AN ABSTRACT OF THE THESIS OF

Ehsan Nasroullahi for the degree of Master of Science in Mechanical Engineering presented on March 9, 2012.

Title:

Combining Coordination Mechanisms to Improve Performance in Multi-robot Teams

Abstract approved: ______________________________

Kagan Tumer

Coordination is essential to achieving good performance in cooperative multiagent systems. To date, most work has focused on either implicit or explicit coordination mechanisms, while relatively little work has focused on the benefits of combining these two approaches. In this work we demonstrate that combining explicit and implicit mechanisms can significantly improve coordination and system performance over either approach individually. First, we use *difference* evaluations (which aim to compute an agent's contribution to the team) and stigmergy to promote implicit coordination. Second, we introduce an explicit coordination mechanism dubbed Intended Destination Enhanced Artificial State (IDEAS), where an agent incorporates other agents' intended destinations directly into its state. The IDEAS approach does not require any formal negotiation between agents, and is based on passive information sharing. Finally, we combine these two approaches on a variant of a team-based multi-robot exploration domain, and show that agents using a both explicit and implicit coordination outperform other learning agents up to 25%.

Combining Coordination Mechanisms to Improve Performance in Multi-robot Teams

by

Ehsan Nasroullahi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented March 9, 2012
Commencement June 2012

ACKNOWLEDGEMENTS

I would like to acknowledge and thank my adviser, Professor Kagan Tumer, for supporting my research, providing guidance and advising me to the completion of this work. I thank my other committee members for their time and support, and the AADI laboratory for providing me with insight, support, and feedback throughout this process.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Chapter 1 – Introduction

In recent years, there has been a growing interest in autonomous systems including autonomous robots, UAVs, and sensor networks [1, 2, 20, 30]. Traditionally, much of the emphasis has been placed on single-device autonomy, where a single device (e.g rover, satellite, Unmanned Aerial Vehicles (UAV)) is operated via autonomous control methods in order to achieve some goal or complete some task. Although this is an important control problem, the capabilities of a single device are inherently limited. In order to address the limitations of single devices, researchers have begun looking into controlling sets of these devices such that they collectively achieve a common goal that none of the individuals could accomplish independently. In such systems, a new type of control problem arises: How to best coordinate the actions of a set of autonomous devices such that they cooperatively work together in order to achieve a common goal. This problem is very complex, as in many cases the dynamics of the system model are unknown, individual devices may be unreliable, and different devices may have conflicting goals. The field of multiagent systems was developed to address the control issues present in these systems.

Coordinating a team of agents such that they collectively achieve a common goal is a complex problem within the field of multiagent systems [31]. Improving coordination in multiagent systems will benefit many application domain including Unmanned Aerial Vehicles (UAV) swarms, search and rescue missions, exploration, and sensor networks [1,2,17]. In general, coordination mechanisms can be broken down into two main categories: implicit and explicit coordination mechanisms.

Implicit coordination relies solely upon an agent's observation of the environment, other agents, and the surrounding system to make decisions. One way is stigmergy, which is a way agents coordinate by communicating with each other within the environment [16]. The other way is coupling the actions of a set of agents who are not permitted to explicitly communicate is through the coupling of their policy evaluation functions. Here, agents share a policy evaluation function that is somehow reflective of how the agents collectively performed at achieving a particular objective. These evaluations are selected in such a way that if every agent aims to optimize its own fitness, it is simultaneously learning to optimize the system objective. In this work, we use two policy evaluation functions that have been shown to be successful in a number of domains (global and difference evaluation functions). Although policy evaluation functions alone have been shown to improve performance, convergence time, coordination, and scalability in multiagent systems, they have been unable to fully address the coordination issues in large multiagent systems. We address this shortcoming by adding our own explicit coordination mechanism to improve performance.

Explicit coordination involves direct interaction and exchange of information between two or more agents. Examples of explicit coordination are negotiations and auctions. However, these methods are frequently complex and require a back-and-forth dialogue between agents before reaching agreement and taking any action, thus slow system response time [11]. Also, in many real world problems communication is limited in bandwidth and availability. In this work, we propose to utilize a novel explicit coordination mechanism called *Intended Destination Enhanced Artificial State (IDEAS)* which only requires passive information sharing between agents. In particular, agents passively communicate the action that they currently intend to take based upon their perceived system state. This information provides agents with an effective "look ahead" at what other agents intend to do during the next time step, allowing them to adjust their actions to compensate accordingly.

While explicit coordination involves direct interaction and exchange of information between two or more agents, implicit coordination relies solely upon an agent's observation of its environment to coordinate and make decisions. Implicit coordination mechanisms tend to be limited by observation restrictions and explicit methods are typically limited by communication restrictions. In many real-world domains agents have access to limited amounts of both observation and communication. In such cases, maximizing the benefit of both types of information by concurrently using implicit and explicit mechanisms is likely to be advantageous. Although both implicit and explicit coordination methods have been shown to work well individually, it is likely that both approaches offer complimentary benefits in multiagent systems. We propose using a combination of explicit and implicit coordination mechanisms to improve coordination and performance over either method individually. We rely on policy evaluations and stigmergy to improve implicit agent-to-agent coordination. While we use communicating information about each agents' IDEAS to improve explicit agent-to-agent coordination (coordinating IDEAS information allows agents to adjust their actions based upon the intended actions of others) in Cooperatively Coupled Rover Domain.

The $CCRD$ is an extension of the Continuous Rover Domain developed by [1], in which a set of rovers must coordinate their actions to collectively optimize coverage over a set of environmental points of interest (POIs) and individual rovers can observe any given POI. The $CCRD$ increases the coordination complexity by requiring teams of agents to observe each POI. Here, agents must not only optimize coverage as a collective, but they must form teams and the teams must coordinate within themselves to optimize coverage of their POI as well. This is difficult because there are two different coordination problems going on concurrently. First, at a high level all agents within the system must coordinate to provide optimal coverage of POIs. Second, agents must coordinate amongst themselves to form teams, and the agents comprising the teams must coordinate their actions to optimally select and cover a given POI (teams either observe a POI together or not at all). This tight coupling between agents both at a system level and at a team level present a complex coordination problem. The key contributions of this thesis are as follows:

1. Introduce a novel explicit coordination mechanism (*IDEAS*) and demonstrate its usefulness by applying it to the $CCRD$.
2. Combine implicit (policy evaluation and stigmergy) and explicit (*IDEAS*) coordination mechanisms to improve performance of multi-rover teams in the $CCRD$ with a static environment.
3. Extend implicit (policy evaluation and stigmergy) and explicit (IDEAS) co-

ordination mechanisms into dynamic environment where the POIs' value and location change at the beginning of every episode.

4. Demonstrate the scalability of the approach in the $CCRD$ that contain maximum of 50 rovers.

The remainder of this thesis is structured as follows. Section 2 provides background information on the Continuous Rover Domain, multiagent systems, multiagent coordination, and multiagent learning. Section 3 provides an introduction to the $CCRD$ used in this work. Section 4 provides an overview of the algorithms, evaluation functions, and methods used in this work. Section 5 contains the experimental results. Finally, Section 6 contains the discussion and conclusions of this work.

Chapter 2 – Background

2.1 Multiagent Systems

The field of multiagent systems contains elements from two primary fields of research, *distributed problem solving* and *artificial intelligence*. Distributed problem solving is the process of decomposing a problem into modular tasks, and allocating those tasks among several individual nodes in order to collectively construct a solution to the problem [31]. We refer the interested reader to [12] for more detailed information on *distributed problem solving*. Artificial intelligence attempts to not just understand, but to build intelligent entities that perceive their environment and take actions to maximize their chances of success [33]. Multiagent techniques have been shown to be successful in a number of domains due to their inherently distributed approach, robustness to component failures and environmental uncertainty, reconfigurability, and adaptability to changing system requirements [31,34].

In this thesis, we are primarily interested in multiagent learning techniques. Machine learning is a subfield of artificial intelligence that has been shown to work well in a number of data processing, pattern recognition, and single-agent domains [3]. Extending machine learning methods to multiagent domains is difficult because of the increased complexities that arise from agent-to-agent interactions, so called emergent behavior.

The use of multiagent techniques in complex distributed systems has many benefits over traditional non-agent-based solutions including increased scalability, robustness to component failure and system noise, and adaptability to sudden unplanned changes in highly dynamic environments [41]. Coordination between cooperative teams of agents can improve the performance of individual agents within the system, as well as the system as a whole [23]. Networks of collaborative agents have performed well in a number of problem domains including robot soccer [34], traffic congestion [39], collaborative exploration [37], coordinating multiple rovers [38], routing data over a network [39], predator-prey, and search and rescue [31]. More recently, there has been increased interest in resource sharing within constellations of autonomous heterogeneous satellites. To date, agent-based satellite coordination research has included: resource allocation between on-board peripherals of individual satellites (processor and power allocation), autonomous coalition formation based upon negotiation mechanisms, and satellite-to-satellite coordination and resource sharing mechanisms to complete complex missions [4–6,10,29]. In dynamic and noisy environments, unanticipated events can easily cause a breakdown in preplanned, hand-coded coordination mechanisms. Applicable solutions must be able to adapt to rapid changes within the system due to noise and component failure without the loss of system functionality [36].

2.2 Multiagent Coordination

Coordination is a critical element in multiagent systems. Coordination can be done either *explicitly* or *implicitly*. Explicit methods involve direct agent-to-agent communication, negotiation, or information exchanging. Implicit coordination on the other hand is indirect and typically occurs through environmental interactions. A couple examples of implicit communication include coupling agent reward functions [2] or leaving pheromones or trails that other agents in the environment can detect and follow [28]. In many cases, multiagent systems rely heavily upon agent-to-agent negotiation in order to coordinate resource allocation, task management, or to achieve objectives [11]. Agents can interact and negotiate with both learning and non-learning agents present within the environment in order to achieve their goals.

2.2.1 Explicit Coordination

Explicit coordination involves direct interaction and exchange of information between two or more agents. Examples of explicit coordination are negotiations and auctions. However, these methods are frequently complex and require a back-and-forth dialogue between agents before reaching agreement and taking any action, thus slow system response time [11]. Also, in many real world problems, communication is limited in bandwidth and availability. In this work, we propose to utilize a novel explicit coordination mechanism called *Intended Destination Enhanced Artificial State (IDEAS)* which only requires passive information sharing between agents. In particular, agents passively communicate the action that they currently intend to take based upon their perceived system state. This information provides agents with an effective “look ahead” at what other agents intend to do during the next time step, allowing them to adjust their actions to compensate accordingly.

2.2.2 Implicit Coordination

Implicit coordination relies solely upon an agent’s observation of the environment, other agents, and the surrounding system to make decisions. One way of implicit coordination is stigmergy, which is a way agents coordinate by communicating with each other within the environment [16]. The other way is coupling the actions of a set of agents who are not permitted to explicitly communicate is through the coupling of their policy evaluation functions. Here, agents share a policy evaluation function that is somehow reflective of how the agents collectively performed at achieving a particular objective. These evaluations are selected in such a way so that if every agent aims to optimize its own fitness, it is simultaneously learning to optimize the system objective.

In this work, we use two policy evaluation functions that have been shown to be successful in a number of domains (global and difference evaluation functions). Although policy evaluation functions alone have been shown to improve performance, convergence time, coordination, and scalability in multiagent systems, they have been unable to fully address the coordination issues in large multiagent systems.

We address this shortcoming by adding our own explicit coordination mechanism to improve performance.

2.2.3 Analysis of Coordination Approaches

Implicit coordination relies solely upon an agent's observation of its environment to make decisions, while explicit coordination involves direct interaction and exchange of information between two or more agents. Implicit coordination mechanisms tend to be limited by observation restrictions and explicit methods are typically limited by communication restrictions. In many real-world domains, agents have access to limited amounts of both observation and communication. In such cases, maximizing the benefit of both types of information by concurrently using implicit and explicit mechanisms is likely to be advantageous. We propose using a combination of explicit and implicit coordination mechanisms to improve coordination and performance over either method individually.

In this work, we use a passive form of explicit communication to promote coordination between learning-based agents (passive information sharing). For allocation of resources, tasks and coordination, successful solutions have been provided through auctions, bidding, and other forms of negotiations [24, 29, 32, 36]. The explicit coordination mechanism that we are using is similar to the reactivity coordination mechanism in [19], which can be described as agents coordinating with each other via reacting with their local environment and also adapting their action based on other agents' actions. In our explicit coordination approach we are observing the agents' local environment as is done with reactive coordination, where agents leave traces within the environment for other agents (similar to pheromone trails in swarm optimization).

2.2.4 Stigmergy

Stigmergy is another coordination method that is similar to our approach, which is inspired from animal to animal interactions. This coordination method is based on the indirect communication between agents, where the communication shared between agents is based upon information and signs left behind in the environment by the agents (implicit coordination) [16]. For instance the environment of ants changes as they clean their nest (nest gets cleaner), thus less participants will help to clean until no ants remain cleaning the nest and the nest is clean. The main difference between our approach and stigmergy is that agents explicitly communicate the signal for coordination where as in stigmergy it is implicit communication (leaving signs behind in environment as result of action).

2.3 Agent Learning

Learning is often an important component of multiagent systems. Learning methods can generally be grouped into one of three main categories: *supervised*, *unsupervised*, and *reward-based* learning. Supervised learning is learning in the presence of a "teacher", which tells an agent whether the action it took was right or wrong. Supervised learning works well for classification problems in which a set of training examples are available. However, in many complex real world domains, dynamic interactions between agents and stochasticity in the environment make it impossible to know what the correct actions are. Unsupervised learning occurs when an agent is simply put in an environment and learns patterns from its inputs and observations without receiving any explicit feedback. A common example of unsupervised learning is clustering: detecting potentially useful or related clusters from a set of input examples [33]. Reward based learning is often called "semi-supervised" learning: there is no explicit target function, but there are rewards which provide feedback for actions taken. In this work we will focus on reward-based learning methods, namely reinforcement learning and evolutionary algorithms. In particular, we focus on utilizing these semi-supervised learning methods within a multiagent system setting. A comprehensive list of single and multiagent reinforcement learning algorithms can be found in [7, 35, 42].

2.3.1 Reinforcement Learning

Within the field of machine learning, reinforcement learning can be considered a computational approach to understanding and automating goal-directed learning and decision making [35]. Reinforcement learners observe the state of the environment and attempt to take actions that maximize their expected future reward. In this work, an agent's environment consists of the world and all other agents and the problem is represented as a four-tuple MDP. A reinforcement learning problem consists of at least one agent and an environment. A reinforcement learning agent has four key elements including a *policy*, *reward function*, *value function*, and optionally an *environmental model* [35].

An agent's *policy* defines the way the agent behaves at any given time [35], it can be thought of as the agent's controller. This controller maps the agent's perceived environmental state to the action it takes in that state. These policies are initialized in an arbitrary manner, and adjusted over time as the agent learns. Each agent's policies are updated via the agent's *reward* and *value function.*

An agent's *reward* function reflects the agent's goal in a reinforcement learning problem [35]. The reward function provides an agent with a learning signal for actions taken. In general, the learning signal is positive if the agent's actions were beneficial to its goal, and negative if the actions were detrimental to the agent's goal. The rewards an agent receives are frequently coupled with a value function in order to update the agent's policy (controller).

A reinforcement learning agent uses some form of a *value function* in order to update its policy based upon the reward it receives. There are many forms of value updates in reinforcement learning, depending upon the domain. Here, we will introduce one of the most common forms of value functions known as a Q-function. At every episode an agent takes an action and then receives a reward evaluating that action. Agents select the actions corresponding to the highest Q-value with probability $1 - \epsilon$, and chooses a random action with probability ϵ. The constant ϵ is an exploration rate. After taking action a and receiving reward R an agent updates its Q table (which contains its estimate of the value for taking action a in

Algorithm 1: This is an algorithm for Q-learning reinforcement algorithm. Here, agents observe their environmental state s and select an action a based upon their current policy. They then receive a reward R, which is used to update the policy.

```
- Agent has $A$ actions
- Agent has a Q-table with $Q(s,a)$ for all possible state-action pairs ($s$,$a$)
for $Run < Run_{max}$ do
    for $Episode < Episode_{max}$ do
        for $t < t_{max}$ do
            - Agent makes an $\epsilon$-greedy action selection
            - Agent receives reward $R(s)$ and updates Q-table
                  $Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) - Q(s,a) + \gamma maxQ(s',a')\right)$
            - Agent observes new environmental state $s'$
        end
    end
end
```

state s [35]) via the Q-update function as follows (Algorithm 1):

$$Q'(s,a) = Q(s,a) + \alpha\left(R(s) - Q(s,a) + \gamma maxQ(s',a')\right) \tag{2.1}$$

where,

- $Q'(s,a)$ is the updated Q-value for taking action a in state s at time t
- $Q(s,a)$ is the current Q-value for taking action a in state s at time t
- $max_{a'}Q(s',a')$ is the maximum possible Q-value associated with taking action a' in state s' at time $t+1$
- $R(s)$ is the reward received for the agent being in state s at time t
- $\alpha \ \epsilon \ \{0,1\}$ is the learning rate

- $\gamma \epsilon \{0,1\}$ is the discount factor

Each of the $Q(s,a)$ values provides an agent with a measure of the amount of reward it can expect to achieve over time for taking action a in state s. The learning rate α controls how quickly an agent learns. If α is set low, new rewards will not impact the agents' $Q(s,a)$ values as quickly, resulting in slower learning and slower changes in behavior. This is useful in domains where the environment changes slowly over time. When the α parameter is set high, learning occurs rapidly, pushing the $Q(s,a)$ values to change quickly with respect to rewards received. This is very useful in domains where the environment is changing rapidly, and the agents policy needs to change accordingly. The discount factor "γ" on the other hand impacts how far ahead an agent looks when considering its actions. A discount factor of $\gamma = 0$ will consider only the immediate reward obtainable from taking an action a in the current state s. A higher discount factor causes an agent to consider the down-stream effects of its actions, how the action it takes in the current state s will impact the cumulative reward it receives in the future.

2.3.2 Neural Networks

Neural networks (NN) have been used in many applications including: control problems, classification, function approximation, and nonlinear signal-processing [8, 15, 18, 25, 27]. They are useful tools for representing functions and can represent both discontinuous and continuous functions to arbitrary accuracy [1, 8, 9, 15].
Neural networks are biologically inspired mathematical tools modeled after the

function of the brain. Each neural network maps a set of inputs to a set of outputs. A neural network consists of a set of input nodes, output nodes, and hidden layer nodes which are connected with weights assigned to each connection (Figure 2.1). The weighted sum of the connections mapped from the inputs through the nodes of the neural network generate a set of outputs (where each node itself has an activation function, which is typically nonlinear) [13]. There are many ways of training a neural network including supervised, unsupervised, and reward based methods. A neural network controller utilizing supervised learning has a "teacher" that knows the correct mapping from inputs to outputs, and provides the neural network with constructive feedback on actions taken, improving the neural networks performance over time. Using unsupervised learning with a neural network attempts to use the neural network to cluster a set of unlabeled data using its similarities. Reward based learning with neural networks on the other hand utilize reward feedback based upon actions taken in order to update the neural networks [1, 9, 15, 25, 27]. In this work, each rover uses 1-hidden layer feed forward neural network as their control policies.

2.3.3 Evolutionary Algorithms

Evolutionary algorithms are biologically-inspired algorithms based upon "the survival of the fittest". These algorithms have been successfully applied to a wide range of problems including optimizing component design, optimizing functions in large search spaces, and developing control policies for individual agents in multi-

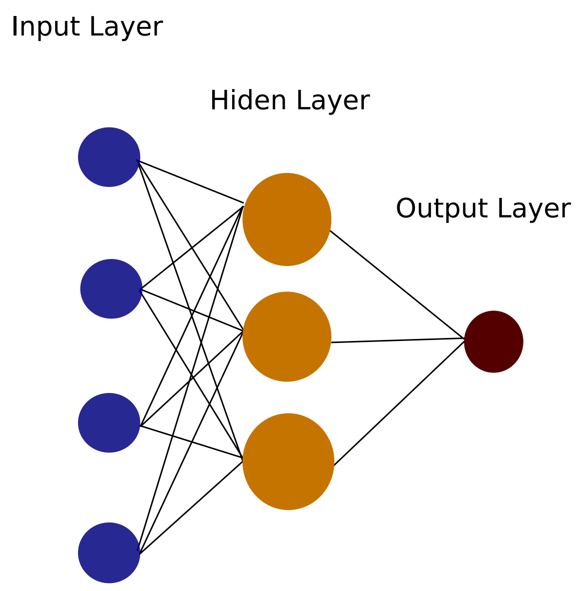

Figure 2.1: Network diagram for a two layer feed forward Neural Network (NN). The input, hidden and output variables are represented by nodes, and the weight parameters are represented by links between the nodes. In feed forward NN the information flow through the network from input to hidden and then to output layer.

agent systems [1, 14, 26].

These algorithms evolve a set of policies called a *population*. Over time they are evolved in a way that they optimize a given *evaluation function*. The population is evolved over time via *mutation* and/or *crossover*, then the best policies are *selected* by how well they perform (evaluation). At each time step, these algorithms select the best policy of their population $100 - \epsilon$ percent of the time and select a random member ϵ percent of the time (ϵ - greedy selection) and alter it via mutation and/or crossover. Then, they compare the quality of all of their policies, including the newly created policy and remove the worst policy with some probability [33].

Over time, this evolutionary process yields to policies that aim to maximize the agent's evaluation function.

Algorithm 2: General Evolutionary algorithm, where this algorithm starts with a populations of policies and mutate them over time in an attempt to optimize a evaluation function.

```
for $Run < Run_{max}$ do
    - Initialize policies ($P$)
    for $Episode < Episode_{max}$ do
        - Select a policy ($P_i$) using $\epsilon-$greedy selection
        - Randomly modify policy parameters ($P_i^{'}$)
        for $t < t_{max}$ do
            Use policy $P_i^{'}$
        end
        - Evaluate performance of policy $P_i^{'}$
        - Re-insert $P_i^{'}$ into the pool
        - Remove the worst policy from pool
    end
end
```

In this work, each *policy* (P_i) is controlled by the agent, which maps the agent's state (S) to it actions (a). Also every agent has a pool of policies (neural network), where each one has a *rank* associated with it. The rank of every policy is its evaluation of that policy toward maximizing its evaluation function. In our work, the agent selects a policy using ϵ - greedy selection algorithm (base on policy's rank) and duplicates it. Where ϵ-greedy selection is when we select the best policy 90% of the time and select random policy 10% of the time. Then, the selected policy is mutated, and used for a certain number of time steps (t_{max}). Finally the selected policy is evaluated based by evaluation function and compared with the members in the pool of policies, where the worst one is removed form the pool. This algorithm moves the agents toward selecting policies that maximize their evaluation, thus agents learn which policies are beneficial to increase their evaluation function.

2.3.4 Neuroevolutionary Algorithms

Neuroevolutionary algorithms are a subset of evolutionary algorithms that evolve sets of neural network policies. These algorithms evolve a set of policies where a pool of neural network policies are randomly created. Next a policy is selected based on an ϵ- greedy selection mechanism, then the selected neural network is mutated by adding a random number from cauchy distribution to its' weights, and then the quality of it is compared to the rest of the policies and the worst one is remove from the pool [13, 31]. Neuroevolutionary methods have been shown to work well in many multiagent domains involving continuous state spaces and complex agent interactions such as micro aerial vehicle control and the continuous rover domain used in this work [1, 21, 22].

2.4 Evaluation Functions for Policy Selection

One of the most important decisions that has to be made in neuroevolution and reinforcement learning is what evaluation function each agent will use to evolve its set of policies. The first and most direct policy evaluation function is to let each agent use the global system objective as the agent policy evaluation function.

However, in many domains, especially domains involving large numbers of agents, such a fitness evaluation often leads to slow evolution. This is because each agent has relatively little impact on its own evaluation. For instance if there were 100 agents and an agent takes an action that improves the system evaluation, it is likely that some of the 99 other agents will take poor actions at the same time, and the agent that took a good action will not be able to observe the benefit of its action.

Next we introduce *factoredness* and *learnability* which are two important properties to consider when selecting policy evaluations.

2.4.1 Factoredness and Learnability

Prior to discussing the agent policy evaluation functions used in this work, we introduce two separate metrics for determining the quality of a policy evaluation function. Ideally, a policy evaluation function should provide an agent with two key pieces of information: 1) How its action impacted the overall system performance, and 2) How its action impacted the evaluation it received. Feedback on how they impacted the system performance allows agents to make decisions that are in-line with the system objective. Providing agents with feedback on how their individual action impacted the evaluation they received allows agents to change their own actions in order to benefit both themselves and the system. We formalize the first property for an agent i, by defining the degree of factoredness (also presented in [2, 39]) between the agent policy evaluation function g_i and system objective G

at state z, as:

$$F_{g_i} = \frac{\sum_z \sum_{z'} u[(g_i(z) - g_i(z'))(G(z) - G(z'))]}{\sum_z \sum_{z'} 1} \tag{2.2}$$

where the states z and z' only differ in the state of agent i, and $u[x]$ is the unit step function, equal to 1 if $x > 0$. The numerator keeps track of the number of state pairs (z, z') for which the agent policy evaluation function $g_i(z) - g_i(z')$ and system objective $G(z) - G(z')$ are aligned (same sign). A high degree of factoredness means that agents improving their own local policy evaluation function are concurrently improving the system performance (with respect to the system objective), while agents harming their local policy evaluation function are also harming system performance. Next, we define the second property as learnability, which is the degree to which an agents policy evaluation function g_i was impacted by its own actions as opposed to the actions of other agents. The learnability of a policy evaluation function g_i for agent i, evaluated at z can be quantified as follows:

$$L_{g_i} = \frac{||g_i(z) - g_i(z - z_i + z_i')||}{||g_i(z) - g_i(z' - z_i' + z_i)||} \tag{2.3}$$

where, in the numerator z' differs from z only in the state of agent i, and in the denominator the state of all other agents is changed from z to z'. Intuitively, the learnability provides a ratio between the portion of the agents evaluation that depended upon its own actions (signal), and portion of its evaluation signal that depended upon all other agents (noise).

2.4.2 Difference Evaluations

One of the policy evaluations that consider as highly factored and highly learnable is Difference policy evaluation. Consider difference evaluations of the form [39,40]:

$$D_\eta(z) = G(z) - G(z - z_\eta) \tag{2.4}$$

where z_η is the action of agent η. All the components of z that are affected by agent η are removed from the system. Intuitively this causes the second term of the difference evaluation to evaluate the performance of the system without η and therefore D evaluates the agent's contribution to the system performance. There are two advantages to using D: First, because the second term removes a significant portion of the impact of other agents in the system, it provides an agent with a "cleaner" signal than G. This benefit has been dubbed "learnability" (agents have an easier time learning) in previous work [2,39]. Second, because the second term does not depend on the actions of agent η, any action by agent η that improves D, also improves G. This term which measures the amount of alignment between two evaluations has been dubbed "factoredness" in previous work [2,39].

Chapter 3 – The Cooperatively Coupled Rover Domain

Team work is highly necessary in many real world domains including UAV swarms, search and rescue missions, and exploration [1, 2, 17]. In such domains, teamwork is necessary to decrease the amount of time it takes to complete tasks and to improve overall performance. The Continuous Rover Domain is a prime example of a system where multiple autonomous devices need to coordinate their actions in order to collectively optimize the system performance. Here a team of rescuers (rovers) are looking for individuals (Points Of Interest (POIs)) to rescue. Every POI in this domain has a value associated with it, and an observation radius which is the maximum distance from which a rover can observe the POI. The goal of these rovers is to collectively observe as many environmental POIs as possible. In such domains, it is impractical for an individual rover to single-handedly search the entire domain. Instead, the rovers must coordinate in order to divide up coverage areas (different rovers search different portions of the domain) in order to maximize the number of POIs found and to minimize the amount of time it takes to find the POIs. This thesis investigates the interactions and coordination between the rovers, specifically when multiple rovers are require to observe each POI. This requirement that multiple rovers are required to observe each POI leads to the Cooperatively Coupled Rover Domain used in this work, which is an extension of the Continuous Rover Domain used in [1].

In this section we discuss the Cooperatively Coupled Rover Domain that has been adapted and modified from the original Continuous Rover Domain [1]. This domain contains a set of rovers (agents) that are able to move around in a two dimensional plane to observe Points of Interest (POIs). Each of these POIs has a value (V_i) assigned to them. The goal of the agents is to form teams and for the teams to optimize their coverage of environmental POIs. In the cooperatively coupled rover domain, each agent has two types of sensors (POIs, Rover) and a total of 8 sensors (four of each type) similar to the original continuous rover domain. These sensors represents the density of either POIs or other rovers in this domain. In the cooperatively coupled rover domain, to earn credit for observing a given POI, a team of M rovers must observe it together (if fewer than M rovers observe the POI or the M rovers are not in the observation distance no credit is given). The teams are implicitly formed and are defined as being the closest M rovers to a given POI. The goal of agents in this domain is for rovers to collectively position themselves such that teams of M rovers are optimally observing each POI.

3.1 Agent State Representation

In the cooperatively coupled rover domain, each agent has two types of sensors (POIs, Rover) and a total of 8 sensors (four of each type). Each rovers' point of view is divided into four quadrants so that each quadrant contains both a POI and a rover sensor. The orientation of these quadrants are based upon the rovers heading. The quadrant axes are oriented according to the rover's current heading.

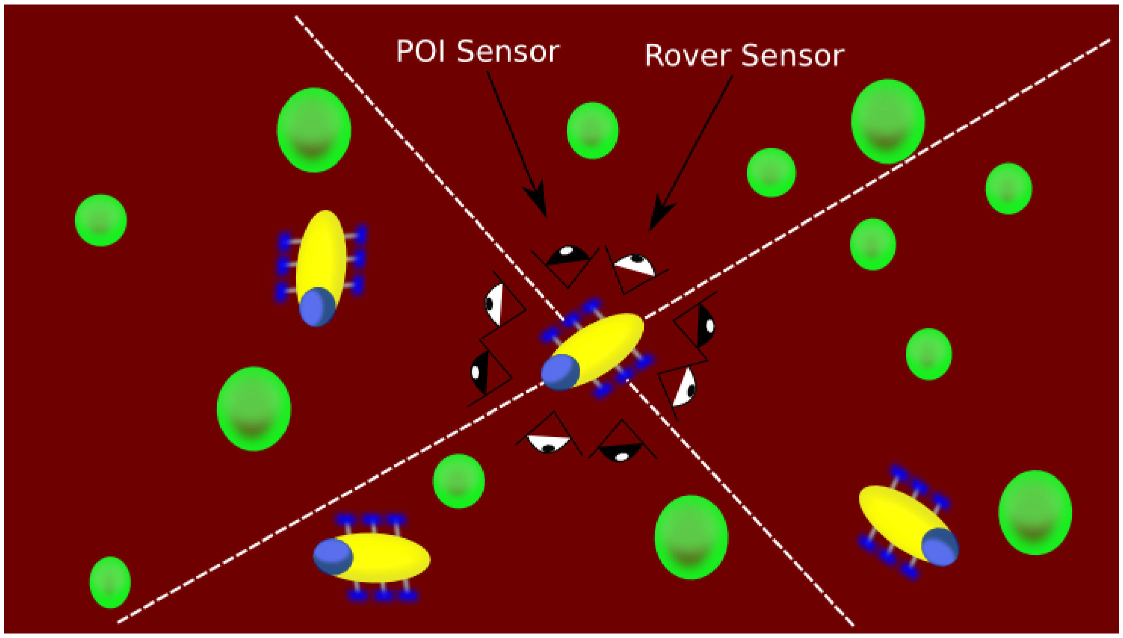

Figure 3.1: The figure shows how rovers observe their environment. Each rover has a set of 8 sensors (4 POI sensors and 4 rover sensors). The sensors provide coverage over 4 quadrants, where each quadrant contains one POI sensor and one rover sensor (Chapter 3).

At every given time t, each of the sensors return a density of POIs or rovers (respectively) in their quadrant. The value of this density is the sum of the values of each of the POIs or rovers divided by their Euclidean distance from the sensor. This yields the POI and rover density values for each quadrant (Equations 3.1 and 3.2, respectively):

$$s_{1,q,\eta,t} = \sum_{i \in POI} \frac{V_i}{\delta(L_i, L_{\eta,t})} \tag{3.1}$$

$$s_{2,q,\eta,t} = \sum_{\eta \in Rover} \frac{1}{\delta(L_{\eta'}, L_{\eta,t})} \tag{3.2}$$

where $s_{1,q,\eta,t}$ and $s_{2,q,\eta,t}$ are the POI and rover sensor readings respectively for quadrant q at time step t for agent η, $\delta(x, y)$ is a Euclidean norm function, L_i is the location of POI i, and $L_{\eta,t}$ is the location of rover η at time t. The variables $s_{1,q,\eta,t}$ and $s_{2,q,\eta,t}$ represent the system state in this domain.

3.2 Agent Action Representation

Every rover (agent) in this domain moves around in a continuous two-dimensional domain based upon its calculated actions. The actions are chosen by the agent's current policy, generated using the neuroevolutionary algorithm described in Section 4.1. The movement of each rover is governed by the neuroevolutionary algorithm as follows:

$$dx = d(O_1 - 0.5) \tag{3.3}$$

$$dy = d(O_2 - 0.5) \tag{3.4}$$

where $d/2$ is the maximum distance a rover can move in a given direction during a single time step (10 units), O_1 and O_2 are the x and y outputs from the agents current neural network policy, and dx and dy are the action selections of the agent.

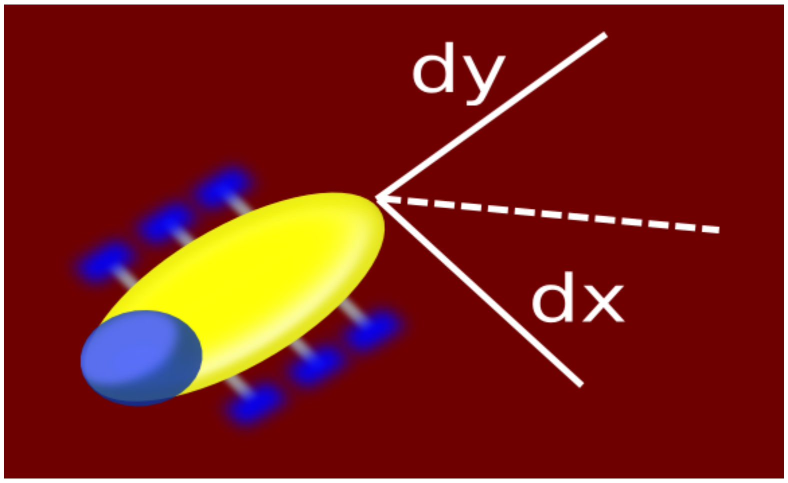

Figure 3.2: This figure represents the rover heading. At each time step the rover has two continuous output (dy, dx) giving the magnitude of the motion in two-dimensional plane relative to the rove's orientation.

3.3 System Objective for the Cooperatively Coupled Rover Domain

In cooperatively coupled rover domain, the system objective is to maximize the POI coverage where a team of rovers are required to observe a POI. In this work, although we have three different policy evaluation such as Global, Difference, and Random. We used system objective (Global) to grade agents' policy. The system objective of the rovers in the standard cooperatively coupled rover domain is to optimize the coverage of the POIs according to the following:

$$G(z) = \sum_{t} \sum_{i \in POI} \sum_{m=1}^{M} \frac{V_i}{\delta_{i,t}^{m}} \tag{3.5}$$

where G is the system objective when M number of rovers are required to observe i^{th} POI, V_i is the value of the POI, $\delta_{i,t}^{m}$ is the distance between the i^{th} POI and the

m^{th} closest rover to it at time step t. Intuitively, the system objective is maximized when the rovers collectively observe all of the POIs in an optimal manner (rovers observe each POI as closely as possible).

Chapter 4 – Agents and Coordination

This section describes our algorithms, policy evaluation functions, and how we implement those in the cooperatively coupled rover domain. In this work, each agent has a neuroevolutionary as a learner that evolved a pool of neural network policies, where the policies are evaluated via agent policy evaluation functions. We used two different policy evaluation functions to rank the policies, each of which is described in section 4.4.

4.1 Rover Policies

In domains with continuous action and state spaces like the cooperatively coupled rover domain, neuroevolutionary algorithms have been shown to be effective [1]. In this work, agents' policy is set by a neuroevolutionary learner. Here, each agent evolves its policy using a pool of 10 neural network policies. Each of these neural networks had 10 hidden units, 8 inputs, 2 outputs (input and output are bounded [0,1]), and utilized sigmoid activation functions. Algorithm 3 shows the process that we used to select, mutate, evaluate, and rank our policies. The algorithm is a population based search with the IDEAS explicit coordination mechanism added into it (Algorithm 3). In our algorithm we initialize rank of all of the policies to zero, used ϵ-greedy to select the next policy, 5000 episode (T_{max}), 15 time step

(t_{max}), 0.3 as mutation constant (ζ), 0.1 as learning rate (α).

Each agent initially began with a random policy selected from its pool of networks (N), and set to our current network (N_i), then we mutated N_i with probability of (1 - ζ) or kept it the same with probability of ζ. Mutating on every iteration results in shifting the population too dramatically to effectively search the policy space, and the neural network weights are updated using a Cauchy distribution centered at 0.3. Then we control our rover base on their N_i for 15 time steps, and at the end we equate the rank of the current network to sum of the previous steps rank, multiply by the (1 - learning rate), and the reward that rover receives base on the objective function multiply by learning rate. At the end of last step we rank the N_i and replace it with the worst one in population if it ranked lower. Then we select the next policy base on ϵ-greedy, and we repeat Mutation, Control, and Ranking process.

4.2 IDEAS for Explicit Coordination

We now introduce an agent's *IDEAS*. In this domain, an agent's IDEAS includes the value of the POI that the agent is currently headed toward along with its Euclidian distance to that POI. This information is explicitly broadcast between agents within the system, allowing the agents to effectively "know" what other agents plan to do in the following time step so they can coordinate and react accordingly. When agents share information about what they intend to do, prior to actually taking any actions it can lead to better coordination and improved

Algorithm 3: This algorithm presents the process of evolving policies. In this algorithm ϵ is exploration rate, ζ is mutation rate, α is learning rate and R is the reward that each agent gets from its evaluation function.

```
Initialize $N$ networks at $T = 0$
for $T < T_{max}$ do
    1. Select a policy $N_i$ from population
            With probability $\epsilon$: $N_i \leftarrow N_{random}$
            With probability $1 - \epsilon$: $N_i \leftarrow N_{best}$
    2. Mutate the selected policy $N_i$:
            with probability $\zeta$: Mutate $N_i$
            with probability $1 - \zeta$: Do Not Mutate $N_i$
    3. Control robot with $N_i$ for next episode
    for $t < t_{max}$ do
        if $IDEAS$ then
            Add IDEAS (Algorithm 2)
        end
        else
            Update State
        end
        Simulate
        Take an Action
        $Rank \leftarrow (1 - \alpha) * Rank + \alpha * R$
    end
    4. Ranking the $N_i$ base on performance
            $N_i$ Rank $\leftarrow Rank$
    5. Replace $N_{worst}$ with $N_i$
end
```

performance. This is because agents are able to actively account for each others' actions, without losing any time (agents are able to gain the insight from a future time step, without losing the time associated with taking that step). In order to determine an agents' IDEAS, the agent observes its environment and based upon what it observes it selects its current intended action (the agent observes the environmental state S and feeds it into its current policy to get out its intended action ($\vec{a} =< dx, dy >$). The agent then uses that "intended" action to predict it next move. In particular, the agent predicts the POI it will be closest to next time step and calculates its predicted distance to that POI. Each agent calculates this piece of information and all agents passively share this piece of information with each other. This piece of information is denoted as $s_{\eta,c}$:

$$s_{\eta,c} = \frac{V_{C_{POI}}}{\delta(L_{C_{POI}}, L_\eta)} \tag{4.1}$$

where $s_{\eta,c}$ is the portion of agent η's state vector that contains its predicted distance from $V_{C_{POI}}$, which is the value of the closest POI to agent η, $\delta(L_{C_{POI}}, L_\eta)$ is the Euclidean norm function, $L_{C_{POI}}$ is the location of the closest POI to rover η, and L_η is the predicted location of agent η at time $t+1$ based upon its current state and predicted action. This information represents the value of the POI that is predicted to be closest to agent η at time step $t+1$, divided by the predicted distance between the POI and agent η at time step $t+1$. This piece of information is what each agent explicitly shares via passive broadcast within each other in system. Each agent receives the set of all $s_{\eta,c}$ values $\{\mathbf{s}\}$. The next section discusses how this particular set of information is utilized by other agents.

4.3 IDEAS: Modified State

Each agent receives the set of $\{\mathbf{s}\}$ values (described in previous section) and uses them as a scaling factor I_i for their state information (given as).

$$I_i = \begin{cases} 1 & \text{Rover is the closest to the POI} \\ s_{\eta,c} & \text{Rover is not the closest to the POI} \end{cases} \tag{4.2}$$

We scale them such that if an agent is the closest to a POI, its value of that POI remains high (scaled by 1.0) and if there are many other rovers closer to the POI the POIs value is scaled by a number less than 1.0 represented by $s_{\eta,c}$. Here, an agent would interpolate each of the received set of $\{s\}$ values according to the equation 4.2. where the values I_i are scaling values between 0 and 1.0 that will be used to modify the agents current perceived state.

Each agent uses the information of I_i values to modify its perceived environmental state during time step t. Here, the state information would change as follows:

$$s_{1,q,\eta,t} = \sum_{i \in POI} \frac{I_i V_i}{\delta(L_i, L_{\eta,t})} \tag{4.3}$$

where $s_{1,q,\eta,t}$ is a modified version of agent η's system state information that incorporates the IDEAS from other agents (Equation 4.2), q is the quadrant, t is the time step, and i indexes the POI. Here, the state information is modified such that it reduces the observed value of POIs that are predicted to be heavily observed by agents and maintains high values for POIs that are not predicted to be observed heavily. The algorithm used to calculate each agents' current IDEAS, communicate them to other agents, convert them into scalar values for scaling the state, and adjust the perceived system state can be found in Algorithm 4.

Algorithm 4: This algorithm represents how $IDEAS$ has been implemented.

```
Update the states ($S$) for all agents
for $t < t_{max}$ do
    1. Simulate using $N'$
    2. Observe Next State ($S'$)
    3. Extract IDEAS from $S'$
    4. $S \leftarrow$ Update $S$ (Equation 4.3)
end
```

Now that we introduced agents' IDEAS and explain how they modify their state, we explain how we actually implement this in our simulation. In every simulation, all agents receive state information from their sensors. Then they run their selected network (N_i) with their perceived states as inputs, to generate intended actions. Then, based on the intended actions agents update their state (S') to include the IDEAS which were explained in Section 4.2. Then their updated states are used to generate the action to take.

4.4 Policy Evaluation Functions

One of the most important decisions that has to be made in neuroevolution is what policy evaluation each agent will use to evolve its set of policies. The first and most direct policy evaluation is to let each agent use the global system objective as the agent policy evaluation. However, in many domains, especially domains involving large numbers of agents, such a policy evaluation often leads to slow evolution. This is because each agent has relatively little impact on its own evaluation. For instance if there were 100 agents and an agent takes an action that improves the system evaluation, it is likely that some of the 99 other agents will take poor actions at the same time, and the agent that took a good action will not be able to observe the benefit of its action. In this work, agents receiving the system performance G as its policy evaluation received values according to Equation 3.5.

4.4.1 Global Policy Evaluation Functions

In this thesis we use Global policy evaluations to evaluate the performance of rovers' (agents') policy in cooperatively coupled rover domain. The system objective of the rovers in the standard cooperatively coupled rover domain is to optimize the coverage of the POIs according to the following:

$$G(z) = \sum_{t} \sum_{i \in POI} \sum_{m=1}^{M} \frac{V_i}{\delta_{i,t}^{m}}$$

where G is the system performance when M number of rovers are required to observe each POI, V_i is the value of each POI, $\delta^m_{i,t}$ is the distance between the i^{th} POI and the m^{th} closest rover to it at time step t. The value $\delta^m_{i,t}$ can be explain as follows:

$$\Delta_{i,t} \equiv \{\delta(L_i, L_{\eta_1,t}), \delta(L_i, L_{\eta_2,t}) \ldots \delta(L_i, L_{\eta_N,t})\} \tag{4.4}$$

where $\Delta_{i,t}$ contains the set of Euclidean norm distances between the i^{th} POI and the η^{th} rover at time step t.

$$\delta^1_{i,t} = min(\Delta_{i,t}) \tag{4.5}$$

$$\delta^2_{i,t} = min(\Delta_{i,t} \ - \ \{\delta^1_{i,t}\}) \tag{4.6}$$

$$\delta^3_{i,t} = min(\Delta_{i,t} \ - \ \{\delta^1_{i,t}\} \ - \ \{\delta^2_{i,t}\}) \tag{4.7}$$

where $\delta^1_{i,t}$, $\delta^2_{i,t}$, and $\delta^3_{i,t}$ are the distance between the i^{th} POI and the closest, second closest, and third closest rovers respectively to the POI at time step t

4.4.2 Difference Policy Evaluation Functions

In this work, we also utilize difference policy evaluations of the form [1, 39]:

$$D_\eta(z) = G(z) - G(z - z_\eta) \tag{4.8}$$

where z_η is the action of agent η. All the components of z that are affected by agent η are removed from the system. Intuitively this causes the second term of the difference evaluation to evaluate the performance of the system without η and therefore D evaluates the agent's contribution to the system performance. There are two advantages to using D: First, because the second term removes a significant portion of the impact of other agents in the system, it provides an agent with a "cleaner" signal than G. This benefit has been dubbed "learnability" (agents have an easier time learning) in previous work [2, 39]. Second, because the second term does not depend on the actions of agent η, any action by agent η that improves D, also improves G. This term which measures the amount of alignment between two evaluations has been dubbed "factoredness" in previous work [2, 39]. Here, we derive D for the Cooperatively Coupled Rover Domain where the number of rovers required to form a team to observe a POI is $M = 3$. In this case, combining Equations 3.5 and 4.8 yields the equation for the difference policy evaluations used in this work:

$$D_\eta(z) = \sum_t \sum_{i \in POI} \sum_{m=1}^{M} \frac{V_i}{\delta_{i,t}^{m}} - \sum_t \sum_{i \in POI} \sum_{m=1}^{M} \frac{V_i}{\delta_{i,t}^{m \neq \eta}}$$

$$D_\eta(z) = \sum_t \sum_{i \in POI} \sum_{m=1}^{M} \left\{ \frac{V_i}{\delta_{i,t}^{m}} - \frac{V_i}{\delta_{i,t}^{m \neq \eta}} \right\} \tag{4.9}$$

where D is the difference evaluation for agent η, M is the number of agents require to observe the i^{th} POI with the value of V_i, $\delta_{i,t}^{m}$ is the Euclidean distance from the m^{th} closest agent to the i^{th} POI, and $\delta_{i,t}^{m \neq \eta}$ is the Euclidean distance from the n^{th}

closest agent to the i^{th} POI where agent m is not η. In this case D is non-zero in three key situations: 1) when the rover is the closest rover to the i^{th} POI, 2) when the rover is the second closet rover to the POI, and 3) when the rover is the third closest rover to POI. In order to compute D_η we need to remove the impact of agent η from the system and calculate G without it, by replacing agent η in the calculation with the fourth closest rover to the POI. Had agent η not been present, the fourth closest agent would have been part of the 3 agent team instead. Comparing the performance with and without the agent provides solid feedback in the agents contributions to the system.

4.5 Stigmergy

Stigmergy is another implicit coordination mechanism that is based on indirect communication between agents within an environment (section 2.2.4). In the cooperatively coupled rover domain, stigmergy introduces a change in the environment, which reduced a POI's value by 3% each time it was observed by a team of rovers.

$$V_{i_{new}} = V_{i_{old}}(1 - \xi) \tag{4.10}$$

where $V_{i_{new}}$ represents the value of the i^{th} POI after the observation, $V_{i_{old}}$ represents the value of the POI before change and ξ is the amount that a POI will reduced by after each observation which is 3%. For instance, in this domain, when a team of rovers observe a POI, the value of that specific POI changes then all the other rovers sense that POI with new reduced value after it is observed. The value of each POI will be reinitialized to its starting value at the beginning of each statistical run. The following algorithm represents how we implement stigmergy in our approach:

Algorithm 5: This algorithm represents how stigmergy is implemented in the cooperatively coupled rover domain.

```
for $T < T_{max}$ do
    Initialize the POIs
    for $t < t_{max}$ do
        if $POI_i$ has been observed then
            $V_i = V_i(1 - \xi)$
        else
            $V_i$
        end
    end
end
```

where we initialize a set of POIs at the begining of every episode (T), then if the i^{th} POI is observed at any given time step t, the value of that POI is reduced by ξ, which is 3%.

Chapter 5 – Experiments and Results

We now apply our approach in both static and dynamic environments (Section 5.1 and 5.6), under various observation distances. These experiments require teams of 3 rovers to observe a given POI. Requiring teams of 3 rovers to observe each individual POI adds significant coordination complexity because each rover must rely upon other rovers' actions in order to observe a POI. Additionally, by adding a minimum observation radius for POIs, the complexity of the problem is increased because the rovers must coordinate their actions in order to explore the environment and find the POIs before they can observe them as a team. These coordination complexities are why the cooperatively coupled rover domain was selected to test our approach of combining learning based agents with various policy evaluations, stigmergy, and IDEAS.

In this work, we conducted six separate experiments:

1. Implementing of policy evaluation functions (global G and difference D policy evaluations) for various POI observation distances in the static Domain.
2. Combining of policy evaluation functions and stigmery (implicit coordination through environmental cues) with various POI observation distances in the static domain.
3. Implementing of policy evaluation functions and communicating agents' IDEAS with various POI observation distances in the static Domain.

4. Combining of policy evaluations, stigmergy, and communicating agents' IDEAS with various POIs' observation distances in the static domain.
5. Scaling the domain while rovers are learning with a combination of policy evaluations, stigmergy, and communicating agents' IDEAS in the static domain. To scale the domain we increased the number of rovers, POIs, and the over all dimensions of the domain.
6. Extend our approach (policy evaluations, stigmergy and IDEAS) to dynamic domain.

All the experiments (except the scaling experiments) were run for 20 statistical runs and 5000 episodes (40 time steps per episode in case of 40 rovers) in the world that contains 40 rovers and 66 POIs.

5.1 Evaluation Functions for Static Domain

The first set of experiments were conducted in the cooperatively coupled rover domain with static environmental conditions. This ensured that nothing in the environment changed except for changes that caused by the actions of agents within the system. In this case the domain contain 40 rovers and 66 POIs. The location of POIs were fixed at all times. Rovers are placed in between a high value POI (value of 10) and 15 low value POIs (value of 3) at the center of the world. We plot the results on a two-dimensional graph that has system performance ($G(z)$) on vertical axis, and POI observation distance on the horizontal axis. Separate experiments were conducted for different POI observation radii (the furthest away a rover can be from a POI and still receive credit for observing it, named POI's observation radius). These experiments were conducted for various POI observation radii between 10 and 100 units.

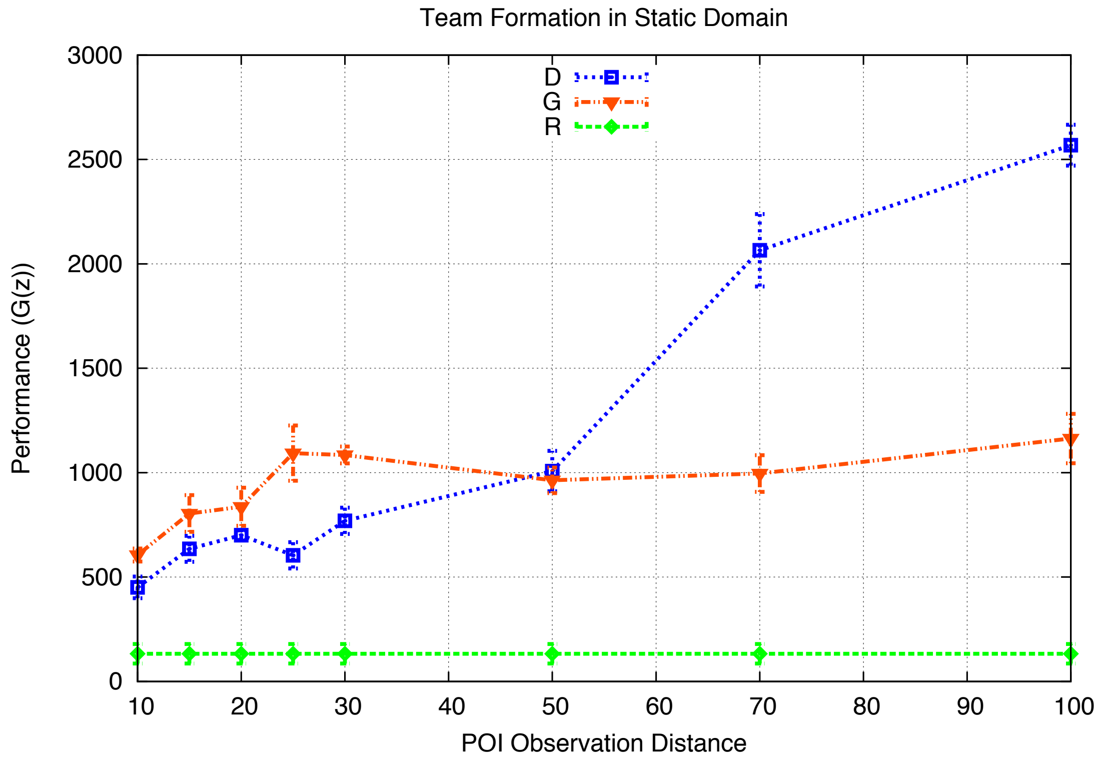

Figure 5.1: Cooperatively coupled rover domain with 40 rovers in a static environment, where teams of 3 agents are required to observe each POI. Agents using D outperform all other methods once the observation distance is above 50 units.

In the first experiment we are investigating the ability of agents using policy evaluation functions to coordinate their actions when team formation is required in static environment. Here, we require that teams of $M = 3$ agents must collectively observe each POI in order for it to count towards the system objective. Hence, agents must learn to coordinate their actions such that the set of agents collectively optimize the system performance. Agents use evaluation functions including a global evaluations function (G) and difference evaluations (D). We also evaluate the performance of agents using random policies to serve as a reference point to compare the performance of our algorithms. In many real world domains complete communication is unavailable, so we observed the performance of agents using evaluation functions under varying communication restrictions (Figure 5.1).

Intuitively, as the observation radius of POIs increases, the overall system performance should increase. As seen in Figure 5.1, the performance of agents using both G and D improve as the communication radius increases. This is because more and more information about the system is available to the agents and agents are able to communicate with additional agents as this radius increases. As the communication radius approaches 50 units, agents using difference evaluations outperform all other approaches. As the observation distance is further increased, difference evaluations outperform other methods by over 100%. This is because agents receive a clean learning signal that allows them to coordinate to optimize both their individual evaluation and the system objective simultaneously. Agents using global evaluations struggle and even when the communication radius is 100 units, their performance struggles. This is because these agents are still unable to communicate with all other agents in the system, meaning that they are frequently coordinating their actions in a suboptimal manner. If the observation distance was increased to include the entire world, global would outperform agents using random policies. It is interesting to note that even though the agent-to-agent coordination complexity increases as the observation distance increases, performance is still improved. This is because although the coordination complexity is increased, agents are able to better align their actions with the system objective instead of a "local" objective.

5.2 Combining Evaluation Functions and Stigmergy for Static Domain

In the next experiment, we examine the behavior of agents using both policy evaluation functions and stigmergy (both are implicit coordination mechanisms). Where stigmergy introduces a detectable change in the environment (the value of POIs decrease by 3% each time they are observed by a team of rovers). Stigmergy is a method that has been used to improve coordination in many domains including robot coordination (section 2.2). So we introduce stigmergy to investigate the effect of stigmergy on the cooperatively coupled rover domain. Here the performance of agents using a combination of either a Global policy evaluation function and stigmergy (G+S), Difference policy evaluation function and stigmergy (D+S), or Random (R) policy evaluation function are compared with the performance of agents using only policy evaluation functions alone (Global (G) and Difference (D) policy evaluation presented in figure 5.2).

Here, agents using policy evaluation functions (global, difference) still outperform agents using random policy evaluation functions. This is because agents using a random policy evaluation function are receiving a completely random

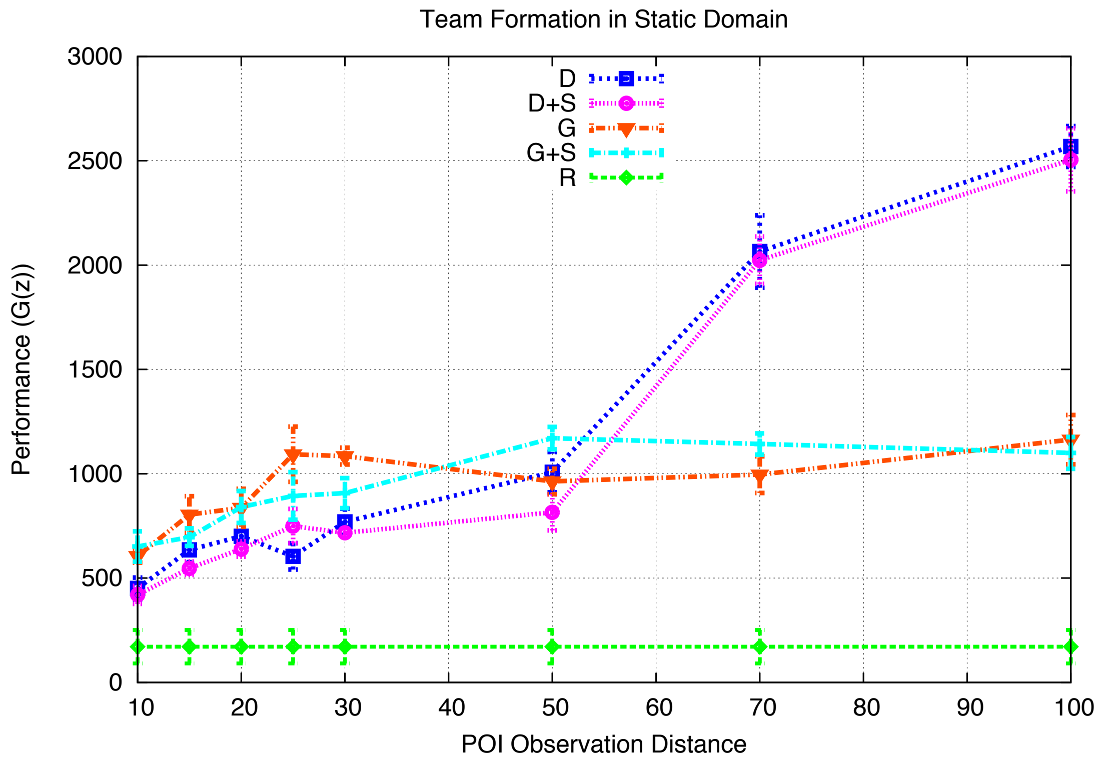

Figure 5.2: Cooperatively coupled rover domain with 40 rovers in a static environment, where teams of 3 agents are required to observe each POI. Agents using D outperform all other methods once the observation distance is above 60 units.

learning signal. In these experiments, stigmergy benefited the performance of agents using global policy evaluation functions for mid range POI observation radii ($r = [40, 90]$) because the environmental triggers were able to cue agents to move towards POIs that had been observed less (essentially encouraging the rovers to explore the environment more instead of concentrating on a few local POIs). When the POI observation radius was smaller than 40 or larger than 90, the agents encountered two specific problems. If the radii were too small (less than 40), agents had a difficult time stumbling across the POIs in general, so the benefits of stigmergy went unrealized (if rover does not observe POIs, there is no stigmergetic drop in their value). On the other hand, when the POI radius was above 90 even with stigmergy, agents using global policy evaluations received too much noise from other agents on their learning signal and were unable to coordinate their actions in order to improve performance. Agents using difference policy evaluations performed approximately the same with stigmergy as they did without, in fact, in this case, stigmergy may have harmed performance slightly. This is because agents were simultaneously using two implicit coordination mechanisms without any other form of feedback. Here, they may have received a good signal from one mechanism and a bad signal from the other mechanism. In this case, a third coordination mechanism would be valuable to serve as the tie breaker. In another experiment, we add a third coordination mechanism (IDEAS) and indeed the performance is increased.

5.3 Combining Evaluation Functions and IDEAS for Static Domain

The following experiment was conducted to investigate the effect of combining policy evaluation functions with explicit coordination (IDEAS) (section 4.2), on the performance of rovers in the cooperatively coupled rover domain. As shown in Figure 5.3 communicating IDEAS affect the agents' that used either global or difference evaluation (a small amount) and that is because of how the IDEAS are modifying the agents' states. By including agents' IDEAS we are reducing the effect of the POIs that are not in the agent's effective areas and focusing on the ones that are in close proximity. In this experiment we see almost the same behavior as we saw in previous experiment (Evaluation Functions), agents that used difference evaluation are outperforming the ones that used either global evaluation or random policy when the observation radius is higher than 50. This is because difference evaluations allow agents to receive evaluations that increase both the system evaluation as well as their individual evaluation simultaneously.

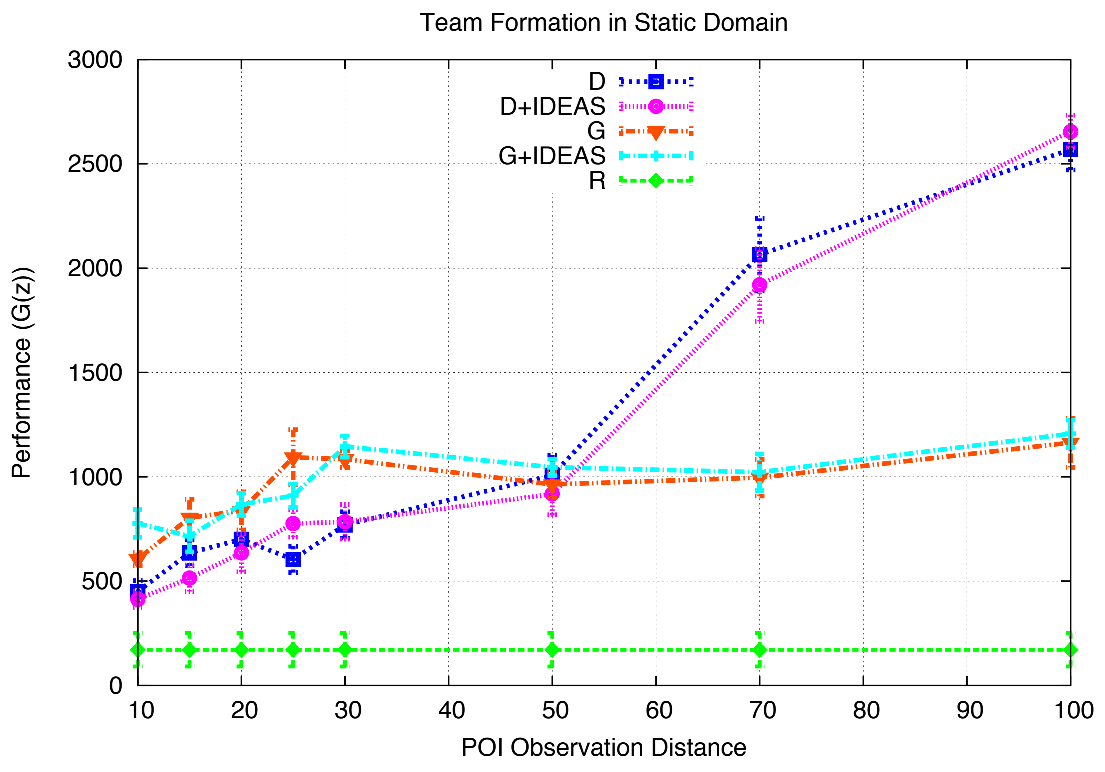

Figure 5.3: Cooperatively coupled rover domain with 40 rovers in a static environment, where teams of 3 agents are required to observe each POI. Agents using D outperform all other methods once the observation distance is above 50 units.

In this experiment, agents using random policy evaluations continue performing poorly. Agents using both global policy evaluations and difference policy evaluations continue to perform approximately the same with IDEAS as they did without IDEAS. This is because although the IDEAS alter individual agent states, emphasizing POIs that are in close proximity to the rovers, it does little to encourage overall exploration of the domain. Instead, agents using IDEAS are encouraged to head towards the POI that is closest to them. In order to address this issue, another coordination mechanism is needed and in fact stigmergy (which promotes broader exploration) is used in conjunction with ideas and policy evaluations in other experiments and results in improved performance over any method individually.

5.4 Combining Evaluation Functions, Stigmergy, and IDEAS for Static Domain

Although agents using implicit coordination via policy evaluations were able to achieve good performance, we also tested the performance of agents using a combination of coordination mechanisms in the static cooperatively coupled rover domain (Figure 5.4). Here, agents used policy evaluations (G and D, respectively), stigmergy, and the IDEAS coordination mechanism. In this case stigmergy introduces a change in the environment, the values of POIs decrease by 3% each time they are observed by a team of rovers. Experiments were also conducted to test the performance of policy evaluations with stigmergy as well as policy evaluations with IDEAS (Figures 5.2 and 5.3).

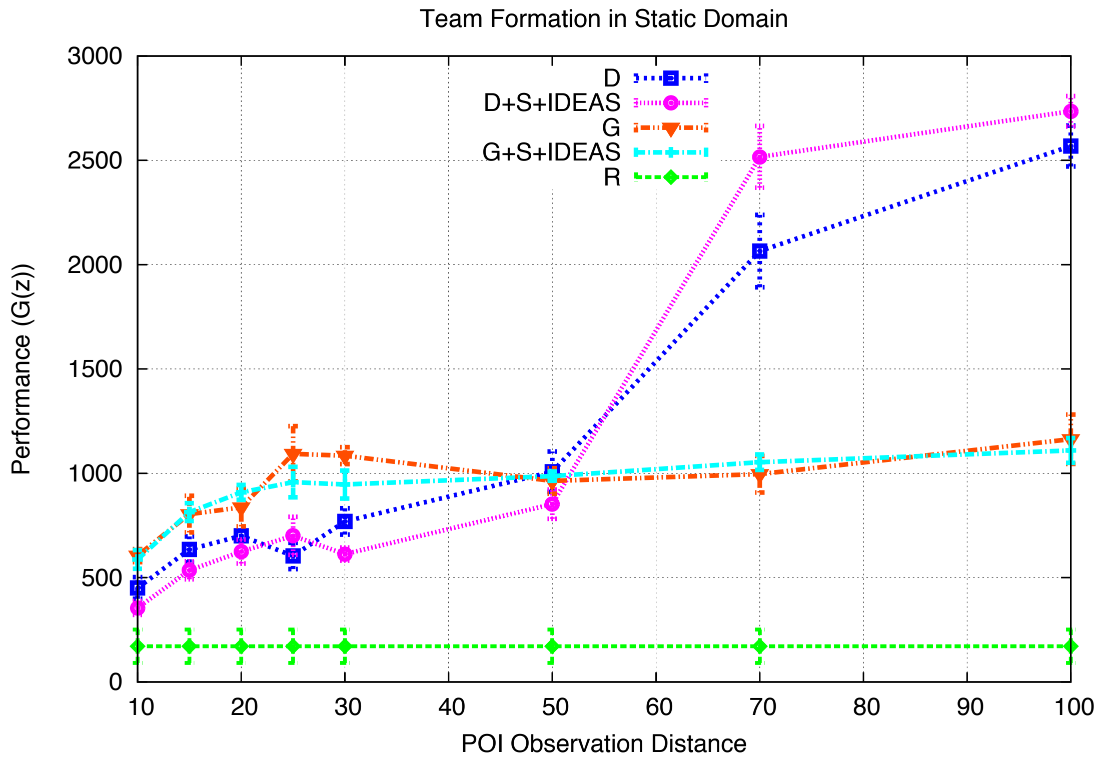

Figure 5.4: Cooperatively coupled rover domain with 40 rovers in a static environment, where teams of 3 agents are required to observe each POI. Agents using $D + S + IDEAS$ outperform all other methods once the observation distance is above 55 units.

The results in Figure 5.4 show that combining implicit (difference policy evaluations (D) and stigmergy) and explicit (IDEAS) coordination mechanisms in the $CCRD$ outperform other methods by as much as 25%. The reason that this combination works is because 1) agents using difference policy evaluations receive a quality learning signal that promotes system-centric coordination, 2) stigmergy provides an environmental cue to agents that emphasizes under observed POIs, and 3) IDEAS provide rovers with an effective "look ahead" of other rovers actions allowing them to act accordingly (agents may decide to pursue areas that are less heavily trafficked, or to attempt to form a team with another agents).

5.5 Scalability of Evaluation Functions, Stigmergy, and IDEAS in The Static Domain

In this experiment we investigated the scalability of our approach in which rovers are using a combination of policy evaluations, stigmergy, and *Intended Destination Enhanced Artificial State (IDEAS)*. Here, we scale up the cooperatively coupled rover domain (50 rovers and 80 POIs). The number of rovers (x-axis), number of POIs, and domain dimensions were all scaled up proportionally. Teams of $M = 3$ are still required to observe each POI as was the case in all previous experiments. Intuitively, as the number of rovers increases in this domain, the rover-to-rover coordination complexity increases exponentially. This increased coordination complexity makes this experiment necessary to demonstrate the effectiveness of our approach (combining both implicit and explicit coordination) at improving coordination in multiagent systems.

In these experiments, the performance of agents using random policy evaluation functions increases slightly as the system size increases. This is because more agents flood the system and when they all behave randomly, they end up stumbling across more POIs. Similarly, agents using global and difference policy evaluations both have improving performance as the number of rovers within the

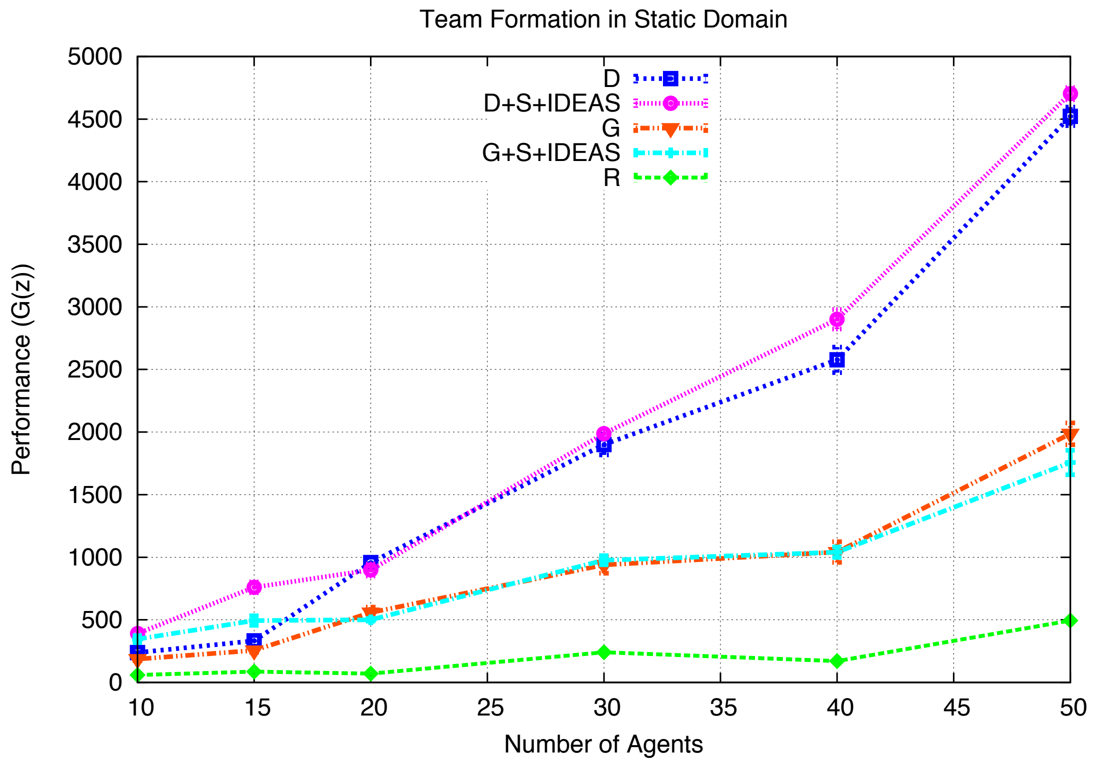

Figure 5.5: Investigating the scalability of evaluation functions (G and D), combination of evaluation function, Stigmergy and IDEAS ($D + S + IDEAS$ and $G + S + IDEAS$) in cooperatively coupled rover domain and static environment. Agents using $D + S + IDEAS$ outperform all other methods almost every where.

system increases. This improvement is less dramatic for agents using Global policy evaluation (G) than for agents using Difference evaluation (D) (agents using D have a sharper increase in learning compared to other methods with increasing scale). This is because the improvement with agents using G is primarily an artifact of the scaling (similar to random agents), more agents exploring stumble across more POIs. Agents using G do better than random agents because they are intentional in their wandering, but they still perform much worse than agents using D. This is because they still have a lot of noise on their learning signal which causes them to struggle with coordination (coordination gets more and more difficult for agents using G as scaling increases). Agents using D outperform most other methods with scaling, and agents using our approach ($D + S + IDEAS$) perform even better. This is because our approach couples the benefits of three different coordination mechanisms, each of which was designed to improve coordination in multiagent systems. Additionally, the benefits of these three coordination mechanisms are mutually beneficial. Agent-specific policy evaluation functions (D) encourage agents to collaborate to improve the system objective, stigmergy encourages exploration by reducing the value of areas that have been heavily explored, and IDEAS encourages agents to focus on particular environmental POIs.

5.6 Extension to Dynamic Domain

Now that we have demonstrated the benefits of combining implicit and explicit coordination mechanisms, we want to extend this one step further by demonstrating the robustness of such an approach. In these experiments, agents in the cooperatively coupled rover domain (with 3 rover teams) have a dynamic environment, where the location and values of POIs change every episode. Dynamically changing the environment each learning episode increases learning difficulty and makes it harder for rovers to coordinate their policies.

As seen in Figure 5.6, agents using global policy evaluations (both with and without additional coordination mechanisms such as stigmergy and IDEAS) per-

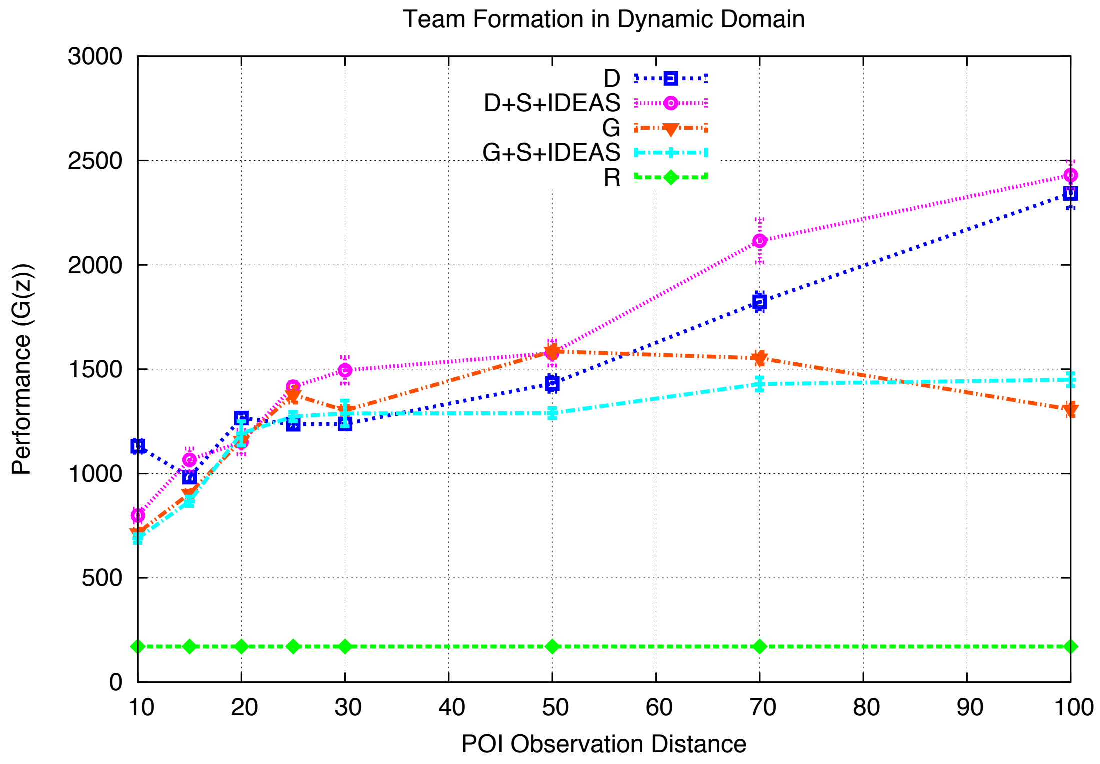

Figure 5.6: Cooperatively coupled rover domain with 40 rovers in a dynamic environment, where teams of 3 agents are needed to observe each POI. As seen, agents using $D+S+IDEAS$ outperform all other methods once the observation distance is above 60 units.

form approximately 50% better than they did in the static cooperatively coupled rover domain. This is because the agents are effectively learning policies that randomly wander and when the POIs are randomly placed throughout the environment every episode, so over a number of episodes, they end up running across more POIs on average. However, they still perform poorly compared to agents using difference policy evaluations. This is because difference policy evaluations are able to account for the "randomness" of the environment and design agent policies that compensate for dynamic environmental factors. Agents using difference policy evaluations coupled with stigmergy and IDEAS outperform agents using only difference policy evaluations by approximately 25% under different observation restrictions, and they outperform all other methods by as much as 60%.

5.7 Analysis of Results

In these experiments, we empirically tested the performance of a set of learning agents using different implicit and explicit coordination mechanisms in both static and dynamic environmental settings. In particular, we tested the impact of three individual coordination methods on the overall system performance, as well as an amalgamation of all three. As seen from the results, combining different implicit and explicit coordination mechanisms can be either beneficial or harmful to system performance. The key is to select a set of coordination mechanisms that are mutually beneficial to the overall system performance. In these experiments, we used stigmergy, policy evaluation functions, and our own explicit coordination mechanism *Intended Destination Enhanced Artificial State* (*IDEAS*). In all experiments, learning agents utilized policy evaluation functions as a base coordination mechanism which, this was required for learning, and was then tested with and without additional coordination mechanisms.

As seen in Figure 5.1 it is clear to see that the selection of policy evaluation function can dramatically impact the performance. Although coupling the policy evaluations implicitly encourages coordination, it does not guarantee perfect coordination. As seen by the rovers using global policy evaluations G receive feedback that is coupled to all other agents, unfortunately it is very noisy with respect to the actions of other agents. This makes it very difficult for agents to learn because they cannot necessarily distinguish their own impact on the system performance (good or bad) from the movement of all other agents. In this case, G ends up being a fairly noisy coordination mechanism. An alternative to this is to use difference policy evaluation functions D, where these policy evaluations address the noise issues associated with G. Here, the agents receive a signal that is representative of their individual contribution on the system performance. This removes the noise on the agents learning signal and results in better coordination than using G alone. Although the performance of agents using policy evaluations was good by itself, we wanted to see how additional coordination mechanisms would impact this performance. In particular, we wanted to determine whether or not it would be possible to incorporate additional coordination mechanisms in conjunction with policy evaluations in such a way that they improve system performance. In order to test this, we tested various combinations of adding stigmergy and IDEAS to agents using policy evaluation functions.

It would seem that if one coordination mechanism is "good", then two coordination mechanisms would be better. However, our next two sets of results (Figures 5.2 and 5.3) suggest otherwise. They suggest that combining different coordination mechanisms can result in worse coordination than either method individually. This occurs because when two coordination mechanisms have conflicting views about what actions the agent should take, the agent ends up attempting to strike a balance between the recommended actions of each mechanism and can end up taking actions that impede performance. These results clearly show that simply combining multiple coordination mechanisms together will not necessarily result in improved performance. Instead, coordination mechanisms should be intelligently selected based upon their compatibilities.

In these experiments, combining stigmergy with policy evaluations led to decreased performance under highly moderate amounts of observability because it encouraged agents to scout out the rest of the environment (which coincided with what the global policy evaluations were encouraging them to do). On the other hand, with large observation radii the there was simply too much noise on the global agents learning signal which caused the action recommendations from both coordination mechanisms to conflict, resulting in worse performance. A similar result occurred when agents were using policy evaluations and IDEAS. Over different ranges of observability, the two coordination methods frequently had conflicting action recommendations, resulting in worse performance than either method combined. In order to address this, we combined policy evaluation functions with stigmergy and IDEAS concurrently in order to improve coordination. This combination was particularly effective because when any two coordination mechanisms disagreed on their action recommendation, the third coordination mechanism was able to serve as a “tie breaker”. When the majority of the coordination functions agreed upon an action, it turned out to be more likely to be beneficial to the system performance.

Chapter 6 – Conclusion

Coordinating the actions of disparate agents such that they collectively complete a complex task is a key problem that must be addressed in order to advance the field of multiagent systems. Although many implicit and explicit mechanisms for solving such coordination problems exist, they are frequently unable to fully address the coordination issues involved due to limited observation and communication restrictions. In order to improve performance of these methods, we selected implicit and explicit coordination mechanisms whose benefits were complementary under limited observation and communication restrictions. In particular, we utilized a combination of two implicit coordination mechanisms policy evaluations and stigmergy and one novel explicit coordination mechanism, IDEAS, in the Cooperatively Coupled Rover Domain under limited observability. Overall, combining evaluation functions, stigmergy, and IDEAS coordination mechanisms resulted in up to 25% improved performance over other approaches.

Combining the benefits of these coordination mechanisms enabled improved performance under varying observation restrictions because the mechanisms were complementary. Coupling policy evaluations enables agents to attempt to work together as a collective unit, what is good for an individual is good for the team. However, under limited observability, agents receive limited information and their policy evaluations become less reflective of the overall team performance and instead emphasize the performance in their local region. This is addressed by allowing agents to passively communicate their IDEAS (explicit coordination), which allows agents in different areas to implicitly "skip" information across the system to each other, improving their ability to globally coordinate their actions (agents may effectively coordinate through interacting with other agents, though they may never interact directly). Finally, stigmergy provides an environmental que that impacts agents locally in a way that has global repercussions. As POIs were observed, their values decreased. This means that if a POI has been heavily observed in the past, although there are currently no rovers near it, as a new rover comes across it they will know to look elsewhere for a higher value POI, effectively encouraging the rovers to disperse and search other areas of the domain.

Although we used specific coordination mechanisms, there are undoubtably more combinations of implicit and explicit coordination mechanisms that will improve performance in many multiagent system domains. In general, implicit coordination mechanisms rely heavily upon agents' observation of the environment and tend to be limited by observation restrictions, while explicit coordination mechanisms rely heavily upon direct agent-to-agent information sharing and negotiation and are typically limited by communication restrictions. In most real-world multiagent system domains both observation and communication restrictions exist and when they do, a combination of implicit and explicit coordination mechanisms will likely be advantageous over either method individually.

This work showed that combining explicit and implicit coordination mechanisms has the potential to be beneficial or harmful to the overall system performance. In order to improve system performance through combining such coordination mechanisms, the mechanisms used must be carefully selected such that they are compatible. This work laid initial groundwork on the feasibility and benefit of combining implicit and explicit mechanisms. Future work will need to define ways of classifying coordination mechanisms in order to determine their compatibility. Such work needs to address the following problem: Given the complete set of coordination mechanisms that exists throughout the literature, how to select compatible subsets of these mechanisms that can work well together and improve performance. Ideally, this work would define metrics for classifying coordination mechanisms as well as specific means for using these metrics to: 1) select useful combinations of existing coordination mechanisms, or 2) design novel coordination mechanisms that optimize performance with respect to these criteria. These extensions would help advancements in coordination techniques within the multi agent community at large.

Bibliography

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

[2] Adrian Agogino and Kagan Tumer. Multi agent reward analysis for learning in noisy domains. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 81–88, Utrecht, Netherlands, July 2005.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2007.

[4] Gregory Bonnet and Catherine Tessier. Collaboration among a satellite swarm. *Proceedings of the 6th international joint conference on autonomous agents and multiagent systems*, page 1585, 2007.

[5] Gregory Bonnet and Catherine Tessier. Coordination despite constrained communications: a satellite constellation case. *3rd National Conference on Control Architectures of Robots*, pages 89–100, 2008.

[6] Sylvain Bouvert, Michel Lemaitre, Helene Fargier, and Jermoe Lang. Allocation of indivisible goods: A general model and some complexity results. *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems*, 2005.

[7] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C - Applications and Reviews*, (156-172), 2008.

[8] Tianping Chen and Robert Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, 1995.

[9] Mitch Colby, Ehsan Nasroullahi, and Kagan Tumer. Optimizing ballast design of wave energy converters using evolutionary algorithms. In *GECCO'11*, pages 1739–1746, July 2011.

[10] Sylvain Damiani, Gerard Verfaillie, and Marie Charmeau. An earth watching satellite constellation: How to manage a team of watching agents with limited communications. *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems*, pages 455–462, 2005.

[11] Paul E. Dunne, Michael Wooldridge, and Michael Laurence. The complexity of contract negotiation. *Artificial Intelligence*, pages 23–46, 2005.

[12] Edmund H. Durfee. *Distributed Problem Solving and Planning.* Springer-Verlag New York, Inc, 2001.

[13] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[14] Al Globus, Greg Hornby, Derek Linden, and Jason Lohn. Automated antenna design with evolutionary algorithms. AAIA Space 2006 Conference, 2006.

[15] Uli Grasemann, Daniel Stronger, and Peter Stone. A neural network-based approach to robot motion control. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup-2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Artificial Intelligence*, pages 480–87. Springer Verlag, Berlin, 2008.

[16] Karuna Hadeli, Paul Valckenaers, Constantin B. Zamfirescu, Hendrik Van Brussel, Bart Saint Germain, Tom Holvoet, and Elke Steegmans. Self-organising in multi-agent coordination and control using stigmergy. In *Engineering Self-Organising Systems*, pages 105–123, 2003.

[17] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. In *The Knowledge Engineering Review*, pages 281–316. Cambridge University Press, 2005.

[18] J.L. Hudson, M. Kube, R.A. Adomaitis, George I. Kevrekidis, Alan Lapedes, and Robert Farbar. Nonlinear signal processing using neural networks: Prediction and system modeling. In *Los Alamos National Laboratory Theoretical Division*, pages 2075–2081, July 1987.

[19] Li Jiang and Da you Liu. A survey of multi-agent coordination. In *IC-AI*, pages 65–71, 2006.

[20] Chris Jones and Maja J. Mataric. Adaptive division of labor in large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, pages 1969–1974, Las Vegas, NV, July 2003.

[21] Matt Knudson and Kagan Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 27–134, Portland, OR, July 2010.

[22] Matt Knudson and Kagan Tumer. Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems*, pages 410–420, June 2011.

[23] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 1997.

[24] Piotr Krysta, Tomasz Michalak, Tuomas Sandholm, and Michael Wooldridge. Combinatorial auctions with externalities. *9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1471–1472, May 2010.

[25] Narendra KS and Parthasarathy K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, pages 4–27, March 1990.

[26] Rainer Leupers and Fabian David. A uniform optimization technique for offset assignment problems. In *11th Int. symp. on system synthesis (ISSS)*, pages 3–8, 1998.

[27] Peter X. Liu, Ming J. Zuo, and Max Q.-H. Meng. Using neural network function approximation for optimal design of continuous-state parallel-series systems. *Computers & Operations research*, 30:339–352, July 2001.

[28] Ndedi Monekosso, Paolo Remagnino, and Adam Szarowicz. An improved q-learning algorithm using synthetic pheromones. *Lecture Notes in Computer Science*, 2296, 2001.

[29] Ranjit Nair, Milind Tambe, and Stacy Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems*, pages 552–559, 2003.

[30] Allison M. Okamura, Maja J. Mataric, and Henrik I. Christensen. Medical and health-care robotics. *Robotics and Automation Magazine, IEEE*, 17(3):26–37, 2010.

[31] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[32] Valentin Robu, Ioannis Vetsikas, Enrico Gerding, and Nicholas Jennings. Flexibly priced options: A new mechanism for sequential auction markets with complementary goods (extended abstract). *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1485–1486, May 2010.

[33] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach (Third Edition).* Prentice Hall, Pearson Publication Inc, 2010.

[34] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robotics*, 8(3):345–383, July 2000.

[35] R.S. Sutton and A.G. Barto. *Reinforcement learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[36] Milind Tambe. Implementing agent teams in dynamic multi-agent environments. *Applied Artificial Intelligence*, pages 85–101, 1997.

[37] Kagan Tumer and Aadrian Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *n Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pages 591–598, Washington, DC, June 2005. ACM Press.

[38] Kagan Tumer and Adrian Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 330–337, Honolulu, HI, May 2007.

[39] Kagan Tumer and David Wolpert, editors. *Collectives and the Design of Complex Systems.* Springer, New York, 2004.

[40] David H. Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2):265–279, 2001.

[41] M. Wooldridge. *An Introduction to MultiAgent Systems.* John Wiley and Sons Ltd, 2002.

[42] Erfu Yang and Dongbing Gu. A survey on multiagent reinforcement learning towards multi-robot systems. *Proceedings of IEEE Symposium on Computational Intelligence and Games*, pages 4–6, April 2005.