DOE Award Number: DE-FE0000857

Title: Distributed Sensor Coordination for Advanced Energy Systems

Final Report

Reporting Period: Nov, 1, 2009 - July 31, 2013

**Principal Investigator
Dr. Kagan Tumer
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University**

**Email: kagan.tumer@oregonstate.edu
Voice: (541) 737 9899**

**Date: Nov, 4, 2013
DOE Award Number: DE-FE0000857**

**Address of Submitting Organization:
Oregon State University
Office of Post Award Administration
Corvallis, OR 97339-1086**

**Prepared for:
U.S. Department of Energy
National Energy Technology Laboratory**

Disclaimer:

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Contents

1	Abstract	5
2	Aligning Private and Team-Based Agent Objectives to Improve System Efficiency	6
2.1	Self-Organizing Sensor Networks	6
2.1.1	Modeling Sources and Agents	7
2.1.2	Covering a Source	8
2.2	Agent Description	8
2.2.1	Environment Observation	9
2.2.2	Mapping States to Actions	10
2.2.3	Agent Learning	10
2.3	Experimental Set-up	11
2.4	Simulation Results	12
3	Improving System Efficiency and Scalability with New Objective Functions	15
3.1	Sensor Models	15
3.2	The Defect Combination Problem (DCP)	15
3.3	The Time-Extended Defect Combination Problem (TEDCP)	16
3.4	Multi-Target Defect Combination Problem	17
3.5	Description of Algorithm	18
3.5.1	Difference Reward	19
3.5.2	Derivation of Rewards for DCP	20
3.6	DCP Experiments	21
3.6.1	Standard DCP	21
3.6.2	Time-Extended DCP	23
3.6.3	Multi-Target DCP	25
4	Reconfigurability for Changes in Environmental and Network Conditions	26
4.1	Rankine Cycle Defect Combination Problem	26
4.1.1	Rankine Cycle	26
4.1.2	DCP For Power Plants	28
4.2	Derivation of Rewards for Rankine Cycle DCP	29
4.2.1	Global Reward	30
4.2.2	Difference Reward	30
4.2.3	Expected Difference Reward and Average Difference Reward	31
5	Hardware-Based Simulation Analysis	32
5.1	Sensitivity Error	33
5.2	Sensitivity Error and Measurement Noise	33
5.3	Sensor Failure and Measurement Error	34
5.4	Changing System Conditions	35
5.5	Hardware-Based Simulations Summary	36

6	Project Summary	38
7	Cost Reporting	40
8	Task Discussion	41
8.1	Task 1.0 : Project Management and Planning	41
8.2	Task 2.0 : Determine Sensor Configurations	41
8.3	Task 3.0 : Determine Sensor Objectives	41
9	Milestone Reporting	41

1 Abstract

The ability to collect key system level information is critical to the safe, efficient and reliable operation of advanced energy systems. With recent advances in sensor development, it is now possible to push some level of decision making directly to computationally sophisticated sensors, rather than wait for data to arrive to a massive centralized location before a decision is made. This type of approach relies on networked sensors (called “agents” from here on) to actively collect and process data, and provide key control decisions to significantly improve both the quality/relevance of the collected data and the associating decision making. The technological bottlenecks for such sensor networks stem from a lack of mathematics and algorithms to manage the systems, rather than difficulties associated with building and deploying them.

Indeed, traditional sensor coordination strategies do not provide adequate solutions for this problem. Passive data collection methods (e.g., large sensor webs) can scale to large systems, but are generally not suited to highly dynamic environments, such as advanced energy systems, where crucial decisions may need to be reached quickly and locally. Approaches based on local decisions on the other hand cannot guarantee that each agent performing its task (maximize an agent objective) will lead to good network wide solution (maximize a network objective) without invoking cumbersome coordination routines. There is currently a lack of algorithms that will enable self-organization and blend the efficiency of local decision making with the system level guarantees of global decision making, particularly when the systems operate in dynamic and stochastic environments.

In this work we addressed this critical gap and provided a comprehensive solution to the problem of sensor coordination to ensure the safe, reliable, and robust operation of advanced energy systems. The differentiating aspect of the proposed work is in shifting the focus towards “what to observe” rather than “how to observe” in large sensor networks, allowing the agents to actively determine both the structure of the network and the relevance of the information they are seeking to collect. In addition to providing an implicit coordination mechanism, this approach allows the system to be reconfigured in response to changing needs (e.g., sudden external events requiring new responses) or changing sensor network characteristics (e.g., sudden changes to plant condition).

Outcome Summary: All milestones associated with this project have been completed. In particular, private sensor objective functions were developed which are aligned with the global objective function, sensor effectiveness has been improved by using “sensor teams,” system efficiency has been improved by 30% using difference evaluation functions, we have demonstrated system reconfigurability for 20% changes in system conditions, we have demonstrated extreme scalability of our proposed algorithm, we have demonstrated that sensor networks can overcome disruptions of up to 20% in network conditions, and have demonstrated system reconfigurability to 20% changes in system conditions in hardware-based simulations. This final report summarizes how each of these milestones was achieved, and gives insight into future research possibilities past the work which has been completed. The following publications support these milestones [6, 8, 9, 10, 16, 18, 19].

2 Aligning Private and Team-Based Agent Objectives to Improve System Efficiency

The following sections detail how objective alignment may be quantified in order to achieve effective self-organization of sensors operating within a larger sensor network.

2.1 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 distributed sources and sensor agents.

2.1.1 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. The sources can offer different types of information where the strength for each type is given by $\nu_{j,t}$, where j indexes the source numbers and t the information types. Similarly, the agents have 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 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 learning, but approximates the way in which some real world types of information propagate through an environment.

The agent drop off function is given by:

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

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 the specify their respective extent. As an example, Figure 2 shows the drop off function for an agent.

Figure 2: Agent drop off function.

2.1.2 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 relationship 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.

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

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

2.2 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 given able to observe the location of sources,

and the strength of information that source provides. Additionally, the agents can observe other agents. These observations comprise the state of the agent, as they do not retain knowledge of their locations to promote generalization across multiple environment configurations.

2.2.1 Environment Observation

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).

Figure 3: Agent observations of the surrounding environment.

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 agent's goal is to learn which type of information is interesting for them to promote high system level performance.

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

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

2.2.2 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 a learning 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 learning 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 [20]. 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 learning 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:

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

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 .

2.2.3 Agent Learning

Training the neural networks occurs with a neuro-evolution search algorithm [12, 15]. 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 learning episode. The shared policy is evaluated only with the global objective, whereas the individual policies are evaluated with two additional learning signals.

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.

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

The local objective $L_i(z)$ is given by Equation 8 and applies only for ranking the learning 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.

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

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 learning signal for agent i to determine the impact it had on system level performance.

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

2.3 Experimental Set-up

A large number of experiments were run to evaluate algorithm and learning objective 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. 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 2.1.1) 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 learning 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 has $2 \cdot TYPES$ 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 policy search was turned off to analyze converged behavior.

2.4 Simulation 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

Figure 4: The result of learning in an environment with 10 agents and 5 sources.

source had at least 2 non-zero type strengths, with the majority of the sources containing all three types. Figure 4 shows the learning 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 policies, they outperform learning a shared policy. The exception here is using the system objective from Equation 7, where the learning signal is too noisy for the agents to determine their contribution to the system as a whole.

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.

These results also show that while using the difference objective for learning outperforms other objectives, learning 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

Figure 5: The maximum system level performance achieved is plotted versus a varying number of agents.

different, and upon examining the policies learned, we can conclude that the difference objective indeed produces a learning signal that closely matches the system level objective (as that is what is used when learning a shared policy). However, learning a shared policy historically does not produce good behavior in stochastic coordination domains.

In the above experiments, the number of agents in the system and the location of the sources did not change over the course of learning (though the agents started in different locations each episode). Further work will determine if learning a shared policy will break down when agents fail or are added to the system, and when sources are added or removed. These results suggest that requiring the agents to develop their own policies is the only method that will be successful in such a dynamic environment, as the policies developed will need to be unique and much more specific to the individual.

3 Improving System Efficiency and Scalability with New Objective Functions

The following sections detail multiple test domains to demonstrate the effectiveness of different sensor objective functions, as well as give algorithms and resulting performance which demonstrates our proposed algorithms improve sensor network accuracy by up to three orders of magnitude.

3.1 Sensor Models

Sensors typically have two main types of error: sensitivity error and noise. Sensitivity error is a constant offset in a sensor's measurement. For example, if a thermocouple measuring the temperature of an object at $0^\circ C$ returned a temperature of $5^\circ C$, then that sensor would be said to have a $+5^\circ C$ sensitivity error. Noise is a random deviation of the signal that varies with time. For example, if a sensor measuring the temperature of an object at $0^\circ C$ returned three separate measurement values of $-1^\circ C$, $3^\circ C$, and $1^\circ C$, then that sensor would be said to have noise in its measurement. In this work, our sensors have both sensitivity error (termed *attenuations*) as well as random error. The error parameters we use for our simulated sensors are based on readily-available thermocouples and pressure sensors, such as those found at [2, 1].

3.2 The Defect Combination Problem (DCP)

Many real world sensing applications require large sets of disparate sensing devices to coordinate their actions in order to collectively optimize their network attenuation, coverage areas, and sensing schedules [11, 17, 22]. In this work, a set of up to 10,000 sensing devices must coordinate their sensing schedules in order to optimize their aggregated attenuation within a sensor network. This work focuses on the Defect Combination Problem (DCP) domain introduced in [7]. This problem assumes that there exists a set of imperfect sensors $\mathbf{X}$ which have constant attenuations due to manufacturing defects or imperfections. Each of the sensors x_i has an associated attenuation a_i (which can be positive or negative) in its reading, such that if it is taking a measurement of A (actual value) it measures $A + a_i$ where a_i is the device's individual error. The problem then becomes how to best choose a subset of the $\mathbf{X}$ sensors that minimizes the aggregated attenuation of the combined readings:

$$G = \frac{\left| \sum_{i=1}^N n_i a_i \right|}{\sum_{i=1}^N n_i} \quad (10)$$

where G is the aggregated attenuation of the combined sensor readings, a_i is the attenuation of a particular sensor i , N is the number of sensors, and $n_i \in \{0, 1\}$ based upon whether the sensor chooses to be "on" or "off".

This is an NP-complete optimization problem [7, 21] and simply choosing the single sensor with the best attenuation is an inadequate solution, as is choosing the best K sensors ($1 \leq K \leq N$). To illustrate this, consider the case where there are 6 sensing devices whose attenuations are $a_1 = -0.19$, $a_2 = 0.54$, $a_3 = 0.1$, $a_4 = -0.14$, $a_5 = -0.05$, and $a_6 = 0.21$. Choosing only the best sensor a_5 would yield an aggregated attenuation of $|0.05|$, while choosing sensors a_3 , a_4 , and a_5 will yield an aggregated attenuation of $|0.03|$, which is better than the single best sensing device a_5 alone. This is still not the optimal solution in this 6 sensor case however, as combining sensors a_1 and a_6 results in an aggregated attenuation of $|0.01|$. In this problem, individual sensors acting independently without coordinating their actions can drastically decrease the system performance. Consider the case where sensors a_1 and a_6 are turned on in conjunction with sensor a_2 , the aggregated attenuation jumps to from $|0.01|$ to $|0.18|$. Finding good solutions requires a great deal of coordination between sensors, as any one sensor can heavily impact the system performance.

3.3 The Time-Extended Defect Combination Problem (TEDCP)

Frequently, distributed sensor networks are comprised of battery-powered sensors which are deployed within an environment to make observations. In such settings, the individual sensing nodes must not only coordinate in order to observe the environment, but they must also coordinate their actions such that they conserve resources such as power (battery life). Here, the majority of battery power is consumed on two key tasks: 1) transmitting information, and 2) taking measurements. The transmission of information can be reduced by applying an organizational structure. Additionally, both the transmission of information as well as the frequency of taking measurements can be reduced for each sensor by having the sensors coordinate their sense/sleep schedules. In general, the goal would be to optimize the tradeoffs between observing the environment and minimizing the number of measurements performed by each individual sensing device.

In the original DCP problem (Equation 10) sensors must choose whether or not to participate in sensing, which involves a decision to be “on” or “off” in order to optimize their aggregated attenuations. We extend this one step further so that sensors must choose not only “if” they want to participate but “when” to participate, adding an additional degree of coordination complexity. Here, each agent may only participate in one of the M time steps per episode. This restriction couples the time slots, making this a significantly harder problem than solving multiple parallel versions of the DCP. In this setting agents try to optimize the average aggregated attenuation over M time slots, where the attenuation of a single time slot m is given by g_m :

$$g_m = \frac{\left| \sum_{i=1}^N n_{i,m} a_i \right|}{\sum_{i=1}^N n_{i,m}} \quad (11)$$

in this equation $n_{i,m} \in \{0, 1\}$ corresponding to whether sensor i chose to be “on” or “off”

during time slot m (if agent i chose to participate during time-slot m , $n_{i,m}$ is 1, otherwise it is 0). Here, g_m represents the resultant attenuation for time slot m (analogous to Equation 10). Averaging the aggregated attenuation over M distinct time slots, the time-extended system objective becomes:

$$G_{TE} = \frac{1}{M} \sum_{m=1}^M g_m \quad (12)$$

where G_{TE} is the time-extended system objective, g_m is the aggregated attenuation of the sensors on time-slot m , and M is the total number of time-slots. Here the objective is to optimize the aggregated attenuation over M time steps.¹ It should be noted that there is no sequence of actions being taken in this case, instead, during each episode of learning agents are choosing which time-slot of the M time-slots they want to participate in as their action.

3.4 Multi-Target Defect Combination Problem

Many real world sensing applications require large sets of disparate sensing devices to coordinate their actions in order to collectively optimize their network attenuation, coverage areas, and sensing schedules [11, 17, 22]. In this domain, a set of sensing devices must coordinate their sensing schedules in order to optimize their aggregated attenuation over a set of targets within an environment. Here, sensing agents are deployed within an environment such that they can each observe a subset of the targets within that environment and the agents must work together in order to collectively optimize the accuracy of their sensor readings over these targets. This domain focuses on the Defect Combination Problem (DCP) introduced in [7] (although similar domains aiming to minimize Bayesian uncertainty in sensor networks for multi-target tracking have also been proposed [13]). The DCP objective function (Equation 10) is extended by considering multiple targets. So, rather than a set of sensors minimizing the aggregate attenuation of the measurements corresponding to a single target, the sensors must collaborate to minimize the aggregate attenuation of many target measurements, where each sensor can only sense a subset of the total set of targets (Figure 6). The objective function for the Multi-Target Defect Combination Problem (MTDCP) is:

$$R = \sum_{i=1}^M \frac{\left| \sum_{j=1}^N n_{j,i} \zeta_j \right|}{\sum_{j=1}^N n_{j,i}} \quad (13)$$

where R is the aggregated attenuation (system reward) of the combined sensor readings over the set of M targets, ζ_j is the attenuation of a particular sensor j , N is the number of sensors, and $n_j \in \{0, 1\}$ is an indicator based upon whether sensor j chooses to be “on” or “off”. The MTDCP is simply a sum of individual instances of the original DCP.

¹Equation 12 reduces to Equation 10 when there is a single time slot ($M = 1$).

Figure 6: The Multi-Target Defect Combination Problem assumes that a set of imperfect sensing devices are scattered throughout an environment containing multiple targets which need to be observed. The sensors each have different attenuations and each sensor can only observe a subset of the targets. The goal of the sensors is to optimize their aggregated accuracy of the observations across all of the targets (Equation 10).

This is an NP-complete optimization problem [7, 21] and simply choosing the single sensor with the best attenuation for each target is an inadequate solution, as is choosing the best K sensors ($1 \leq K \leq N$). To illustrate this, consider the case where there is a single target ($M = 1$) and there are 6 sensing devices whose attenuations are $\zeta_1 = -0.19$, $\zeta_2 = 0.54$, $\zeta_3 = 0.1$, $\zeta_4 = -0.14$, $\zeta_5 = -0.05$, and $\zeta_6 = 0.21$. Choosing only the best sensor ζ_5 would yield an aggregated attenuation of $|0.05|$, while choosing sensors ζ_3 , ζ_4 , and ζ_5 would yield an aggregated attenuation of $|0.03|$, which is better than the single best sensing device ζ_5 alone. This is still not the optimal solution in this 6 sensor case however, as combining sensors ζ_1 and ζ_6 results in an aggregated attenuation of -0.01 . In this problem, individual sensors acting independently without coordinating their actions can drastically decrease the system performance. Consider the case where sensors ζ_1 and ζ_6 are turned on in conjunction with sensor ζ_2 , the aggregated attenuation jumps to from $|0.01|$ to $|0.18|$. Finding good solutions requires a great deal of coordination between sensors, as any one sensor can heavily impact the system performance.

3.5 Description of Algorithm

Agent learning in the Time-Extended DCP, the Multi-Target DCP, and the Rankine cycle DCP is achieved through standard multiagent Q-learning, shown in Algorithm 1. Each agent maintains a private Q-table, and updates the Q-table at each time step based on the reward that the agent receives.

The following sections introduce the reward structures utilized in this work, as well as their derivations in the Rankine cycle DCP.

Algorithm 1 Reinforcement Learning Algorithm

Each agent i generates a randomly seeded Q-table Q_i ;
 $episode = 1$;
while $episode < maxEpisodes$ **do**
 1. Each agent i selects an action from Q-table using ϵ -greedy;
 2. Calculate system objective based on joint action;
 3. Calculate rewards R_i for each agent;
 4. Q-update: $Q_i(a) \leftarrow Q(a)(1 - \alpha) + \alpha R_i$;
 5. $episode = episode + 1$;
end while

3.5.1 Difference Reward

The Difference Reward $D_i(z)$ is defined as [21]:

$$D_i(z) = G(z) - G(z_{-i}) \quad (14)$$

where $G(z)$ is the global reward, and $G(z_{-i})$ is the global reward without the influence of agent i . Intuitively, the Difference Reward gives agent i 's specific impact on the system performance. Note that:

$$\frac{\partial D_i(z)}{\partial a(i)} = \frac{\partial G(z)}{\partial a(i)} \quad (15)$$

where $a(i)$ is agent i . Thus, an agent acting to increase the Difference Reward will also act to increase the global reward. This property is termed *factoredness* [5]. Further, because the Difference Reward only depends on the actions of agent i , noise from other agents is reduced in the feedback given by D_i . This property is termed *learnability* [5].

An extension of the Difference Reward, the Expected Difference Reward $ED_i(z)$, is defined as in [21]:

$$ED_i(z) = G(z) - E_i(a)[G_z] \quad (16)$$

where $E_i(a)[G(z)]$ is the expected value of the global reward across all actions that agent i may take. In the case of a discrete action space, Equation 16 becomes:

$$ED_i(z) = G(z) - \sum_{a \in A} P_i(a) G_i(z_a) \quad (17)$$

where $P_i(a)$ is the probability that agent i takes action a , and $G_i(z_a)$ is the global reward when agent i takes action a . While the Difference Reward gives the impact of agent i on the global reward, the Expected Difference Reward gives the expected impact of agent i on the global reward. The difference reward has been shown to promote good learned policies in many domains, including rover coordination [5] and distributed sensor network control [21].

3.5.2 Derivation of Rewards for DCP

First, we derive the D and EDR shaped rewards for the standard *DefectCombinationProblem* problem. Here, each agent is required to coordinate directly with all other agents in the system:

$$D_j = \begin{cases} \frac{\left| \sum_{i=1}^N n_i a_i \right|}{\sum_{i=1}^N n_i} - \frac{\left| \sum_{i \neq j}^N n_i a_i \right|}{\sum_{i \neq j}^N n_i}, & \text{if } n_j = 1 \\ 0, & \text{if } n_j = 0 \end{cases} \quad (18)$$

D_j only provides agents with a clear learning signal if $n_j \neq 0$. If an agent chooses $n_j = 0$, it receives a reward of 0 whether that action was good or bad for the system performance, meaning half of the actions an agent takes will effectively be random as far as the system performance is concerned.

Next, we derived the Expected Difference Reward for the basic Defect Combination Problem. We consider the case where the the probabilities are equivalent for each action “on” and “off”, $P_{n_j=0} = 0.50$ and $P_{n_j=1} = 0.50$, EDR_j becomes:

$$EDR_j = \begin{cases} 0.50 \frac{\sum_{i \neq j}^N n_i a_i - a_j}{\sum_{i \neq j}^N n_i - 1} - 0.50 \frac{\sum_{i=1}^N n_i a_i}{\sum_{i=1}^N n_i}, & \text{if } n_j = 1 \\ 0.50 \frac{\sum_{i \neq j}^N n_i a_i + a_j}{\sum_{i \neq j}^N n_i + 1} - 0.50 \frac{\sum_{i=1}^N n_i a_i}{\sum_{i=1}^N n_i}, & \text{if } n_j = 0 \end{cases} \quad (19)$$

EDR_j provides a clear learning signal: if it is positive, the action taken by agent j was beneficial to system performance, and if EDR_j is negative, the action was harmful to system performance. Agents trying to maximize EDR_j will implicitly maximize system performance simultaneously.

Deriving the Difference Reward and Expected Difference Reward for the Time-Extended DCP and the Multi-Target DCP is a simple extension of Equations 18 and 19. For the Time-Extended DCP, the reward is the sum of the rewards at each time step. For the Multi-Target DCP, the reward is the sum of rewards of each target a particular agent is involved in sensing.

3.6 DCP Experiments

3.6.1 Standard DCP

The following experiments show the performance of our multiagent learning algorithm in the standard DCP. We compare the performance of the Difference Reward D , the Expected Difference Reward EDR , The Global Reward G , and the single best sensor CBS . The aggregate measurement error for 100, 1,000, and 10,000 sensors are shown in Figure 7, 8, and 9, respectively.

Figure 7: DCP performance with 100 agents

Figure 8: DCP performance with 1,000 agents

The first set of experiments shows the performance of agents solving the DCP problem. Here, each agent must coordinate directly with all other agents in the system. Agents using a global reward G turn on approximately half of the sensing devices and make better decisions selecting which sensors to turn on. However, agents using G still have difficulty differentiating the impact of their own actions on their reward signal from the actions of other agents (in this setting, each agent’s reward signal is directly impacted by the actions of all other agents). This is because with G , all agents receive the system performance as their reward signal, regardless of how their own actions impacted the system performance. This makes it difficult for these agents to coordinate their actions, inhibiting system performance.

Difference rewards, D , and Expected Difference Rewards, EDR , address this shortcoming by effectively filtering out the impact of other agents on an agents’ reward signal and accounting for each agent’s individual contribution to the system performance. When relatively few agents are present in the system, agents using D outperform many other methods (Figures 7 and 8).

Figure 9: DCP performance with 10,000 agents

However, as seen in Figure 9, when scaling is increased agents using D perform poorly in this experiment. This is because D only provides constructive feedback when the agent elects to be active. Agents using D receive a reward of 0 when they choose to remain off so they do not receive enough feedback to successfully coordinate their actions. This is because if an agent using D initially harms the system and gets a negative reward, it chooses to remain “off” a majority of the time (except for a small amount of exploration). If the agent continues to get negative rewards for being turned on while it explores, it will be hard to overcome. This is because while the value of being turned off remains at 0, the negative rewards for being turned on effectively “stack” and discourage the agent from turning on. It would take a significant amount of “positive” reward to make up for these initial negative rewards. One solution to this problem would be to decrease the value of

the learning rate α , which would effectively truncate the effective build up of “negative” rewards on the value.

On the other hand, the Expected Difference Reward structure, *EDR*, does not give a 0 reward to an agent for being on or off, instead it gives an estimated value of the agents cumulative impact on the system over time based upon its historic action selections, resulting in better performance than *D* in this case (Figures 7-9). As seen, in this setting *EDR* significantly outperforms all other rewards. Figure 10 shows the converged aggregate measurement errors as a function of the number of sensors in the system.

Figure 10: DCP performance: scaling plot

As seen in Figure 10, as the number of sensors in the system is increased, the performance of the sensor network using *EDR* actually increases. This is because the addition of more sensors gives more possible combinations of sensors, which allows for better solutions to be found. This result indicates that not only can our algorithm handle extreme scaling of the system, it actually benefits because there are more devices to coordinate, which allows for better solutions to be found.

3.6.2 Time-Extended DCP

The next set of experiments show the performance of our multiagent learning algorithm in the Time-Extended DCP. Figure 11 shows the aggregate measurement error for 10,000 sensors. Figure 12 shows the scaling performance in the Time-Extended DCP.

As seen in Figure 11, agents using *EDR* provide over two orders of magnitude more accurate measurements than agents using *G*, and perform better than any method tested. With 10,000 agents, agents using *D* perform worse than the single best sensor. This is again due to the limitations of *D* discussed in Section 3.6.1. However, in the more complex problem including a time-extension, *EDR* performs well even in a system with 10,000 sensing devices.

Figure 11: Time-Extended DCP performance with 10,000 agents

Figure 12: Time-Extended DCP performance: scaling

As seen in Figure 12, as the number of sensors in the systems increases, the performance of the network trained using *EDR* also increases. Again, this is due to the fact that more sensors allows for more possible combinations, which results in an increase in feasible performance of the distributed sensor network. These experiments further reinforce the notion that our algorithm is well-equipped to deal with extreme scalability situations.

3.6.3 Multi-Target DCP

In this set of experiments in the MTDCP domain we have $N = 10,000$ sensing agents, $T = 100$ targets, and each agent is connected to $C_j = 10$ targets randomly. Agents using traditional global rewards G perform poorly within this setting. This is because agents struggle to coordinate their actions due to noise on their learning signals caused by the actions of other agents. Figure 13 shows system performance in the Multi-Target DCP with 10,000 agents, and Figure 14 shows scaling performance in the Multi-Target DCP.

As seen in Figures 13 and 14, agents using *EDR* significantly outperform other methods. This is because as multiagent systems become increasingly large, the coordination problem becomes increasingly difficult. In this setting, Expected Difference Rewards reduce the coordination requirements of agents, resulting in improved coordination. These results show that *EDR* is robust to scaling and is beneficial for systems of ranging sizes (Figure 14). These results further show the ability for the distributed sensor network to compensate for extreme scalability situations.

Figure 13: Multi-Target DCP performance with 10,000 agents

Figure 14: Multi-Target DCP performance: scaling

4 Reconfigurability for Changes in Environmental and Network Conditions

The following sections detail a hardware based simulator, as well as experiments which demonstrate the sensor network can successfully reconfigure itself after large changes in environmental or network conditions in order to continually provide accurate measurement data.

4.1 Rankine Cycle Defect Combination Problem

The following sections introduce the Rankine cycle and the modified DCP for use in a steam-cycle power plant. This model is used to create the hardware-based simulation, which will be used to determine reconfigurability performance of the distributed sensor network.

4.1.1 Rankine Cycle

We develop a sensor network in a well known power generation system: a vapor power Rankine cycle [14]. The Rankine cycle is one of the simplest thermodynamic power-producing cycles, and serves as a testbed to demonstrate the effectiveness of our approach. It is important to note that although the Rankine cycle is not complex enough to demonstrate effectiveness of control algorithms, it is adequate to demonstrate the effectiveness of our sensor network training algorithm. Further, the Rankine cycle represents a hardware-based power plant simulation, as we deal with all the typical components (e.g. boiler, turbine, condenser, pump) that we see in steam cycle power plants. In our approach, the model is treated as a black box, and only the output of the model is used to

Figure 15: A vapor power Rankine cycle. The working fluid travels through a boiler, turbine, condenser, and pump in succession. The work output of the turbine is used to generate electricity.

train sensors operating within that model. In a Rankine cycle, the working fluid passes through a boiler and becomes saturated vapor. Next, the fluid goes through the turbine, which results in an energy output which is used to produce electricity. The fluid then passes through a condenser and becomes a saturated liquid. Finally, the fluid passes through a pump and returns to the boiler, completing the cycle. The Rankine cycle is shown in Figure 15. For the purposes of this analysis, we make the following assumptions:

- A1.** Each component of the cycle is considered to be a control volume.
- A2.** All processes of the working fluid are internally reversible.
- A3.** The turbine and pump operate adiabatically.
- A4.** Kinetic and potential energy effects are negligible.
- A5.** Saturated vapor enters the turbine. Condensate exits the condenser as a saturated liquid.
- A6.** The working fluid is water.

As seen in Figure 15, there are four distinct states in the Rankine cycle, each of which lies between two of the components. At each state, the working fluid has an enthalpy h , which is a thermodynamic value indicating the energy stored in the working fluid. The enthalpy of a fluid is a function of temperature and pressure. The system performance is

related to the enthalpy h_s at each plant state s by the following relations:

$$\frac{\dot{W}_t}{\dot{m}} = h_1 - h_2 \quad (20)$$

$$\frac{\dot{Q}_{out}}{\dot{m}} = h_2 - h_3 \quad (21)$$

$$\frac{\dot{W}_p}{\dot{m}} = h_4 - h_3 \quad (22)$$

$$\frac{\dot{Q}_{in}}{\dot{m}} = h_1 - h_4 \quad (23)$$

where $\dot{m}$ is the mass flow rate of the working fluid, $\dot{W}_t$ is the work output of the turbine, $\dot{Q}_{out}$ is the heat output of the condenser, $\dot{W}_p$ is the work input to the pump, and $\dot{Q}_{in}$ is the heat input to the boiler. In order to evaluate these relations, the enthalpy of the working fluid at each state must be determined by measuring the temperature and pressure at each state; this requires the development of a sensing policy. Such a policy was studied in the Defect Combination Problem.

4.1.2 DCP For Power Plants

We apply a modified version of the DCP to a Rankine cycle power plant. There is a set of motes X_s at each of the four plant states, where $s \in \{1, 2, 3, 4\}$ is the state of the power plant (see Figure 15). Each mote $x_{s,i} \in X_s$ has sensors capable of measuring temperature and pressure, the two parameters needed to determine the enthalpy of the working fluid. The sensors in mote $x_{s,i}$ have a mean temperature attenuation $t_{s,i}$, and a mean pressure attenuation $p_{s,i}$. Further, each sensor has an associated measurement noise defined by the Gaussian distribution, where σ_t and σ_p are the standard deviations for temperature and pressure attenuations respectively. Thus, the temperature and pressure attenuations of the sensors on each mote are given by the following normal distributions:

$$e_{T,s,i} = N(\sigma_t, t_{s,i}) \quad (24)$$

$$e_{P,s,i} = N(\sigma_p, p_{s,i}) \quad (25)$$

Each mote is considered to be an agent. First, an agent decides whether to be “on” or “off.” If an agent decides to be “on,” then it must decide if it will measure temperature, pressure, or both temperature and pressure. The goal of the agents is to collectively take actions which will minimize the aggregate error in temperature and pressure readings. The aggregate attenuation for temperature at a state s is defined as:

$$g_{T,s} = \frac{\sum_{i=1}^{N_s} n_{s,i} e_{T,s,i}}{\sum_{i=1}^{N_s} n_{s,i}} \quad (26)$$

where N_s is the number of motes in state s , and $n_{s,i} \in \{0, 1\}$ denotes whether mote $x_{s,i}$ is measuring temperature or not. Similarly, the aggregate attenuation for pressure at state s is defined as:

$$g_{P,s} = \frac{\sum_{i=1}^{N_s} n_{s,i} e_{P,s,i}}{\sum_{i=1}^{N_s} n_{s,i}} \quad (27)$$

where $n_{s,i} \in \{0, 1\}$ denotes whether mote $x_{s,i}$ is measuring pressure or not. From Equations 26 and 27, the measured values of temperature and pressure at state s are:

$$T_{s,sensed} = T_s + g_{T,s} \quad (28)$$

$$P_{s,sensed} = P_s + g_{P,s} \quad (29)$$

where T_s and P_s are the true temperature and pressure at state s , respectively. Equations 26 and 27 can not be used to provide feedback to learning agents, because they can not be calculated directly in real-world applications, because the attenuation of each sensor is not known. However, using the system model and knowledge of the control inputs, the enthalpy at each state may be analytically determined. Thus, the enthalpy found from the sensor readings may be compared with the true enthalpy (found with system model) to determine the accuracy of the sensor network.

The enthalpy of the working fluid is a thermodynamic property which quantifies the level of energy in the fluid. Enthalpy change in a fluid corresponds to the fluid either absorbing or expelling energy, and is used to determine power levels in a power cycle. In the Rankine cycle power plant, the control inputs are $\dot{Q}_{out}$, $\dot{W}_p$, $\dot{Q}_{in}$, and $\dot{m}$, and are known values. Thus, using Equations 20 through 23 in addition to the assumptions made about the Rankine cycle, the enthalpy values h_1 through h_4 may be directly determined. The enthalpy at each state is also estimated by the sensor network, where the estimation of enthalpy is defined by:

$$h_{s,sensed} = f(T_{s,sensed}, P_{s,sensed}) \quad (30)$$

where $f(T, S)$ is the enthalpy equation based on thermodynamic empirical data, and $T_{s,sensed}$ and $P_{s,sensed}$ are the sensed temperature and pressure values, as defined in Equations 28 and 29. The error in the enthalpy reading at a given state is thus:

$$h_{s,error} = |h_s - h_{s,sensed}| \quad (31)$$

where h_s is the true enthalpy of state s , found using the Rankine cycle model. The error in enthalpy gives an indication of the effectiveness of the sensors measuring temperature and pressure. The objective of the entire sensor network is to minimize the total attenuation of enthalpy measurements at each state, which is equivalent to maximizing:

$$G(z) = - \sum_{s=1}^4 h_{s,error} \quad (32)$$

The key difference between this approach and the DCP is the fact that the objective function given by Equation 32 uses data which is readily available in order to judge sensor efficacy. Recall that the DCP objective function (Equation 10) includes individual sensor attenuations, which are extremely impractical to obtain, especially as the size of the sensor network grows. Thus, the modification we have made to the DCP allows for implementation in real-world applications.

4.2 Derivation of Rewards for Rankine Cycle DCP

The following sections derive each reward (D , ED , AD , and G) used for learning in the Rankine Cycle DCP.

4.2.1 Global Reward

The simplest approach to solving the learning problem is to simply use the Global Reward to provide feedback to each agent. Although this method is simple, it is an ineffective approach. As the number of sensors in the network grows, the Global Reward provides poor agent-specific impact. For example, in a sensor network with 1000 motes, the Global Reward provides poor feedback on the action selection of one particular agent, because of the small impact on G that one agent has. For this reason, it is not expected that using the Global Reward for feedback will provide satisfactory results, but will provide a good baseline for comparison of our shaped rewards D and ED .

4.2.2 Difference Reward

We will now derive the Difference Reward for the Rankine cycle DCP. Recall from Equation 14, the Difference Reward is defined as:

$$D_i(z) = G(z) - G(z_{-i}) \quad (33)$$

There are four actions $a \in A$ that each agent may take in the Rankine cycle DCP, where $A = \{a_1, a_2, a_3, a_4\}$, and:

- a_1 : sense nothing
- a_2 : sense pressure
- a_3 : sense temperature
- a_4 : sense both temperature and pressure

Removing an agent from the system is equivalent to having it sense nothing, or to take action a_1 . The aggregate attenuation for temperature at a state s without the impact of agent i is:

$$g_{T,s,-i} = \frac{\sum_{j=1, j \neq i}^{N_s} n_{s,j} e_{T,s,i}}{\sum_{j=1, j \neq i}^{N_s} n_{s,j}} \quad (34)$$

The aggregate attenuation for pressure at a state s without the impact of agent i is:

$$g_{P,s,-i} = \frac{\sum_{j=1, j \neq i}^{N_s} n_{s,j} e_{P,s,j}}{\sum_{j=1, j \neq i}^{N_s} n_{s,j}} \quad (35)$$

Thus, the temperature and pressure measurements at any state s without the impact of agent i are given by:

$$T_{s,sensed,-i} = T_s + g_{T,s,-i} \quad (36)$$

$$P_{s,sensed,-i} = P_s + g_{P,s,-i} \quad (37)$$

The estimated enthalpy at state s without the effects of agent i is given by:

$$h_{s,sensed,-i} = f(T_{s,sensed,-i}, P_{s,sensed,-i}) \quad (38)$$

The error in the enthalpy measurement without the effects of agent i is:

$$h_{s,error,-i} = |h_s - h_{s,sensed,-i}| \quad (39)$$

The total system reward without the effects of agent i is:

$$G(z_{-i}) = - \sum_{i=1}^4 h_{s,error,-i} \quad (40)$$

The Difference Reward for the Rankine cycle DCP is thus:

$$D_i(z) = G(z) - G(z_{-i}) \quad (41)$$

where $G(z)$ is defined in Equation 32 and $G(z_{-i})$ is defined in Equation 40.

4.2.3 Expected Difference Reward and Average Difference Reward

We will now derive the Expected Difference Reward for the Rankine cycle DCP. Recall that the Expected Difference Reward is defined by:

$$ED_i(z) = G(z) - \sum_{a \in A} P_i(a) G_i(z_a) \quad (42)$$

In order to calculate the Expected Difference Reward for some agent i , we must determine the global reward when agent i selects any action $a \in A$. We begin by defining temperature and pressure indicator functions based on the action selected:

$$n_t(a) = \begin{cases} 0, & a = a_1 \\ 0, & a = a_2 \\ 1, & a = a_3 \\ 1, & a = a_4 \end{cases} \quad (43)$$

$$n_p(a) = \begin{cases} 0, & a = a_1 \\ 1, & a = a_2 \\ 0, & a = a_3 \\ 1, & a = a_4 \end{cases} \quad (44)$$

The aggregate attenuation for temperature at state s when agent i takes action a is:

$$g_{T,s,i}(a) = \frac{\left[\sum_{j=1, j \neq i}^{N_s} n_{s,j} e_{T,s,j} \right] + n_t(a) e_{T,s,i}}{n_t(a) + \sum_{j=1, j \neq i}^{N_s} n_{s,j}} \quad (45)$$

The aggregate attenuation for pressure at state s when agent i takes action a is:

$$g_{P,s,i}(a) = \frac{\left[\sum_{j=1, j \neq i}^{N_s} n_{s,j} e_{P,s,j} \right] + n_p(a) e_{P,s,i}}{n_p(a) + \sum_{j=1, j \neq i}^{N_s} n_{s,j}} \quad (46)$$

The temperature and pressure measurements at state s when agent i takes action a are thus:

$$T_{s,sensed,i}(a) = T_s + g_{T,s,i}(a) \quad (47)$$

$$P_{s,sensed,i}(a) = P_s + g_{P,s,i}(a) \quad (48)$$

The estimated enthalpy at state s when agent i takes action a is:

$$h_{s,sensed,i}(a) = f(T_{s,sensed,i}(a), P_{s,sensed,i}(a)) \quad (49)$$

The error in the enthalpy measurement at state s when agent i takes action a is:

$$h_{s,error,i}(a) = |h_s - h_{s,sensed,i}(a)| \quad (50)$$

The total system reward when agent i takes action a is thus:

$$G_i(z_a) = - \sum_{s=1}^4 h_{s,error,i}(a) \quad (51)$$

The probabilities in the Expected Difference Reward are based on previous actions taken by each agent. Initially, each probability $P(a)$ is set to $1/|A|$, where $|A|$ is the cardinality of the set A . During learning, each agent keeps a record of the number of times it has taken every action. These records allow for direct calculation of the probabilities. Suppose that agent i has taken action a_k a total of c_k times, then the probability of agent i taking a particular action a_j is given by:

$$P_i(a_j) = \frac{c_j}{\sum_{k=1}^4 c_k} \quad (52)$$

With the probabilities defined, we can now calculate the Expected Difference Reward as:

$$ED_i(z) = G(z) - \sum_{j=1}^4 P_i(a_j) G_i(z_{a_j}) \quad (53)$$

We can define a similar reward, the Average Difference Reward, which is equivalent to the expected difference reward when the probabilities of taking each action are assumed to be constant:

$$AD_i(z) = G(z) - \frac{1}{4} \sum_{j=1}^4 G_i(z_{a_j}) \quad (54)$$

5 Hardware-Based Simulation Analysis

In order to demonstrate the efficacy of the sensor network in real-world applications, a hardware-based simulator is used to test reconfigurability of the network for up to 20% change in system conditions. In order for a sensor network to be a viable option for

use in an advanced power system, it must be able to overcome changes in the system such as sensitivity error, measurement noise, sensor failure, or sudden changes in system conditions.

A distributed sensor network must be able to account for measurement noise in individual sensors and determine an accurate measurement value based on different measurements from multiple sensors. Typically, sensors have sensitivity error and noise in their measurements [2]. Sensitivity error is a constant offset in a sensor’s measurement. For example, a sensor which always measures temperature to be 1°C higher than the actual temperature has a sensitivity error of 1°C . Noise is random error, which typically follows a Gaussian distribution. For example, a temperature sensor having no sensitivity error and measuring an object with a temperature of 100°C will return measurements close, but not equal to 100°C . The following sections detail results in the Rankine cycle hardware-based simulation (Section 4.1.1) for sensor networks with sensitivity error, measurement noise, sensor failure, and sudden changes in system conditions.

5.1 Sensitivity Error

For this experiment, a sensor network consisting of 1000 sensors is implemented in the hardware-based simulation, where each sensor has some level of sensitivity error. The sensitivity error of each individual temperature sensor was drawn from a uniform random distribution varying between -10°C and 10°C for temperature sensors, while the sensitivity error of each individual pressure sensor was drawn from a uniform random distribution varying between -50Pa and 50Pa . These sensitivity errors are much higher than we would expect to see from any realistic sensors, so showing that the sensor network provides adequate performance under these conditions shows that the sensor network can be implemented in hardware and properly account for sensitivity error. The results for these experiments are shown in Figure 16. As seen in the figure, the variants of the Difference Reward provide much better performance than the Global reward or a random sensor network. The measurement error of the sensor network is on the order of 10^{-5} kJ/kg , which is an extremely accurate measurement suitable for process control in an advanced power system.

5.2 Sensitivity Error and Measurement Noise

Although the results from Section 5.1 give promising results for sensor networks with sensitivity error, this is not enough to show the real-world applicability of the distributed sensor network operating in an advanced power system. We now extend these results by adding random measurement noise to the sensors, while retaining the sensitivity error. We implement a sensor network with 1000 sensors in the hardware-based simulator. Each sensor has sensitivity error as defined in Section 5.1. If a sensor has a sensitivity error of a , then when it measures a value A , it returns a measurement $A + a$. In addition to this sensitivity error, a random measurement noise is drawn from a Gaussian distribution each time a sensor takes a measurement. Thus, the sensor measuring A will return the value $A + N(\sigma, a)$, where $N(\sigma, a)$ is a normal distribution with a mean of a and a standard

Figure 16: Hardware-based DCP performance with 1000 agents and sensitivity error

deviation of σ . For the temperature sensors, σ was set to 10°C ; for the pressure sensors, σ was set to 20Pa . Adding random noise to the sensor measurements is critical to show the real-world applicability of the sensor network. If a real sensor takes multiple measurements of the same value, the sensor will typically not return the exact same measurement each time, as sensor noise will affect each measurement.

The results of the sensor network operating in the hardware-based simulation with sensors having sensitivity error and measurement noise are shown in Figure 17. As seen in the figure, the variants of the Difference Reward all significantly outperform the Global Reward and a random sensor network. It is of note that when only sensitivity error was present, the Expected Difference Reward outperformed the Difference Reward and the Average Difference Reward. However, with the addition of random measurement noise, the variants of the Difference Reward perform almost identically. This is because the addition of measurement noise slightly distorts the probability distributions used by the Average and Expected Difference Rewards, resulting in them performing similarly to the Difference Reward.

5.3 Sensor Failure and Measurement Error

The sensor network has been shown to be capable of accounting for sensitivity error and random measurement noise. The final step in showing the sensor network is adequate for implementation in an advanced power system is to show that the sensor network can adequately handle sensor failures. In large sensor networks (1000 sensors or more), sensor failure is inevitable. One of the key advantages to using distributed sensor networks is redundancy in measurements. If sensors fail, other sensors are still in operation and can adjust their policies to account for the reconfiguration of the system.

In this experiment, we take the sensor network with sensitivity error and random measurement noise from Section 5.2, and have 20% of the sensors fail during operation. It is highly unlikely that 20% of the sensors in a network would fail, so showing the network

Figure 17: Hardware-based DCP performance with 1000 agents with sensitivity error and 20% measurement noise

can reconfigure to such a drastic change in system conditions provides evidence that the network can effectively operate in an advanced power system. We train the network in the same manner as in Section 5.2, but after 1000 episodes of training, 20% of the sensors fail and the network continues training to reconfigure to the new system conditions. The results for this experiment are shown in Figure 18. As expected, the variants of the Difference Reward significantly outperform the Global Reward and a random sensor network.

An especially interesting result is that after the system reconfigures to the 20% change in system conditions, there is almost no difference in performance from the sensor network prior to the sensor failures. This means that the system was able to reconfigure to a 20% change in the system with no loss of performance. In contrast, the Global Reward and random sensor network both experience a permanent loss in performance after 20% system failure. The fact that the Difference Rewards allow for reconfigurability to such a large change in system conditions without any permanent loss of performance is strong evidence that a distributed sensor network trained with Difference Rewards can adequately operate in an advanced power system.

5.4 Changing System Conditions

In Sections 5.1-5.3, we show that our algorithm can train highly accurate sensor networks which can account for sensitivity error, the combination of sensitivity error and measurement noise, and can successfully reconfigure after 20% changes to the system conditions. Now, we will analyze the performance of the sensor network operating in a test case run in our hardware-based simulation.

For this experiment, we train a sensor network with sensitivity error and measurement noise, as in Section 5.2. We then have the sensor track the temperature at the boiler outlet during startup. We add artificial oscillations to the temperature at the boiler outlet to demonstrate that the sensor network can track constantly changing system parame-

Figure 18: Hardware-based DCP performance with 1000 agents with sensitivity error and 20% measurement noise with 20% sensor failure

ters during operation in a power system. The temperature tracking of the boiler outlet is shown in Figure 19, showing the difference between the true temperature and the temperature measured by a sensor network trained using the Difference Reward. As seen in the figure, there is no discernible difference between the two temperature profiles.

Figure 20 shows the tracking errors of networks trained using the Average Difference Reward and the Global Reward. As seen in the figure, the network trained with the Expected Difference Reward significantly outperforms the network trained with the Global Reward. Further, after 20% sensor failure, the network using the Expected Difference Reward reconfigures itself faster than the network using the Global Reward, and recovers more performance after reconfiguration than the network using the Global Reward. This shows that the Difference Reward can train a sensor network to take extremely accurate measurements, as well as quickly reconfigure itself without significant loss in performance after large changes to the system conditions.

5.5 Hardware-Based Simulations Summary

The previous experiments demonstrate the ability of our multiagent learning algorithm to provide extremely accurate measurements in hardware-based simulations, as well as successfully reconfigure itself without loss of performance after large (20%) changes to the system conditions.

Section 5.1 show that the sensor network can achieve extremely accurate measurements with sensors having large sensitivity errors. Section 5.2 show that the sensor network can achieve extremely accurate measurements with sensors having large sensitivity errors and random measurement noise. Section 5.3 show that the sensor network can successfully reconfigure itself after 20% changes in system conditions.

Finally, Section 5.4 shows a test case where the sensor network successfully tracks changing system parameters and reconfigures itself without significant loss of perfor-

Figure 19: Hardware-based DCP performance with 1000 agents. Agents have sensitivity and measurement error, and 20% of the sensors fail during the simulation.

Figure 20: Tracking error of hardware-based DCP performance with 1000 agents. Agents have sensitivity and measurement error, and 20% of the sensors fail during the simulation.

mance after 20% changes to system conditions in a hardware-based simulation. These results show that our multiagent learning algorithm can provide an extremely accurate sensor network which can successfully reconfigure itself to changing system conditions, whether the changes in system conditions are system parameters changing or large failures in the sensor network itself.

6 Project Summary

Collecting accurate system level information is a critical step in ensuring the safe, efficient, and reliable operation of advanced power systems. Recent advances in sensor hardware allow for pushing some level of decision making to computationally sophisticated sensors, reducing the computational load on higher level system controllers. With the added increase in hardware capabilities of sensor suites, new algorithms and software must be developed in order to take advantage of new computational capabilities. This work addresses developing such algorithms by casting each computational device as an agent, and extending multiagent learning techniques into the field of computationally sophisticated sensor coordination. The differentiating aspect of this work is in allowing agents to actively determine both the structure of the network and the relevance of information they are seeking to collect. A summary of this work is given below:

Milestone 1: When dealing with multiple agents which must cooperate, it is imperative that private objective functions given to each agent are aligned with the overall system objective function. Each agent aims to maximize its private objective function, so it is critical that maximizing private objective functions leads to maximizing the overall system objective function, or agents will act in ways which reduce overall system performance. By ensuring that objective functions are *factored* with respect to the system objective function, agents maximizing their private objective functions will act to improve system performance. The Difference Reward (Section 2), the Expected Difference Reward (Section 3), and the Average Difference Reward (Section 4) are all perfectly factored, meaning that any action an agent takes to improve its private objective function will also improve overall system performance.

Milestone 2: As sensor networks become larger, team formation has an important role in improving system performance. Coordination becomes increasingly difficult as the number of sensor increases; however, by splitting sensors into different “sensor teams,” each agent within a specific team can coordinate in order to maximize that team’s performance. Further, each team maximizing its own performance also maximizes overall system performance. Section 2 details how sensor teams can improve sensor performance in an abstract simulation. In Section 4, sensor teams were created based on the locations of sensors operating in a power plant simulation. Milestone 2 is very closely related to milestone 1, as agent objective functions must be aligned with the team objective function, and team objective functions must be aligned with the overall system objective function. Again, variants of the difference reward ensure perfect factoredness, allowing for agents to act in ways which improve overall system performance.

Milestone 3: Variants of the Difference Reward (Standard, Expected, and Average) were all utilized in the Defect Combination Problem as well as the Rankine Cycle Defect Combination Problem (Sections 3 and 4). Difference Rewards result in sensor network measurements which are up to three orders of magnitude more accurate than global or local

rewards. Thus, our new sensor objective functions improved system efficiency by well over 30%, the goal set by milestone 3.

Milestone 4: For a sensor network to be successful, it must be capable of reconfiguring itself after drastic changes in environmental conditions. If a sensor network only operates nominally in optimal system conditions, it will fail when unexpected environmental changes occur. This is an unacceptable result, because accurate and effective system measurements are critical for plant controllers to make informed and effective control decisions when environmental conditions unexpectedly change. In Section 4, we demonstrate that the sensor network can quickly reconfigure itself to changing system conditions, allowing for uninterrupted flow of accurate sensor measurements even while the plant environment is rapidly changing in a nonlinear manner.

Milestone 5: As advanced power systems become larger and more complex, larger sensor networks will be required to provide optimal plant coverage. It is imperative that sensor network algorithms can scale to larger networks. In Section 3, we demonstrate that our algorithm can effectively train sensor networks with up to 10,000 coordinating devices, while ensuring that the system still provides accurate measurements and is capable of reconfiguration to changing environment and network conditions.

Milestone 6: In any sensor network with numerous devices, it is inevitable that some portion of sensors will fail during operation. In order to continually provide reliable measurement data and ensure safe and reliable plant operation, the sensor network must be capable of quickly reconfiguring itself in order to address changing network conditions. By casting each sensor as an agent in a multiagent system, the sensor network becomes extremely robust to sensor failures. Sections 3 and 4 demonstrate that even with up to 20% failure in the sensor network, accurate and reliable measurements are still obtainable. In fact, after reconfiguration, there is a minimal loss of sensor network fidelity; after 20% of sensor failed in the network, less than 1% of measurement accuracy was lost.

Milestone 7: After demonstrating the ability of our algorithm to provide robust, scalable, and reconfigurable sensor networks, we developed a hardware-based simulator to demonstrate that these network properties are attainable in real-world systems. A vapor cycle power plant simulation was created (Section 4), and our experiments demonstrate that the sensor network is capable of reconfiguring itself to 20% changes in environmental or network conditions while maintaining extremely accurate measurements. These experiments extend results from milestones 1-6 into a hardware-based simulation, demonstrating the efficacy of our approach in advanced power systems.

Future Work: Completion of this project has opened multiple avenues of future research. First, we can investigate how to best combine these sensor networks with plant controllers. As the key responsibility of a sensor network is to provide relevant information to plant controllers, it is critical to analyze how to ensure that controllers and sensor

suites operate together harmoniously. Second, we can implement our sensor network algorithms on more complex plant environments. For example, we could implement our sensor network in a HyPer simulation, demonstrating the effectiveness of our algorithms on systems currently being researched by the NETL. Finally, we could look at extending our multiagent learning algorithms to plant control. As advanced power systems become more complex, modeling these systems with physics becomes difficult or impossible, because of the extremely complex interactions between various plant subsystems. However, individual subsystems could be controlled independently. This type of control is parallel to how we train sensor networks with multiagent learning algorithms. Each subsystem of the plant would be controlled by an agent, and the agents would need to learn to coordinate such that controlling each subsystem independently leads to optimal control of the overall plant.

7 Cost Reporting

Table 1: Cost Summary (Original Grant plus Extension)

Quarter	Actual Expense Cumulative (in \$)	Projected Expense Cumulative (in \$)
Q4 2009	6,754	28,000
Q1 2010	36,487	70,000
Q2 2010	92,569	132,000
Q3 2010	133,297	198,000
Q4 2010	173,164	244,000
Q1 2011	192,208	290,000
Q2 2011	263,359	352,000
Q3 2011	290,049	434,000
Q4 2011	328,588	486,000
Q1 2012	441,323	538,000
Q2 2012	481,876	606,000
Q3 2012	557,379	722,000
Q4 2012	658,156	783,210
Q1 2013	724,934	783,210
Q2 2013	755,379	783,210
Q3 2013	783,210	783,210

As part of the cost share agreement, Oregon State University contributed \$197,389 to this project, for a total project cost of \$980,599. The Oregon State University contribution was 20% of the total project expense.

Note 1: 2012 Q3 and Q4 Projected Expenses include grant extension.

Note 2: 2013 projected expenses are the same as Q4 2012 due to no cost extension.

8 Task Discussion

8.1 Task 1.0 : Project Management and Planning

This task continuously provided updates on the progress of the project and provide adjustments if necessary.

8.2 Task 2.0 : Determine Sensor Configurations

Identify sensor configurations and impact of sensor pairings on system performance:

Subtask 2.1 : Quantify effectiveness of Sensor Configurations. (supports milestone 1)

Subtask 2.2 : Form “Sensor Teams” to Augment Sensor Effectiveness. (supports milestone 2)

8.3 Task 3.0 : Determine Sensor Objectives

Determine sensor objectives leading to good system behavior and demonstrate system reconfigurability:

Subtask 3.1. Derive effective sensor objective functions. (supports milestone 3)

Subtask 3.2. Demonstrate System reconfigurability in simulations. (supports milestone 4)

Subtask 3.3: Demonstrate System scalability for advanced energy systems in *hardware-based* simulation. (supports milestone 5)

Subtask 3.4. Demonstrate communication robustness for sensor networks for advanced energy systems. (supports milestone 6)

Subtask 3.5: Demonstrate System Reconfigurability for advanced energy systems in *hardware-based* simulation. (supports milestone 7)

9 Milestone Reporting

All project milestones have been met. This research resulted in multiple publications that support the milestones [6, 8, 9, 10, 16, 18, 19].

Milestone 0 (June 2010) : Provide an assessment of current sensor coordination algorithms employed in power systems, with a particular emphasis of sensor objective functions.

Status: This milestone has been met.

Milestone 1 (September 2010) : Quantify sensor/system objective alignment for effective self-organization.

Status: This milestone has been met. Section 2 details completion of this milestone.

Milestone 2 (February 2011) : Improve Sensor Effectiveness by at least 10% using “Sensor Teams”.

Status: This milestone has been met. Sections 2 and 4 detail completion of this milestone.

Milestone 3 (December 2011) : Improve System Efficiency by at least 30% with new Sensor Objective Functions.

Status: This milestone has been met. Section 3 details completion of this milestone.

Milestone 4 (February 2012) : Show System Reconfigurability for 20% Change in System Conditions in Simulation.

Status: This milestone has been met. Section 4 details completion of this milestone.

Milestone 5 (October 2012) : Demonstrate Extreme Scalability of Proposed Algorithms.

Status: This milestone has been met. Section 3 details completion of this milestone.

Milestone 6 (March 2013) : Show Sensor Communication Network can overcome disruptions up to 20% of the network and provide reliable data.

Status: This milestone has been met. Section 4 details completion of this milestone.

Milestone 7 (May 2013) : Demonstrate System Reconfigurability for 20% Change in System Conditions in Simulation.

Status: This milestone has been met. Section 4 details completion of this milestone.

References

- [1] Pt-l1/l3/l10 amplified output pressure sensors. <http://www.apgsensors.com/products/pressure-sensors/pressure-transducers/pt-l1-l3-l10>, Oct. 2012.
- [2] Thermocouples. <http://www.omega.com/prodinfo/thermocouples.html>, Oct. 2012.

- [3] A. Agogino, C. Holmes Parker, and K. Tumer. Evolving distributed resource sharing for cubesat constellations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Philadelphia, PA, July 2012.
- [4] A. Agogino, C. Holmes Parker, and K. Tumer. Evolving large scale uav communication systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Philadelphia, PA, July 2012. **Best "Real World Applications" paper award.**
- [5] A. Agogino and K. Tumer. Analyzing and visualizing multi-agent rewards in dynamic and stochastic domains. In *Journal of Autonomous Agents and Multi-Agent Systems*, pages 320–338, 2008.
- [6] C. HolmesParker. *Improving Coordination and Scalability for Multiagent Systems*. PhD thesis, Oregon State University, June 2013.
- [7] D. Challet and N.F. Johnson. Optimal combinations of imperfect objects. *Physical Review Letters*, 89:028701, 2002.
- [8] M. Colby, C. Holmes Parker, and K. Tumer. Coordination and control for large distributed sensor networks. In *Future of Instrumentation International Workshop (FIIW-2012)*. Gatlinburg, TN, October 2012.
- [9] M. Colby and K. Tumer. Multiagent reinforcement learning in a distributed sensor network with indirect feedback. In *In Proceedings of the Twelfth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Saint Paul, Minnesota.
- [10] M. Colby and K. Tumer. Performance and fiscal analysis of distributed sensor networks in a power plant. In *AAMAS-2012 Workshop on Agent Technologies for Energy Systems*. Valencia, Spain, June 2012.
- [11] A. Farinelli, A. Rogers, and N.R. Jennings. Maximising sensor network efficiency through agent-based coordination of sense/sleep schedules. In *Proceedings of the International Workshop on Energy in Wireless Sensor Networks*, 2008.
- [12] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2010.
- [13] Lamber and Sinno. Bioinspired resource management for multi-sensor target tracking systems. In *MIT Lincoln Laboratory Project Report MD-26*, 2011.
- [14] M.J. Moran and H.N. Shapiro. *Fundamentals of Engineering Thermodynamics*. John Wiley and Sons, Inc., fifth edition, 2004.
- [15] D. Moriarty and R. Miikkulainen. Forming Neural Networks through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5, 1998.
- [16] C. Holmes Parker and K. Tumer. Combining difference rewards and hierarchies for scaling to large multiagent system. In *AAMAS-2012 Workshop on Adaptive and Learning Agents*. Valencia, Spain, June 2012.

- [17] A. Rogers, A. Farinelli, and N. Jennings. Self-organising sensors for wide area surveillance using the max-sum algorithm. *Lecture Notes in Computer Science. Self-Organizing Architectures*, 2010.
- [18] C. Roth. Agent objectives for evolving coordinated sensor networks. Master's thesis, University of Applied Sciences Offenburg, Germany, 2010.
- [19] C. Roth, M. Knudson, and K. Tumer. Agent fitness functions for evolving coordinated sensor networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Dublin, Ireland, July 2011.
- [20] S.J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.
- [21] K. Tumer. Designing agent utilities for coordinated, scalable and robust multi-agent systems. In P. Scerri, R. Mailler, and R. Vincent, editors, *Challenges in the Coordination of Large Scale Multiagent Systems*. Springer, 2005.
- [22] S. Williamson, E. Gerding, and N. Jennings. Reward shaping for valuing communications during multi-agent coordination. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.