)) \quad (2)$$

$$C_{j,t} = \frac{\nu_{j,t} - \nu'_{j,t}}{\nu_{j,t}} \cdot 100.0 \quad (3)$$

3. AGENT DESCRIPTION

The agents in this domain operate as mobile sensor units for special types of information. As discussed in the previous section, a sensing unit can physically sense more than one type of information. Each agent therefore is able to observe the location of sources, and the strengths of information that source provides. Additionally, the agents can observe other agents and their information type efficiencies. These observations comprise the state of the agent, as they do not retain knowledge of their locations to promote generalization across multiple environment configurations.

3.1 Environment Observation

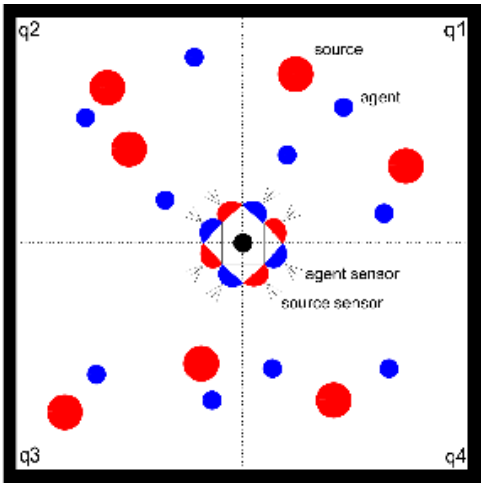

Figure 3: Agent observations of the surrounding environment, divided into other agent and information source sensing, additionally separated into four quadrants.

The environment around the agent is broken up into four quadrants q. For each quadrant, the observations are separated into (1) sensing of sources $S_{i,t}(q)$ and (2) sensing of other agents $A_{i,t}(q)$ where t indexes the type of information and i the agent making the observations (See Figure 3).

The measurements for $S_{i,t}(q)$ consist of the information type strength $\nu_{j,t}$ of source j located in quadrant q, the maximum information type strength, where all source strengths are considered, and the distance δ between the agent and this source. Similarly, for calculating the measurements for

$A_{i,t}(q)$, the information type efficiency $\mu_{a,t}$ of an agent a located in quadrant q applies. For computing the distance, the Euclidean norm is used with a constant added to prevent singularities when an agent is close to a source or another agent. In addition to controlling the movement of the agent, the agents goal is to learn which type of information is interesting for them to promote high system level performance.

$$\begin{aligned} \delta_{i,j} &= ||i-j|| + 1.0 \\ S_{i,t}(q) &= \sum_{j \in q} \frac{\nu_{j,t}}{max(\nu) \cdot \delta_{i,j}} \end{aligned} \tag{4}$$

$$\begin{aligned} \delta_{i,a} &= ||i-a|| + 1.0 \\ A_{i,t}(q) &= \sum_{a \in q} \frac{\mu_{a,t}}{max(\mu) \cdot \delta_{i,a}} \end{aligned} \tag{5}$$

3.2 Mapping States to Actions

The two parameters $S_{i,t}(q)$ and $A_{i,t}(q)$ describe the state of agent i. However, the mapping of state/action combinations takes place via an evolution algorithm. First, agent i assigns a "quality" value for each quadrant q depending on the state information. After this assignment, these quality values $Q_i(q)$ are used to select a quadrant for the agent action. This quality represents the importance of the quadrant for investigation. It however is not intuitive, which is where the evolutionary algorithm comes in. Simply choosing a quadrant that has many sources and few agents does not necessarily promote coordination in a team of many individuals. This system property is discussed further in the experimentation section below.

The quality value for a quadrant is assigned by an artificial neural network [19]. For this work, a two-layer, feed forward, sigmoid activated network with $(2 \cdot TYPES)$ input units, 16 hidden units and 1 output unit is used. Previous experimentation selected the number of hidden units and various other parameters that produced successful behavior.

For each of the four quadrants, the neural network produces an output from the state information for that quadrant. This output is used as the quality value $Q_i(q)$ for the quadrant and the quadrant with the highest quality is chosen for the action. The angle of the direction to move is in the middle of this quadrant. With this angle, the new position of the agent is given by:

$$\begin{aligned} x_{k+1} &= x_k + step\_size \cdot \cos(\frac{\varphi \cdot \pi}{180^\circ}) \\ y_{k+1} &= y_k + step\_size \cdot \sin(\frac{\varphi \cdot \pi}{180^\circ}) \end{aligned} \tag{6}$$

where the new coordinates (x_{k+1}, y_{k+1}) are calculated depending on the current position (x_k, y_k), the *step_size* and the direction to move, indicated by φ. The step numbers are indexed by k.

3.3 Agent Fitness Functions

Training the neural networks occurs with a neuro-evolution search algorithm [12, 16]. In addition to a shared policy, where all agents develop one control policy that they all use, individual control policies were evolved. In this case, each agent maintains its own population of neural networks and develops its own policy. The evaluation of network performance is done at the end of a evaluation episode. The shared policy is evaluated only with the global objective, whereas the individual policies are evaluated with two additional fitness functions.

The global objective function $G(z)$ measures the full system performance, given by Equation 7. In this equation, the sum over all coverage values is calculated and divided by the product of the number of sources j and types t in the world to ensure that the values of the global objective are in the range $[0.0, 100.0]$ to allow for comparison of performance across varying situations.

$$G(z) = \frac{\sum_j \sum_t C_{j,t}}{j \cdot t}, \{G(z) \in \mathbb{R} | 0.0 \leq G(z) \leq 100.0\} \tag{7}$$

The local objective $L_i(z)$ is given by Equation 8 and applies only for ranking the fitness of an individual policy in this multiagent system. For calculating $L_i(z)$, all the agents except agent i are deactivated when calculating $\nu'_{j,t}$ of the sources. In this way, these values depend only on the relationship of agent i. After this process, all other agents are reactivated.

$$L_i(z) \equiv G(z_{+i}) \tag{8}$$

The difference objective $D_i(z)$ is given in Equation 9. For calculating $G(z_{-i})$, agent i is deactivated when computing the relationship $\nu'_{j,t}$ between all other agents and the source j. This removes agent i's influence on the second term, which after the subtraction of the two terms, gives a low noise fitness function for agent i to determine the impact it had on system level performance.

$$D_i(z) \equiv G(z) - G(z_{-i}) \tag{9}$$

4. EXPERIMENTATION

A large number of experiments were run to evaluate algorithm and fitness function performance. The simple domains where the agents had full observability of the surrounding environment (had total knowledge) are excluded in this report as they are unrealistic to real-world problems. They were used as a proof of concept and demonstrate that the solution strategy discussed in preceding sections was appropriate [18]. The experiments discussed here are a more realistic configuration, where the agents did not have full knowledge of the environment, and where sources had multiple types of information to be covered.

Several parameters dictate the experiment configuration. Specifically:

- **Plane Area**

  The two dimensional plane area had a size of 40.0×40.0. In the beginning of each episode, the location of sources and the starting locations of agents were randomly distributed. However, the minimum distance between sources was set to 5.0 to ensure that sources do not have an influence on each other. In doing so, agent behavior could be easily analyzed, and it could be determined if an agent would select a location between sources as a possibility.

- **Source Parameters**

  The information type strength values $\nu_{j,t}$ of sources were randomly initialized in a range of 3.0 to 6.0. For multi-type experiments it was also possible that a certain type strength was set to 0.0, meaning that the source does not offer this type of information.

- **Agent Parameters**

  Similarly, the information type efficiency values $\mu_{i,t}$ of agents were randomly initialized in a range of 1.0 to 2.0. For multi-type experiments it was also possible that a certain type efficiency was set to 0.0, meaning that an agent is not able to measure this type of information. The agents were given a small wide range value (Section 2.1) which meant that the agent had very little influence on a source until it was very close. This forced the agent to make clear decisions on where in the environment was desirable.

- **Training Configuration**

  The network population for evolving a shared policy consisted of $N = 50$ neural networks. For individual policies, a population of $N = 20$ was utilized. The number of input units increases with the number of possible information types in the world. Therefore, the network had $2 \cdot TYPES$ input units, 16 hidden units and 1 output unit. The number of hidden units must be increased if more types of information are available. Experiments in this work showed that a number of 16 hidden units with three types of information is sufficient.

- **Experiment Configuration**

  In each training episode, the number of steps was set to 1500. With this high number of steps it is possible for all the agents to move to a destination where the agent can improve the system performance the most. Ranking of the network under consideration for the episode occurred once at the end of an episode. Training with a certain configuration ended after 1500 episodes and was repeated 30 times to average the results for analysis. The standard deviation of the results is indicated by error bars. In the last 500 episodes, the best network (control policy) in the population is used exclusively, meaning that evolution was turned off to analyze converged behavior.

4.1 Results

In this experiment domain, the environment contained 5 sources, each with 3 types of information. The type strengths were randomly initialized such that each information source had at least 2 non-zero type strengths, with the majority of the sources containing all three types. Figure 4 shows the fitness curves for both the single policy and individual policy as well as the objectives discussed above. With 10 sensor units and 5 sources, the environment can not be completely covered. This was done intentionally to determine if the agents would learn to "do the best" that they could to cover as much information as possible. As shown, when agents develop their own individual polices, they outperform evolving a shared policy. The exception here is using the system objective from Equation 7, where the fitness function is too noisy for the agents to determine their contribution to the system as a whole.

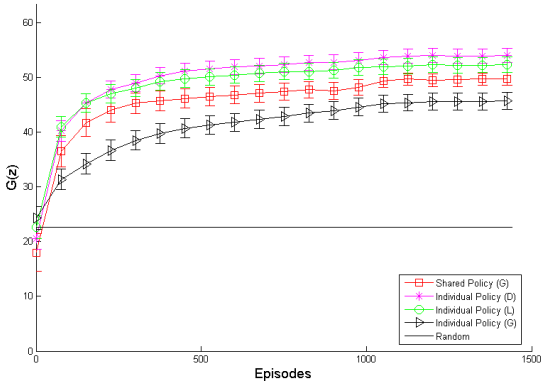

Figure 4: The result of evolution in an environment with 10 agents and 5 information sources. Developing a shared policy is compared with individual policies trained using the system (G), local (L), and difference (D) objective functions.

While using the difference objective to learn individual policies performs best, the agents trained with the local objective perform statistically similar. This is due to the limited number of agents and sources, which produces a situation that is forgiving to greedy behavior. Since the environment can not be fully observed, the agents must simply move toward sources that they can most efficiently cover. This brings the difference and local objectives into alignment regarding the level of coordination required of the agents. Alternatively, as shown in Figure 5, as the number of sensor units in the environment increases, the level of coordination required also increases, and the local objective no longer performs as well.

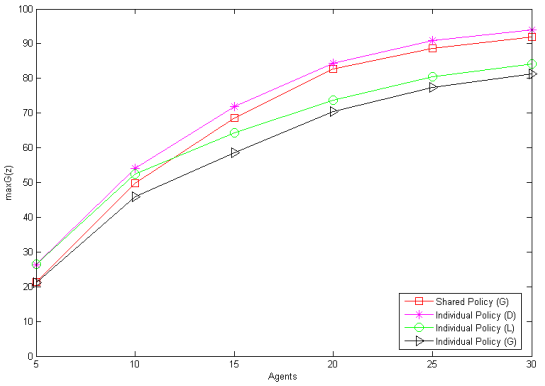

Figure 5: The maximum system level performance achieved is plotted versus a varying number of agents. Developing a shared policy is compared with individual policies trained using the system (G), local (L), and difference (D) objective functions.

We also show that while using the difference objective for evolution again outperforms all others, evolving a shared

policy is similarly stable for an increase in the number of agents within the system. This result indicates that coordination is successful in this domain both when the agents develop their own unique policies, and when they collectively develop a single policy. Since these two approaches are fundamentally different, and upon examining the policies learned, we can conclude that the difference objective indeed produces a fitness signal that closely matches the system level objective (as that is what is used when evolving a shared policy). However, evolving a shared policy historically does not produce good behavior in stochastic coordination domains.

4.2 System Robustness

Redundancy in a sensor network is often necessary because of the possibility of sensor failures. A system recovery where the agents realize failures of other agents and are able to recover the system performance autonomously can be applied in online system health management. In these experiments, simulated sensor failures took place to all agents located around one random selected source of information. Agents located around other sources or distributed in the plane area without any influence to sources should realize the agent failures and reorganize to cover the environment as best as is possible for the situation.

A configuration of 30 agents and 5 sources with three types of information was chosen for this type of experiment. First, training individual policies evaluated with the difference objective occurred until the fitness curve converged. After this standard procedure, each agent selected the best ranked network from its network population and used this network within the next episode as an individual control policy. During this episode, the agents were moving within the environment to find a good location. However, at the halfway point in time, agent failures on a random source were simulated. The task of the other agents was to recover the system performance by changing their locations autonomously.

In Figure 6 the system performance for this situation is shown. The figure shows that after approximately 200 steps the quality of the system performance was at 100.0%. At step 750, five agents surrounding a randomly selected source failed. When this failure occurs, the other agents in the system observed the failure and autonomously reconfigured to recover as much system performance as possible.

During experimentation, it was observed that the agents on occasion did not move or were unable to recover any system performance. While rare, we found it important to discuss, particularly the two reasons why this occurred:

1. Other agents were far enough from the point of failure that they were not encouraged to move away from the source they were currently covering. This is a realistic situation in the real-world, as it is undesirable for agents very far away from a failure to abandon their current good location to chase after a reconfiguration that would significantly reduce system performance.

2. The point of agent failures was too far away to be observed by the other agents in the system. Recall that the sensor units have only partial observability of the environment surrounding them, and therefore the potential exists that a suddenly uncovered source may go unobserved. In our experiments, sensor units and sources were randomly distributed, which allowed for a good initial network organization.

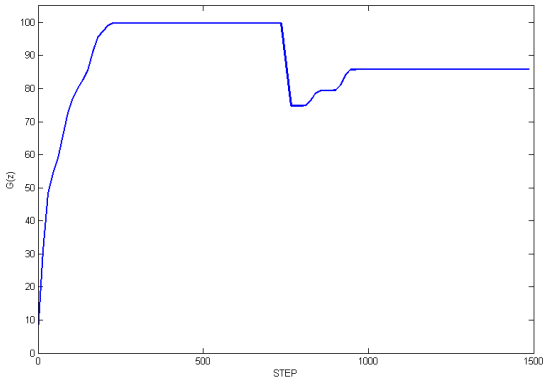

Figure 6: *System Robustness*: After 750 steps, five agents surrounding a randomly selected source failed. Remaining sensor units are shown to reconfigure, recovering as much system performance as possible for the situation.

4.3 Policy Reconfiguration

In the above experiments, evolution was not done after a change in the system, particularly the failure of sensor units. This was done to determine if the learned policies themselves were sufficient to promote agent reconfiguration should such an event occur. In the forthcoming experiments, we determine if the agents can modify their current policies and learn to reconfigure not only to accommodate agent or information source failures, but also to accommodate the *addition* of agents or sources. This is to determine whether the network can modify the individual control policies after convergence without the need to retrain from scratch.

As an example, a configuration of 15 agents and 5 sources with three different types of information was chosen. The training of the individual policies was evaluated with the difference, local or system objective. After 1000 evolution episodes when fitness was converged, 5 agents were either removed from or added to the system. Additionally an experiment were instead of agents, 3 sources were added to the system. Removing sources was found to be a trivial event, as there was an abundance of agents remaining and therefore little or no change occurred in the relative system performance.

Each agent maintains its own individual network population, which has already converged. However, the added agents initialized their network population randomly when added to the system. The goal of these experiments was to achieve the same results with a new number of agents or sources in a system like in the experiments shown before. To achieve these results, new agents had to learn their individual policies and the other agents had to adapt their policies in consideration of the new circumstances. For adding new sources to the system, all agents had to adapt their policies.

The fitness curves given in Figure 7 show that agents were able to adapt their individual policies in case of network reconfiguration, if agents were removed from the system. As already seen in Section 4.1, individual policies evalu-

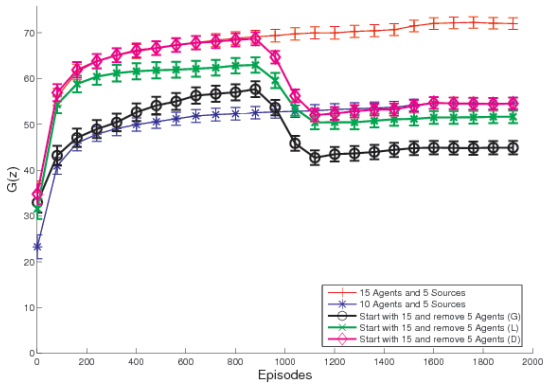

Figure 7: The results of evolution where the environment begins with 15 agents and 5 sources. After approximate convergence, five agents are removed from randomly selected information sources. Evolution in similar systems to the initial and final configurations are compared against the system (G), local (L), and difference (D) objectives when the removal occurs.

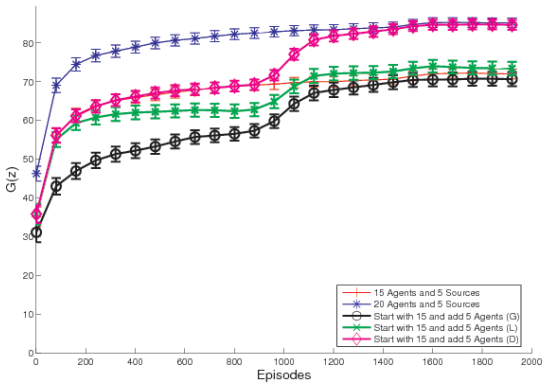

Figure 8: The results of evolution where the environment begins with 15 agents and 5 sources. After approximate convergence, five agents are added at random locations within the environment. Evolution in similar systems to the initial and final configurations are compared against the system (G), local (L), and difference (D) objectives when the addition occurs.

ated with the difference objective outperform the evaluation of individual policies with the system or local objective. Compared with the results given in Figure 5, the maximum system level performance for these two objectives is even slightly lower after adaptation. For comparing the difference objective, the fitness curve in a system with 10 agents is also shown. Compared with these results, there is no drop in system performance when the policies were trained with a higher number of agents.

In Figure 8, the results show that the added agents were able to develop their individual policies and preexisting agents were able to adapt their individual policies in the case of network reconfiguration where new agents are added to the system. As expected, ranking the networks with the difference objective achieved the best results compared to local and system objective. Here, the differences in system level performance between the local and difference objectives are greater because of a higher number of agents in the system after adding new agents. As discussed in Section 4.1, a higher number of agents leads to a better performance of the difference objective compared to the local objective. To compare the results of the difference objective, the fitness curve of a system with 20 agents is also shown. After new agents were added to the system, it took approximately 250 episodes to reach the same system level performance as the system trained with 20 agents from the beginning.

Figure 9 shows the results of evolution for the situation where new sources were added to the system. After adding the new information sources, the system performance settles slightly below that of a system with the same number of agents and sources. This is small change is due to the agents attempting to balance the amount of information gathered by settling in locations between sources. Since the agents have already settled on a source to sense, when another source enters the environment nearby, they will move to balance the coverage between two sources, rather than move to a single source to cover. However, we again see that allowing

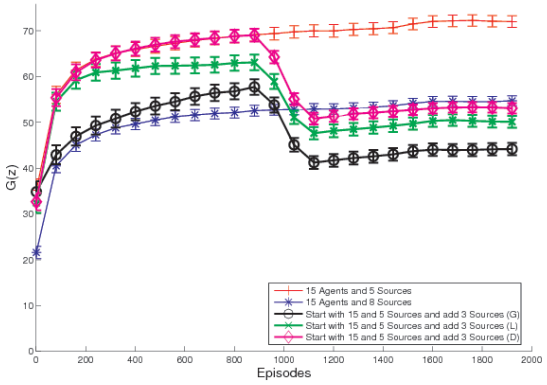

Figure 9: The results of evolution where the environment begins with 15 agents and 5 sources. After approximate convergence, three information sources are added at random locations within the environment. Evolution in similar systems to the initial and final configurations are compared against the system (G), local (L), and difference (D) objectives when the addition occurs.

the agents to develop their own control policies with ranking done via the difference objective far outperforms using the system or local objective. In addition, the performance difference is slight enough to indicate tuning for a specific domain would reduce or eliminate the disparity in system performance.

5. CONCLUSION

In this work, we presented an evolutionary algorithm evaluated with difference agent fitness functions for solving problems regarding distributed sensor networks. As the results

show, in a complex environment with multiple types of information, individual control policy development based on the difference objective function outperforms all the other algorithms and objective functions. These developed control policies can be used for both the system design process and system health management. In the case of system design, there would be no need for a lengthly engineered solution because intelligent sensors will be have the ability to self-deploy and organize the network autonomously. The experiments regarding system robustness and policy reconfiguration showed that the sensor agents are able either to adapt existing control policies in the case of a dynamic environment or use their policies for a system recovery in case of sensor failures within the system.

The primary area of further research we are pursuing is communication among sensor units. It is desirable to get enough information about both the environment and the properties of other sensor entities in the group. This overcomes challenges in unit separation making portions of the environment unobservable. For example, in a situation where sensors are dropped by an airplane or where sensor failures occur and a reorganization is advisable, the units also need the ability to navigate and move within the world to find locations for a good system performance. While the results in this paper show that with evolutionary algorithms based on artificial neural networks good control policies can be developed, further work will also consider communication. As with many multiagent domains, communication among agents can help to increase the overall system performance by giving agents more information about the environment when observation capabilities are limited.

6. REFERENCES

[1] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

[2] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1724–1729, May 2006.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine 40, 8*, 2002.

[4] M. H. Bowling, R. M. Jensen, and M. M. Veloso. *Planning in Intelligent Systems: Aspects, Motivations and Methods*. John Wiley and Sons, Inc., 2005.

[5] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann. Scalable Coordination for Wireless Sensor Networks: Self-Configuring Localization Systems. In *Proceedings of the Sixth International Symposium on Communication Theory and Applications ISCTA01*, 2001.

[6] S. S. Dhillon and K. Chakrabarty. Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks. In *Proc. of IEEE Wireless Communications and Networking Conference*, pages 1609–1614, 2003.

[7] D. Goldberg and M. J. Mataric. Maximizing reward in a non-stationary mobile robot environment. *Autonomous Agents and Multi-Agent Systems*, 3(6):281–316, 2003.

[8] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Int. Conference on Machine Learning*, pages 41–48, 2002.

[9] A. Howard, M. J. Mataric, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.

[10] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *Proc. Int. Conf. Distributed Autonomous Robotic Systems*, 2002.

[11] C.-F. Huang and Y.-C. Tseng. The Coverage Problem in a Wireless Sensor Network. *Mobile Networks and Applications, v.10, n.4*, 2005.

[12] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2010.

[13] V. Lesser, C. L. O. Jr., and M. Tambe. *Distributed Sensor Networks - A Multiagent Perspective*. Kluwer Academic Publishers, 2003.

[14] B. Liu and D. Towsley. A Study of the Coverage of Large-scale Sensor Networks. In *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS04), Fort Lauderdale, FL*, pages 475–483, 2004.

[15] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks. In *IEEE INFOCOM*, pages 1380–1387, 2001.

[16] D. Moriarty and R. Miikkulainen. Forming Neural Networks through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5, 1998.

[17] D. Parkes and S. Singh. An MDP-based approach to online mechanism design. In *NIPS 16*, pages 791–798, 2004.

[18] C. Roth. Agent Objectives for Evolving Coordinated Sensor Networks. Master's thesis, University of Applied Sciences Offenburg - in association with Oregon State University, 2010.

[19] S. J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.

[20] P. Scerri, J. Giampapa, and K. Sycara. Techniques and directions for building very large agent teams. In *Int. Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS*, pages 79–84, 2005.

[21] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165Ű188, 2005.

[22] L. Tong, Q. Zhao, and S. Adireddy. Sensor Networks with Mobile Agents. In *Proc. 2003 Military Communications Intl Symp*, pages 688–693, 2003.

[23] J. Wang, M. Lewis, and P. Scerri. Cooperating robots for search and rescue. In *Agent Technology for Disaster Management Workshop at AAMAS'06*, 2006.

[24] X. Zheng, S. Jain, S. Koenig, and D. Kempe. Forest-based multirobot coverage. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2318–2323, 2005.