Shaping Fitness Functions for Coevolving Cooperative Multiagent Systems

Mitchell Colby
Oregon State University
442 Rogers Hall
Corvallis, OR 97331
colbym@engr.orst.edu

Kagan Tumer
Oregon State University
204 Rogers Hall
Corvallis, OR 97331
kagan.tumer@oregonstate.edu

ABSTRACT

Coevolution is a natural approach to evolve teams of agents which must cooperate to achieve some system objective. However, in many coevolutionary approaches, credit assignment is often subjective and context dependent, as the fitness of an individual agent strongly depends on the actions of the agents with which it collaborates. In order to alleviate this problem, we introduce a cooperative coevolutionary algorithm which biases the evolutionary search as well as shapes agent fitness functions to reward behavior that benefits the system. More specifically, we bias the search using a hall of fame approximation of optimal collaborators, and we shape the agent fitness using the difference evaluation function. Our results show that shaping agent fitness with the difference evaluation improves system performance by up to 50%, and adding an additional fitness bias can improve performance by up to 75%.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Distributed Systems

General Terms

Algorithms, Experimentation

Keywords

Co-evolution, Multiagent learning

1. INTRODUCTION

Coordinating multiple agents in order to achieve some system objective is an important area of research, and is critical in many domains including rover coordinaton, air traffic control, search and rescue, and unmanned aerial vehicle coordination[1, 2]. One approach to achieving coordination is the use of Cooperative Coevolutionary Algorithms (CCEAs), which involve evolving multiple populations simultaneously and evaluating the fitness of individuals based on the individual's interactions with other agents in the system [10]. By evolving multiple populations at once, CCEAs project the search space into multiple, smaller, search spaces. Coevolving agents have only a fraction of the total state space available to them. Further, agents' fitness assignments are context dependent and are influenced by agents from other populations. Thus, CCEAs have the tendency to create agents which are capable of performing adequately with a wide range of collaborators, rather than specializing to perform well with the best set of collaborators; in other words, CCEAs often produce stable, rather than optimal solutions [3, 4, 10]. In order to make CCEAs a viable option for coordinating multiagent systems, it is critical that steps be taken to achieve optimal coordination policies.

There have been multiple approaches to address the issues of suboptimal stable policies. One approach involves shaping local fitness functions to align with the system evaluation function. Proper shaping of these fitness functions leads to faster learning and more optimal policies [1, 8]. Another approach is to bias searches based on the notion of optimal teammates [10], which involves estimating agent utilities as if they were paired with optimal collaborators. Although these biased searches generally increase the effectiveness of CCEAs, an issue with this approach is that in complex domains, estimating system evaluations as if optimal collaborators were present is exceedingly difficult. Other methods involve altering the evolutionary mechanisms in CEAs in order to optimize CEA performance, such as lenient learners or hall of fame methods [11, 13].

In this work we address the suboptimal solutions created by CCEAs by shaping fitness functions and approximating optimal collaborators. Two domains are used to test whether these problems are addressed adequately. First, a *scatter domain*, which involves agents moving in a two dimensional plane in order to become as "spread out" as possible. Secondly, a *rover domain* involving robots gathering data from points of interest tested the algorithms on a more real-world problem.

The contribution of this paper is a CCEA which:

- Shapes fitness functions using the Difference Utility
- Biases search with optimal collaborators using a hall of fame approach, which is much less complex than approximating optimal collaborators in an ad-hoc manner

In our experiments, this algorithm outperformed a CCEA using the system evaluation to assign fitness by an average of 48.7%. The remainder of this paper is organized as follows: section 2 describes related work and background information. Section 3 describes the domains analyzed. Section 4 describes the algorithms used in this research. Section 5

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

gives the experimental results. Finally, Section 6 discusses the results and potential areas for future research.

2. EVOLUTION AND COEVOLUTION

Evolutionary Algorithms (EAs) are a class of stochastic search algorithms which often outperform classical optimization techniques, particularly in complex domains where gradient information is not available [6]. An evolutionary algorithm typically contains three basic mechanisms: solution generation, mutation, and selection. These mechanisms are used on an initial set of candidate solutions, or a population to generate new solutions and retain solutions that show improvement. Simple EAs are excellent tools, but need to be modified to be applicable to large multiagent search problems. One such modification is *coevolution*, where multiple populations evolve simultaneously in order to develop policies for interacting agents. The following sections introduce coevolutionary algorithms, and approaches to optimize the output of these algorithms.

Coevolutionary Algorithms (CEAs) are an extension of evolutionary algorithms and are often well-suited for multiagent domains [5]. In a CEA, the fitness of an individual is based on its interactions with other agents it collaborates with. Thus, assessing the fitness of each agent is context-sensitive and subjective [10]. In *competitive* coevolution, individuals benefit when other agents fail. In *cooperative* coevolution, individuals succeed or fail as a team. This paper is focused on Cooperative Coevolutionary Algorithms (CCEAs). One of the advantages to coevolution is that the algorithm only needs to search subspaces of the state space, rather than the entire state space. This reduced state space often makes the learning process simpler for the cooperating agents. However, these simpler subspaces represent a large loss in information; the consequence of this is that the policies obtained by using these state projections are strongly influenced by other populations. The result is that agents evolve to partner well with a broad range of other agents, rather than evolving to form optimal partnerships [10]. Thus, in addition to trying to decrease the complexity of the learning process, research in coevolution aims to achieve optimal policies rather than stable ones.

Cooperative Coevolutionary Algorithms.

CCEAs are a natural approach in domains where agents succeed or fail as a team [12]. In CCEAs, distinct populations evolve simultaneously, and agents from these populations collaborate to reach good system solutions. One issue with CCEAs is that they tend to favor stable solutions, rather than optimal solutions [16]. This phenomena occurs because the different evolving populations adapt to each other, rather than adapting to form an optimal policy. Another issue that arises with CCEAs is the problem of credit assignment. Since the agents succeed or fail as a team, the fitness of each agent becomes subjective and context-dependent (e.g. an agent might be a "good" agent, but the agents it collaborates with are "bad," and the objective isn't reached. In this case, the "good" agent may be perceived as "bad") [16]. Generally speaking, research in CCEAs involves either making a more computationally efficient algorithm or reaching better solutions (avoiding suboptimal equilibria) [1]. The following sections outline some methods that have been developed to mitigate problems associated with CCEAs.

Biasing Coevolutionary Search

Panait *et al.*[10] biased the evolutionary search in order to find optimal solutions, rather than becoming stuck in local minima. In standard coevolution, a single agent is rated on how well the team does as a whole; every agent in the team gets equal credit. This results in an unfavorable signal-to-noise ratio, as each agent is unaware of its individual contribution to the team's performance. In a *Biased Cooperative Coevolutionary Algorithm* (BCCEA), the fitness of an individual is based partly on its interactions with other agents (as in usual CCEAs), and partly on an estimate of the best possible fitness for that individual if it is partnered with optimal collaborators. By biasing the coevolutionary search in this manner, the algorithm is better able to search for optimal policies, rather than stable policies, because each agent receives feedback related to its performance, rather than solely the team's performance. One issue with this approach is that estimating how an agent would perform with optimal collaborators is a nontrivial task, and becomes increasingly difficult in complex domains.

Fitness Function Shaping

Hoen and De Jong shaped the utilities of the agents so as to contribute to the system evaluation, such that an agent maximizing its individual utility would act to increase the system evaluation. By shaping agent fitnesses, the learning process is sped up considerably [8]. This work is similar to that of Agogino and Tumer, and Knudson and Tumer, who utilized difference evaluations as fitness functions to evolve coordination in multiagent systems [1, 9]. The *Difference Evaluation* is defined as:

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

where $G(z)$ is the total system evaluation, z_{-i} are all the states on which agent i has no effect, and c_i is the *counterfactual* term, which is a fixed vector to replace the effects of agent i. Intuitively, the second term in Equation 1 evaluates the fitness of the system without the effects of agent i, so the difference evaluation gives agent i's contribution to the system evaluation [1]. By shaping fitness functions such that each agent's fitness is related to the individual's contribution to team performance, the signal-to-noise ratio is improved considerably.

Evolving Teams

Coevolution is frequently a good algorithm to use in multiagent systems with heterogeneous agents. Haynes *et al.* used genetic algorithms to evolve *teams* of predators [7]. Rather than evolving single predators, one member of the population consisted of four predators. In this manner, the predators can evolve to cooperate. By evolving teams rather than individual agents, communication is not required in the domain; in place of communication, the team of predators evolves to act as if they know the other agents' future actions based on the state of the system [15] Though effective, these approaches are designed for small teams, and become slow to converge when scaled to large multiagent systems.

Hall of Fame

Rosin and Belew[13] introduced the concept of the *Hall of Fame* for competitive coevolution, in which top individuals are saved in order to test against in later generations. There

are two reasons why it is beneficial to save these top individuals. First, keeping top individuals contributes genetic information to later generations, which is imperative when conducting any evolutionary algorithm. Secondly, by keeping top individuals, new individuals in later generations may be tested against the hall of fame members. This concept can be extended to CCEAs by keeping hall of fame *teams*, rather than hall of fame *individuals*. In this manner, desirable genotypes won't be lost if they perform poorly for a few generations due to stochasticity.

Leniency

Panait *et al.*[11] introduced the concept of *lenient* learners in coevolutionary algorithms, which are agents which forgive possible mismatched teammate actions that result in poor team performance. Using lenient learners in coevolution is shown to provide learners with more accurate information about their policies, which increases the likelihood of converging to an optimal solution. Leniency is achieved by pairing agents with multiple sets of collaborators, and taking the highest team fitness achieved from all runs. This lowers the likelihood that a learning agent will receive poor feedback simply because it was paired with suboptimal teammates.

3. EVALUATION DOMAINS

In this section, we introduce the two problems analyzed in this work, and provide a detailed explanation of the system dynamics and evaluation functions used in each domain.

3.1 Scatter Domain

The scatter domain used in this paper is a variant of the mixing problem [14]. In the scatter domain, a team of agents on a two dimensional plane aim to move around and configure themselves to be as "spread out" as possible (Figure 1). The world is continuous, as are the actions of each agent. Each agent calculates the distance between itself and the closest teammate using the standard Euclidian distance. So, if there are N agents, each agent calculates how far away the closest agent is at any time t using:

$$\delta_i(t) = \min_j \left\{ \sqrt{(x_{i,t} - x_{j,t})^2 + (y_{i,t} - y_{j,t})^2} \middle| i \neq j \right\} \quad (2)$$

where $\{x_{i,t}, y_{i,t}\}$ is agent i's x and y position in the world at time t, and j is used to index all agents other than agent i. The total state of the system z is the set of all agent positions

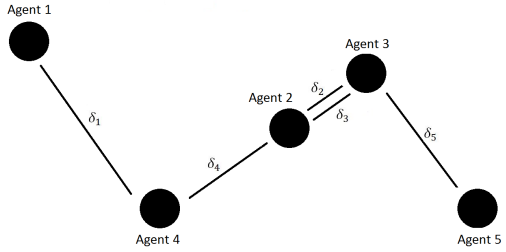

Figure 1: Scatter Domain Representation. Each agent i calculates the distance from itself to the closest agent as δ_i. The global utility is the average of all of these distances. The goal in this domain is for the agents to be as "spread out" as possible.

in the world. At each time step in an episode, an agent takes two actions Δ_x and Δ_y, corresponding to its x and y movements, respectively. The magnitude of these actions are bounded by some upper limit Δ_{max}, which requires that the agents take multiple actions over multiple time steps in order to traverse the domain. The system evaluation function for the scatter domain with N agents is the average minimum distance between agents, given by:

$$G(z(t)) = \frac{\sum_{i=1}^{N} \delta_i(t)}{N} \quad (3)$$

Thus, maximizing the average minimum distance between agents will result in maximizing the system evaluation.

3.2 Rover Domain

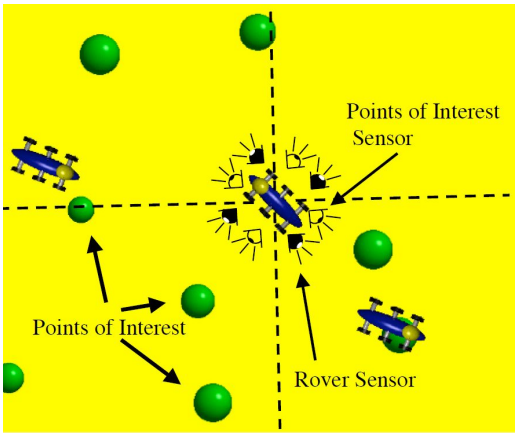

Figure 2: Rover Domain Representation. Each rover senses the closest rover and POI from each of its four sensing quadrants. The rovers must coordinate in order to effectively observe the POIs.

In the rover problem, a collective of rovers on a two dimensional plane aim to observe points of interest (POIs) scattered across the domain (Figure 2). Each POI has an associated value, and each observation of a POI made by a rover yields an observation value which is inversely proportional to the distance that the rover is from the POI. The distance metric used in this domain is the squared Euclidian norm, bounded by a minimum observation value:

$$\delta(x, y) = min\left\{||x - y||^2, \delta^2_{min}\right\} \quad (4)$$

The objective of the rovers is to maximize the observation values of the POIs over the course of an episode, and the system evaluation is calculated as:

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

where V_j is the value associated with POI j, L_j is the location of POI j, and $L_{i,t}$ is the location of the ith rover at time t.

Although any rover may observe any POI, the system evaluation only takes into account the closest observation made for each POI. In this instantiation of the rover domain, the POI locations are static throughout each experiment. The rovers have eight total sensors, two sensors per quadrant. The rovers sense the closest POI and rover in each of the four quadrants, and these eight readings compose the controller

inputs. The rovers must coordinate in order to achieve high POI coverage. An increasing system evaluation corresponds to better observation coverage of the POIs. As in the scatter domain problem, at each time step the rovers take two actions Δ_x and Δ_y, corresponding to their x and y movements; the magnitude of these actions are bounded by some value Δ_{max}.

4. ALGORITHMS

In this section we provide detailed explanations of the four algorithms investigated in this research, which are:

1. Standard CCEA using system evaluation
2. CCEA using the difference evaluation
3. CCEA using lenient learners and the difference evaluation
4. CCEA using the hall of fame and the difference evaluation

The standard CCEA using the system evaluation is a "standard" CCEA algorithm we use a baseline to assess performance. The second and third algorithms are modifications of existing algorithms which address the problems of suboptimal convergence. The final algorithm is a modification of the hall of fame algorithm combined with the difference evaluation, which also aims to address suboptimal convergence and is the main contribution of this paper.

4.1 Standard CCEA

In the standard CCEA, N coevolving populations of neural networks are utilized to form teams comprised of M agents. In the most general case, M is equal to N. One member of each population is extracted, and these agents are combined to form a team which operate in the problem domain. Each population is initially comprised of k neural networks, randomly initialized. At each generation, k successor networks are generated in each population, which are mutated versions of the parent networks. Then, $2k$ teams of M agents are formed by taking agents from each population and placing these agents into a team. The performance of each of the teams is then evaluated in the domain, and the fitness of every agent in the team is set according to the team's performance. Next, k networks from each population are selected to proceed to the next generation, with the fitness of a network influencing its selection probability. This process is repeated for a set number of generations. The standard CCEA is detailed in Algorithm 1.

In the standard CCEA, each member of a team receives equal credit for that team's performance. This form of credit assignment results in agents' fitness values to be heavily dependent upon the performance of teammates, because each member of the team receives equal credit for the team's performance. The standard CCEA is used as the baseline algorithmm, and serves as a comparison for the other algorithms considered.

Initialize N populations of k neural networks
foreach *Generation* **do**
  foreach *Population* **do**
    produce k successor solutions
    mutate successor solutions
  end
  for $i = 1 \rightarrow 2k$ **do**
    randomly select one agent from each population
    add agents to team T_i
    simulate T_i in domain
    assign fitness to each agent in T_i using $G(z)$
  end
  foreach *Population* **do**
    select k networks using ϵ-greedy
  end
end

Algorithm 1: Standard CCEA (See Section 5 for parameters)

4.2 CCEA with the Difference Evaluation

The CCEA with the difference evaluation is carried out in a similar manner to the standard CCEA, except that when a team of agents is evaluated, the fitness of each agent is calculated with the difference evaluation, rather than the system evaluation. Thus, the fitness of each agent of a team is calculated as that agents's contribution to the team's performance, rather than the system evaluation itself. The CCEA with the difference evaluation is equivalent to the CCEA detailed in Algorithm 1, except at the fitness assignment stage, the fitness of each agent is calculated with the difference evaluation rather than the system evaluation.

Thus, the key difference between the CCEA using the difference evaluation and the standard CCEA is credit assignment for the agents. By utilizing the difference evaluation to assign fitness, the fitness of each agent becomes less dependent upon the actions of its teammates. Below, we derive the difference evaluation for the two domains used in this work.

Scatter Domain.

Directly computing the different evaluation using Equation 1 corresponds to simply leaving out agent j from the computation:

$$D_j(z,t) = \frac{\sum_{i \neq j} \delta_i(t)}{N-1} \tag{6}$$

However, this evaluation always increases the system evaluation, because of the nature of the system evaluation function. As the goal in this domain is to maximize average distance between agents, removing any agent will always have a positive effect on the system evaluation. As such agents will need to distinguish between very small positive variations, making the evaluation function in Equation 6 a poor choice for agent fitness function. Instead, in this work, we introduce a *default agent effect*. To compute agent j's fitness then, we replace agent j with this default agent, which yields:

$$D_j = \frac{\sum_{i \neq j} \delta_i(t) + \delta_{def}(t)}{N} \tag{7}$$

where $\delta_{def}(t)$ is a distance associated with the default agent. This distance is set at the beginning of an experiment, and remains constant for each individual calculation of the difference evaluation. This default agent distance corresponds to the c_i (*counterfactual*) term in Equation 1.

Rover Domain.

For the rover problem, the difference evaluation is calcu-

lated by directly applying Equation 1 to Equation 5:

$$D_i(L) = \sum_t \sum_j I_{j,i,t}(z) \left[\frac{V_j}{\delta(L_j, L_{i,t})} - \frac{V_j}{\delta(L_j, L_{k_j,t})} \right] \quad (8)$$

where k_j is the second closest rover to POI j, and $I_{j,i,t}(z)$ is an indicator function which returns 1.0 if and only if rover i is the closest rover to POI j at time t. If rover i is not the closest rover to any POI at time t, then its difference evaluation is zero, indicating that the rover is not contributing to the system utility at time t.

4.3 CCEA with Lenient Learners and Difference Evaluation

The CCEA with lenient learners and the difference evaluation is carried out in a similar manner to the CCEA with the difference evaluation, except that each agent is tested with multiple sets of collaborators (i.e. the agent will be placed in multiple teams), and the highest fitness achieved is the fitness assigned to that agent. As the algorithm progresses, agents become less lenient learners; that is, they are tested against fewer sets of collaborators. The CCEA with lenient learners and the difference evaluation is detailed in Algorithm 2.

```
Initialize N populations of k neural networks
foreach Generation do
    foreach Population do
        produce k successor solutions
        mutate successor solutions
    end
    for i = 1 → 2k · m do
        randomly select one agent from each population
        add agents to team T_i
        simulate T_i in domain
        assign fitness to each agent in T_i using D(z)
    end
    foreach Population do
        foreach Member do
            fitness ← maximum fitness attained
        end
    end
    foreach Population do
        select k networks using ε-greedy
    end
end
```

Algorithm 2: Lenient CCEA using Difference Evaluation (See Section 5 for parameters)

It is important to note that in Algorithm 2, each member of each population is selected exactly m times, and the value of m is decreased as the algorithm progresses. This corresponds to agents being lenient in the early stages of evolution, and becoming less and less lenient as evolutionary time passes. By incorporating leniency and the difference evaluation into the CCEA, the effects of an agent's teammates on the fitness of that agent are minimized.

4.4 CCEA with Hall of Fame and Difference Evaluation

As noted in Section 2, CCEAs have been shown to provide better solutions when the fitness of an individual is based partly on how it performs with its team, and partly on how it would perform if it were paired with optimal collaborators. However, estimating the fitness of an agent paired with optimal collaborators is a difficult task, especially in complex domains. Rather than estimating what optimal collaborators would be for a particular agent, the *hall of fame* method is altered to estimate the behavior of optimal collaborators. The CCEA with the hall of fame and difference evaluation is carried out in a similar manner to the CCEA with the difference evaluation, except that the fitness assignment is altered. At the end of each generation, the team that achieves the highest system evaluation is compared against the hall of fame members. If that team achieved a higher system evaluation than all of the hall of fame members, then that team is added to the hall of fame. When assigning fitness to each agent of a team, the difference evaluation of that agent is calculated, as well as the difference utility of that agent when it replaces an agent from the best hall of fame team. The performance of the best hall of fame team is non-decreasing with respect to evolutionary time, so this team approaches the optimal team as the CCEA progresses. The difference evaluation of an agent when compared with the hall of fame team is calculated as:

$$D_{HOF,i} = G_{HOF+i} - G_{HOF} \quad (9)$$

where G_{HOF+i} is the system evaluation of the best hall of fame team when agent i replaces the corresponding member of the hall of fame team, and G_{HOF} is the system evaluation of the best hall of fame team. So, in the CCEA with the hall of fame and difference evaluation, the fitness of an agent is calculated as:

$$F(i) = \alpha \cdot D_i + (1 - \alpha) \cdot D_{HOF,i} \quad (10)$$

where D_i is the difference evaluation of agent i when collaborating with its team, $D_{HOF,i}$ is the agent's difference evaluation when paired with the best hall of fame team as in Equation 9, and $\alpha \in [0, 1]$ is a weight corresponding to the relative importance of the difference evaluation and the difference evaluation with estimated optimal collaborators. The CCEA with the hall of fame and difference evaluation is detailed in Algorithm 3.

By assuming that the best hall of fame team is the set of optimal collaborators for any agent, the complexities of estimating what a set of optimal collaborators would be are eliminated. The CCEA using the difference evaluation and the hall of fame includes shaped fitness functions to tell agents what their individual contributions to team performance are, as well as biasing the fitness functions using the concept of estimated optimal collaborators via the hall of fame. This approach modifies the hall of fame algorithm in two ways. First, the hall of fame is now comprised of *teams*, rather than *individuals*. Secondly, the hall of fame is now utilized in *cooperative* coevolution, rather than *competitive* coevolution.

5. EXPERIMENTAL RESULTS

The algorithms outlined in section 4 were all tested in the scatter domain and the rover domain, with team sizes varying from 10 to 100 agents. For experiments with 10 agent teams, 10 coevolving populations of 200 members each were used. For experiments with 100 agent teams, 100 coevolving populations of 25 members each were utilized. For the standard CCEA, Equation 3 with a default agent distance

Initialize N populations of k neural networks
foreach *Generation* **do**
  foreach *Population* **do**
    produce k successor solutions
    mutate successor solutions
  end
  for $i = 1 \rightarrow 2k \cdot m$ **do**
    randomly select one agent from each population
    add agents to team T_i
    simulate T_i in domain
    assign fitness to each agent with Eq. 10
  end
  foreach *Team* T_i **do**
    if $G(z|T_i) > G(z|HOF_{best})$ **then**
      add T_i to HOF
    end
  end
  foreach *Population* **do**
    select k networks using ϵ-greedy
  end
end

Algorithm 3: CCEA using Difference Evaluation and Hall of Fame (See Section 5 for parameters)

of 1.0 was used to assign fitness in the scatter domain and Equation 5 was used in the rover domain. For the CCEA algorithm with the difference evaluation, Equation 7 was used to assign fitness for the scatter domain and Equation 8 for the rover domain. For the CCEA with lenient learners and the difference evaluation, the same fitness equations were used as in the CCEA with the difference evaluation experiments. The leniency value m was initially set to 10, and decreased by 1 every 200 generations, resulting in no leniency after 2000 generations. The CCEA with the hall of fame and difference evaluation had the same fitness assignments as in the CCEA with the difference utility. When calculating the fitness from equation 9, α was set to 0.5. For all experiments, network mutation was carried out by adding values drawn from a Gaussian distribution to network weights. In the beginning of the evolution, one weight per network was mutated with a standard deviation of 1.0. At the end of the evolution, each network weight was mutated with a standard deviation of 0.1. These mutation parameters were varied linearly throughout the evolutionary process. For each experiment, 100 statistical runs were completed, with the standard error in the mean ($\sigma/\sqrt{N}$) being reported. The experiment details and results are given in the following sections.

5.1 Scatter Domain

First, we applied each of the four CCEA algorithms to the scatter domain. For the 10 agent experiment, the world was set to a 10 by 10 plane world and run for 10 time steps, and Δ_{max} was set to 1.0. For the 100 agent experiment, the world was set to a 31.6 by 31.6 plane world and run for 10 time steps, and Δ_{max} was set to 3.16. These values were chosen such that the plane area to number of agents ratio was constant for each experiment, and the agents could traverse the world in the same number of time steps. At the beginning of each experiment, the agents all started at the center of the plane worlds. Figure 3 shows the learning curve for the 10 agent problem. Figure 4 shows the learning curve for the 100 agent problem. Finally, Figure 5 shows the scaling properties of each algorithm in the scatter domain.

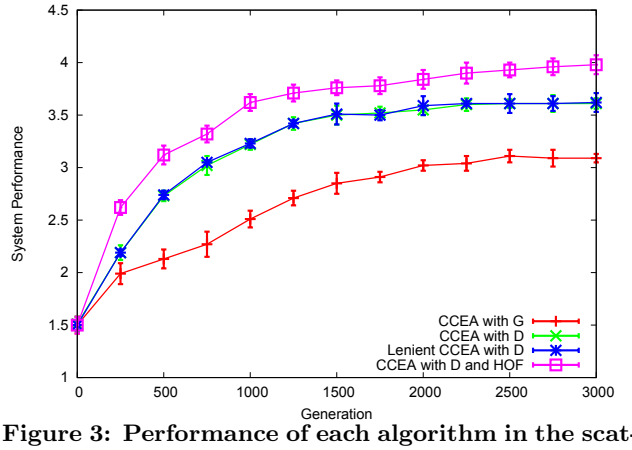

Figure 3: Performance of each algorithm in the scatter domain, with 10 agents. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

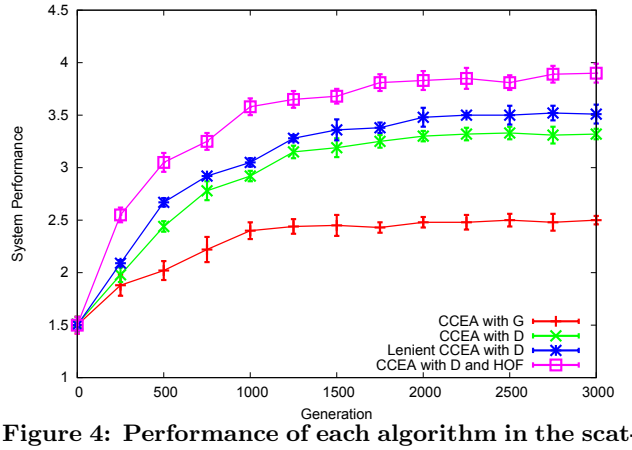

Figure 4: Performance of each algorithm in the scatter domain, with 100 agents. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

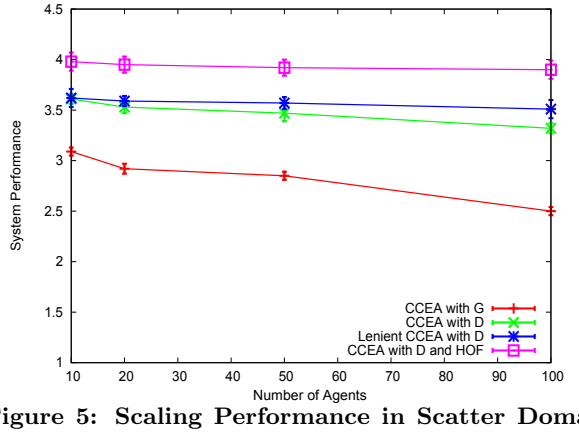

Figure 5: Scaling Performance in Scatter Domain. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested, and this difference in performance increases with team size. The addition of leniency does not significantly help when there are a small number of agents, but as the team size goes up, leniency becomes increasingly useful.

All three algorithms using the different evaluation function outperformed the standard CCEA in this domain. This is not surprising, because credit assignment in this algorithm is highly subjective, and the fitness of each agent was greatly influenced by its teammates. The CCEA with the difference evaluation and the CCEA with leniency and the difference utility performed almost identically in the 10 agent case, but the addition of leniency provided improved performance in the 100 agent case. This is an interesting result, and gives insight to the properties of leniency. Leniency and the difference utility both perform similar functions. Leniency aims to reduce the subjectiveness of credit assignments in CCEAs by partnering agents with multiple sets of collaborators. By testing an agent with multiple teams and taking the highest fitness achieved, the likelihood that an agent's fitness is too strongly biased by its teammates is minimized. The difference utility also reduces the subjectiveness of credit assignment, by isolating an agent's individual contribution to its team's performance. Thus, leniency and the difference evaluation achieve a similar goal, although in different manners. However, as the problem becomes more complex by increasing the team size, coupling both approaches provided benefits, as seen in Figure 5. As the team size is increased, the performance of the CCEA with the difference evaluation and lenient learners outperforms the CCEA with the difference reward by larger and larger margins.

The CCEA with the hall of fame and difference evaluation performed the best out of all of the algorithms tested. The goal of introducing the hall of fame was to approximate optimal collaborators, in order to bias the CCEA toward optimal solutions. The scatter domain results show that approximating optimal collaborators with the best known team is an acceptable approach to bias the CCEA. The hall of fame approach has the benefit that it is much simpler to implement than an ad hoc estimate of optimal collaborators, and is a more attractive alternative as the domain becomes more complex, because such an ad hoc estimate becomes increasingly difficult to develop as the domain complexifies. As seen in Figure 5, this approach becomes better compared to the other approaches as the number of agents increases.

5.2 Rover Domain

Finally, we applied each of the four CCEA algorithms to the rover domain. At the beginning of each experiment, n POIs were placed randomly in the domain, and their positions remained constant throughout each experiment, where n is equivalent to the number of agents in the domain. The minimum observation distance δ_{min} was set to 0.1. The simulations were carried out in a 10 by 10 world for 25 time steps, and Δ_{max} was set to 1.0. At the beginning of each simulation, each rover started in the center of the world. Each algorithm was tested over 50 statistical runs. Figure 6 shows the results for 10 agent teams, Figure 7 shows the results for 100 agent teams, and Figure 8 shows the scaling results for the rover domain.

As in the scatter domain experiments, all three algorithms using the different evaluation function outperformed the standard CCEA in the rover domain. This can be attributed to the fact that the system evaluation does not give an agent good feedback on its individual contribution to the team's performance, and the agent thus has a difficult time learning an optimal policy. In the scatter domain, the CCEA with the difference evaluation and the CCEA with lenient learners and the difference utility performed nearly identically with 10 agent teams, but leniency became more important as the team size went up. In the more complex rover domain, the CCEA with lenient learners and the difference evaluation performed slightly better than the CCEA with the difference reward, and this difference also increased with the team size. This indicates that although leniency and the difference utility have similar effects on the learning process, leniency may become more beneficial in CCEAs as the domain becomes more complex or the number of cooperating agents increases.

As in the scatter domain, the CCEA with the hall of fame and difference evaluation performed the best out of all algorithms tested. This further supports the conclusion that biasing the CCEA with hall of fame teams approximating optimal collaborators, as well as shaping the fitness functions, helps guide the search towards better solutions. As seen in Figure 8, the difference in performance between the CCEA with the difference evaluation and hall of fame and the CCEA with the global reward increases with team size, indicating that for increasingly complex systems, this new algorithm becomes more and more useful. In all the experiments that were conducted, the CCEA with the hall of fame and difference evaluation significantly outperformed

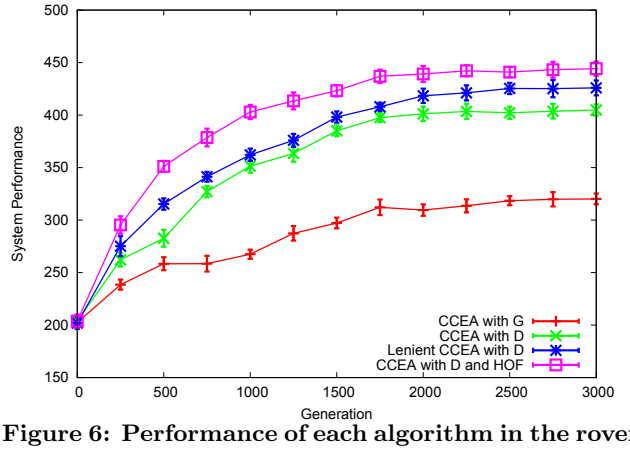

Figure 6: Performance of each algorithm in the rover domain, with 10 agents and 10 POIs. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

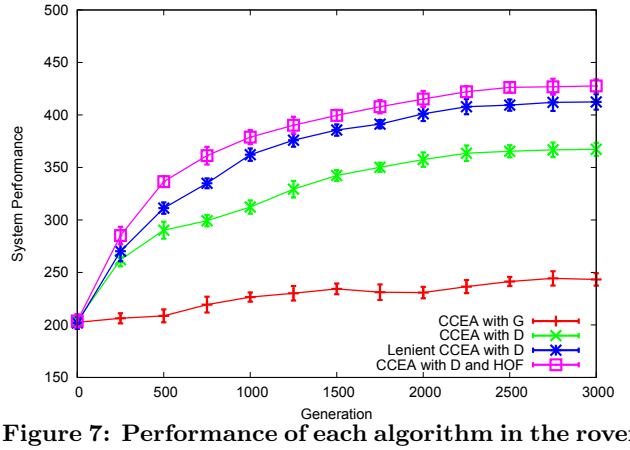

Figure 7: Performance of each algorithm in the rover domain, with 100 agents. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

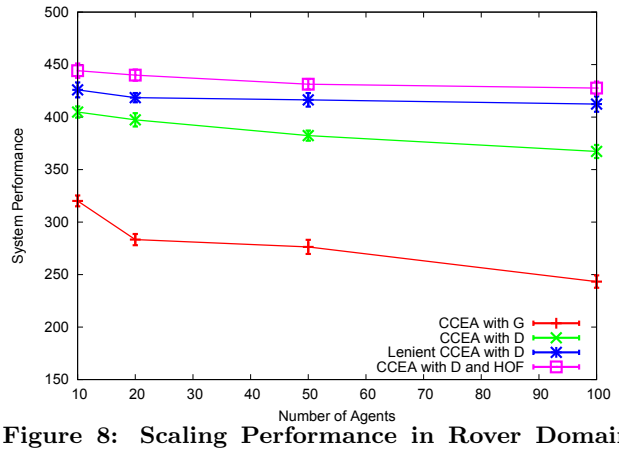

Figure 8: Scaling Performance in Rover Domain. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods, and the gap in performance increases with team size.

all other algorithms, and was able to easily estimate optimal collaborators in order to bias the fitness. This fitness biasing, in addition to shaping the fitness values with the difference evaluation, contributed to significantly better performance than any other algorithm tested.

6. DISCUSSION

This paper presented three CCEA algorithms where leniency and hall of fame methods were used in combination with fitness shaping. Combining hall of fame and difference evaluation outperformed all other algorithms in two different domains. It is known that shaping fitness functions can greatly improve the efficacy of a CCEA [8], but this often isn't enough to obtain optimal performance. Biasing a CCEA with an estimate optimal collaborators results in a more effective search for optimal policies, but this estimate is problematic because it is an ad hoc, domain dependent estimate [10]. Our algorithm circumvents the problem of estimating optimal collaborators, so the CCEA search is easily biased in a domain independent fashion. Furthermore, the algorithm shapes agent fitnesses in order to provide each agent a measure of its individual contribution to the system objective.

A particularly interesting result was that in the simpler scatter domain, using the difference evaluation to shape the fitness functions performed equivalently to a method which used lenient learners and the difference utility for 10 agent teams. Intuitively, the difference utility and lenient learners both aim to achieve the same goal, which is to isolate an agent's individual contribution to the system evaluation. This is one key reason their performance was similar in the simpler domains. However, with larger teams or a more complicated domain, leniency in addition to the difference evaluation performed better than the difference evaluation alone. Our current research focuses in two directions: (i) investigating the theoretical relationship between leniency and difference evaluation functions, including determining when adding leniency to fitness shaping is desirable; and (ii) the impact of biasing the search while using difference evaluation functions, including alternative biasing methods.

Acknowledgements

This work was partially supported by NSF Grant IIS-0910358, and DoE NETL grant DE-FE0000857.

7. REFERENCES

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

[2] A. K. Agogino and K. Tumer. A multiagent approach to managing air traffic flow. *Journal of Autonomous Agents and Multi-Agent Systems*, 24:1–25, 2012. DOI: 10.1007/s10458-010-9142-5.

[3] A. Bucci and J. B. Pollack. Thoughts on solution concepts. In *"Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2007.* ACM, 2007.

[4] S. G. Ficici. Monotonic solution concepts in coevolution, 2005.

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

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

[7] T. D. Haynes, D. A. Schoenefeld, and R. L. Wainwright. Type inheritance in strongly typed genetic programming. In *Advances in Genetic Programming 2, chapter 18*, pages 359–376. MIT Press, 1996.

[8] P. J. Hoen and E. D. D. Jong. Evolutionary multi-agent systems. In *In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature PPSN-04*, pages 872–881, 2004.

[9] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010.

[10] L. Panait, S. Luke, and R. P. Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, 2006.

[11] L. Panait, K. Tuyls, and S. Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *J. Mach. Learn. Res.*, 9:423–457, June 2008.

[12] M. A. Potter and K. A. De Jong. Evolving Neural Networks with Collaborative Species. *Computer Simulation Conference*, 1995.

[13] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5:1–29, 1996.

[14] T. Soule and R. B. Heckendorn. A developmental approach to evolving scalable hierarchies for multi-agent swarms. In *"Proceedings of the 12th Annual Conference on Genetic and Evolutionary Compuation – GECCO-2010.* ACM, 2010.

[15] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *AUTONOMOUS ROBOTS*, 8:345–383, 1997.

[16] R. P. Wiegand, K. A. D. Jong, and W. C. Liles. Modeling variation in cooperative coevolution using evolutionary game theory, 2002.