The Analysis and Design of Concurrent Learning Algorithms for Cooperative Multiagent Systems

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Liviu Panait

Bachelor of Science
University of Bucharest, Romania, 1999
Master of Science
George Mason University, USA, 2002

Director: Dr. Sean Luke, Assistant Professor
Department of Computer Science
The Volgenau School of Information Technology and Engineering
George Mason University

Fall 2006
George Mason University
Fairfax, VA

Dedication

This thesis is dedicated to my family and friends. The hope that they are proud of what I accomplish has been a major driving force for all my endeavors.

Acknowledgments

I thank my committee members for their constant help and guidance during my graduate studies. First, Sean Luke, my dissertation director, my advisor, my boss, and my friend, has exceeded my greatest expectations for a mentor. During the past six years, he has encouraged me to pursue challenging research questions in a variety of areas, while shielding me from bureaucratic and financial concerns at the same time. Sean has taught me the skills required to manage multiple research projects at the same time, and has consistently worked on polishing my presentation skills. Ken De Jong has always been a great model for how to conduct and present my research. Claudio Cioffi-Revilla and Jana Košecká have pointed out many rich sources of inspiration for my work and have made sure it has a broad applicability.

This thesis benefited significantly from the help of many others. Dana Richards and Paul Wiegand provided careful checks of the formal analysis in Chapter 3. Karl Tuyls helped me decipher his EGT model for multiagent Q-learning, which I use and extend in Chapter 3 as well. I thank Keith Sullivan, Martha McJunkin, and Joey Harrison, for carefully checking the thesis to eliminate all grammatical errors. I also thank Rafal Kicinger, Bill Liles, Mark Coletti, Elizabeth White, and Harry Wechsler for their comments, suggestions, and advices throughout the years.

Next, I thank my friends for helping me relax while away from my work. Their jokes, stories, advices, comments, or simple presence, have often put a smile on my face. Guido Cervone has been a great friend ever since I started working at George Mason University, and he has always had a good advice when I needed one. Marcel Bărbulescu was one friend I could always count on at any time of day or night. My current and former roommates, Jacek Radzikowski, Sepideh Mirza, Valentin and Vica Prudius, and Florin Ciucu, have always known how to lighten up an evening with a dinner, a drink, a joke, a conversation, or even with a few nice words. Many thanks also go to Adrian, Paula, and Ana Grăjdeanu, Mac and Al Nakari and their entire family, Aladdin El-Nattar, Sumer Jabri, Cindy Halim, Jeff and Hideko Bassett, Zbigniew Skolicki (aka Z), Elena Popovici, Neely Belai and Yared Aklilu, Luigina and Alex Pinzino and Nonno, Linda Dodd-Major and HB and Evan Major.

I also thank the many friends who helped me use the excess of energy in a peaceful manner. It has been a great pleasure to play volleyball with Dasha Fedorova, Kirill Sukhorukov, and the rest of the Russian gang, particularly in the intramural tournaments at George Mason University. I thank Joey Harrison and Gabriel Bălan for the unforgettable shock on the faces of skilled basketball opponents when our trio crushed any hope they had to win. I also thank Larry Clark, Iman

Nikmaram, and the Georgetown gang of students for the many opportunities to show off my soccer skills.

Finally, I am forever grateful to my mother Maria (aka Bonnie), my father Costantin (aka Costică), my brother Vlad (aka Panseluţă), and to my grandmother Ioana (aka Vrăbiuţă), for their constant love and support. I also acknowledge the endless tolerance, candid love, and continued encouragements of Camelia Lică, who has been by my side during the ups and downs of my life through graduate studies. She has always enforced a necessary balance between my personal and professional activities, making sure that sustained research efforts are always accompanied by endless shopping trips to nearby malls.

Table of Contents

List of Tables

List of Figures

List of Symbols

CCEA	Cooperative Coevolutionary Algorithm
EA	Evolutionary Algorithm
EGT	Evolutionary Game Theory
iCCEA	Informative Cooperative Coevolutionary Algorithm
LMQ	Lenient Multiagent Q-learning
RD	Replicator Dynamics

Abstract

THE ANALYSIS AND DESIGN OF CONCURRENT LEARNING ALGORITHMS FOR COOPERATIVE MULTIAGENT SYSTEMS

Liviu Panait, PhD

George Mason University, 2006

Dissertation Director: Dr. Sean Luke

Concurrent learning is the application of several machine learning algorithms in parallel to automatically discover behaviors for teams of agents. As machine learning techniques tend to find better and better solutions if they are allowed additional time and resources, the same would be expected from concurrent learning algorithms. Surprisingly, previous empirical and theoretical analysis has shown this not to be the case. Instead, concurrent learning algorithms often drift towards suboptimal solutions due to the learners' coadaptation to one another. This phenomenon is a significant obstacle to using these techniques to discover optimal team behaviors.

This thesis presents theoretical and empirical research on what causes the aforementioned drift, as well as on how to minimize it altogether. I present evidence that the drift often occurs because learners have poor estimates for the quality of their possible behaviors. Interestingly, improving a learner's quality estimate does not require more sophisticated sensing capabilities; rather, it can be simply achieved if the learner ignores certain reward information that it received for performing actions. I provide formal proofs that concurrent learning algorithms will converge to the globally optimal solution, provided that each learner has sufficiently accurate estimates.

This theoretical analysis provides the foundation for the design of novel concurrent learning algorithms that benefit from accurate quality estimates. First, the estimates of learners employing the biased cooperative coevolutionary algorithm are greatly improved based on reward information that was received in the past. Second, the informative cooperative coevolutionary algorithm provides learners with simpler, functionally-equivalent estimates at a reduced computational cost. Finally, I describe the lenient multiagent Q-learning algorithm, which benefits from more accurate estimates when tackling challenging coordination tasks in stochastic domains.

Chapter 1: Introduction

Many complex real-world problems have certain characteristics in common that make them suitable for decentralized solutions. As a result, recent years have witnessed increased interest in the area of multiagent systems, which emphasizes the joint behaviors of agents with some degree of autonomy. In this thesis, the range of problem domains is additionally restricted to ones in which multiple agents cooperate to accomplish common goals. Disaster search and rescue is a good example of such a problem domain; here, multiple agents (robots) work together to locate survivors of a disaster and to help with rescue operations.

In many cooperative multiagent domains, it is relatively straightforward to describe (at a macro level) what one would like the team of agents to accomplish. For example, agents in a disaster search and rescue application should perform a thorough sweep of the area, locate survivors as quickly as possible, provide assistance with rescue operations, minimize damage to the robots, etc. On the other hand, it is often extremely difficult to specify (at the micro level) what each agent should do such that the entire team behaves as desired. The causes for this discrepancy are twofold. First, there may be little connection between a desired macro-level phenomenon and the many simple agent actions and interactions that lead to it. It is often the case that unexpected and sophisticated team behaviors emerge from basic agent behaviors. Second, the complexity of the multiagent solution can rise exponentially with the number of components and their behavioral sophistication.

For these reasons, machine learning approaches are a viable alternative to hard-coding behaviors for multiagent teams. Machine learning explores ways to automate the inductive process: getting a machine agent to discover on its own how to solve a given task or to minimize error. Automation is attractive. Additionally, agents capable of learning might also be able to adapt their behavior in the case of unexpected changes in the environment (for example, if one of their teammates fails).

Unfortunately, recent theoretical and empirical research has shown that naïve extensions of single-agent learning algorithms to multiagent scenarios often drift away from optimal solutions. This is intriguing: most single-agent learning algorithms are guaranteed to converge to the optimal solution if properly set and given enough resources, but these guarantees mysteriously vanish in multiagent domains. What causes this drift away from the optima? The answer to this question is the primary focus of this thesis. Additionally, I demonstrate that the isolation and elimination of these causes facilitates the design of novel multiagent learning algorithms with superior performance.

Before delving further into these issues, I start with brief introductions to the areas of multiagent systems and multiagent learning, followed by an argument on the significance of research in multiagent learning. This chapter concludes with a summary of the research contributions of this thesis, and an overview of the remaining chapters.

1.1 Multiagent Systems

The terms agent and multiagent are not well-defined in the research community; this thesis uses an admittedly broad definition of these concepts. An agent is a computational mechanism that exhibits a high degree of autonomy, performing actions in its environment based on information (sensors, feedback) gathered from the environment. Multiagent systems involve several such agents interacting with one another, and have proven advantageous in real-world domains that feature agents with incomplete information about the environment, lack of centralized control, decentralized and distributed information, and asynchronous computation (Jennings *et al.*, 1998).

The literature contains multiple taxonomies for multiagent systems. For example, multi-robot applications can be characterized based on team size, range, communication topology and throughput, team composition and reconfigurability, and the processing ability of individual agents (Dudek *et al.*, 1993). Agent characteristics (team heterogeneity, control architectures, input/output abilities) and system characteristics (for example, communication settings) are considered the primary criteria to differentiate among real-world applications of multiagent systems to industry (Parunak, 1996). In (Stone and Veloso, 2000), the authors explicitly distinguish four groups of approaches, divided by heterogeneity versus homogeneity of the team, and by communication versus lack thereof. Finally, multiagent systems can be characterized based on types of environments, agents, and inter-agent interactions (Weiß, 1997; Weiß, 1999; Huhns and Singh, 1998).

1.2 Multiagent Learning

I define multiagent learning as the application of machine learning to problems involving multiple agents. There are two features of multiagent learning that provide merit for its study as a separate field. First, because multiagent learning deals with problem domains involving multiple agents, the search space can be unusually large; and due to the interaction of those agents, small changes in learned behaviors can often result in unpredictable changes in the resulting macro-level ("emergent") properties of the multiagent group as a whole. Second, multiagent learning may involve multiple learners, each learning and adapting in the context of others; this co-adaptation of

the learners to one another represents a significant violation of a fundamental assumption of most machine learning techniques.

While one might restrictively define cooperative domains in terms of joint shared reward for all agents in the team, much of the literature tends to use a much broader notion of cooperation, particularly with the introduction of credit assignment (discussed in Section 2.2.1). I define cooperative multiagent learning in terms of the intent of the experimenter. If the design of the problem and the learning system is constructed so as to (hopefully) encourage cooperation among the agents, then this is sufficient, even if in the end they fail to do so. As an observation, learning might obviously target competitive teams of cooperating agents, such as in the game of soccer — while a researcher might apply machine learning techniques to search for behaviors for all 22 agents (11 per each team), he or she would usually be interested in only the behaviors for the agents in one of the teams.

In this thesis, I ignore a large class of learning approaches for multiagent systems, namely ones where a single agent learns while the other agents' behaviors are fixed. An example of such a scenario is presented in (Grefenstette, 1991). This is single-agent learning: there is only one learner, and the behaviors are plugged into only one agent, rather than distributed to multiple agents.

There are three main approaches to learning: supervised, unsupervised, and reinforcement learning. These methods are distinguished by the kind of feedback the critic provides to the learner. In supervised learning, the critic provides the correct answer. In unsupervised learning, no feedback is provided at all. In reinforcement learning, the critic provides rewards or penalties for the answers a learner selects.

Because of the inherent complexity in the interactions of multiple agents, various machine learning methods — notably supervised learning methods — are not easily applied to the problem because they typically assume a critic which can provide the agents with the "correct" behavior for a given situation[1]. Given this difficulty, the vast majority of papers in this field have used reinforcement learning methods. The reinforcement learning literature may be approximately divided into two subsets: methods which estimate value functions, such as Q-learning, TD(λ), SARSA, etc; and stochastic search methods which directly learn behaviors without appealing to value functions, such as evolutionary algorithms, simulated annealing, and stochastic hill-climbing. In the stochastic search literature, most multiagent discussion concentrates on evolutionary algorithms. The similarities between these two classes of learning mechanisms permit a rich infusion of ideas from one to the other: the bucket-brigade algorithm (Holland, 1985), the

[1] A notable exception presented in (Goldman and Rosenschein, 1996) involves teaching in the context of mutual supervised learners.

SAMUEL system (Grefenstette *et al.*, 1990), and the recent Stochastic Direct Reinforcement policy gradient algorithm (Moody *et al.*, 2004).

One simple multiagent learning approach is to employ a single learning process for the entire team of agents. Sections 1.2.1 – 1.2.2 detail two such techniques. Another approach is to distribute the learning process by endowing all team members with learning capabilities, such that each agent tries to improve the performance of the entire team by improving its own behavior. Sections 1.2.3 – 1.2.4 summarize two straightforward extensions of traditional machine learning techniques for such distributed learning tasks.

1.2.1 Evolutionary Algorithms

Evolutionary computation is the research field that studies the properties and applications of evolutionary algorithms (EAs), which are techniques that apply abstract Darwinian models of evolution to iteratively refine candidate solutions to a given problem. Evolutionary algorithms are stochastic beam search methods: they employ randomness in conjunction with sets of candidate solutions to escape locally optimal solutions. The EA terminology is borrowed from biology to reflect the main source of inspiration for the field; for example, candidate solutions are termed individuals, sets of individuals are known as populations, and the creation of new solutions from previous ones is usually referred to as breeding.

An evolutionary algorithm begins with an initial population of random individuals. Each member of this population is then evaluated and assigned a fitness (a quality assessment). The EA then selects a set of fitter-than-average parents and alters them via crossover and mutation operations to produce a set of children, which are added to the population, replacing older, less fit individuals. One evaluation, selection, and breeding cycle is known as a generation. Successive generations continue to refine the population until time is exhausted, or until a sufficiently fit individual is discovered. There are many sources of additional information on evolutionary algorithms; good choices include (Bäck, 1996; De Jong, 2006; Fogel, 1999; Goldberg, 1989; Holland, 1975; Koza, 1992; Michalewicz, 1996).

1.2.2 Q-learning

Q-learning (Watkins, 1989) is particularly useful in domains where reinforcement information (expressed as penalties or rewards) is observed after a sequence of actions has been performed. Q-learning associates a utility Q with each (s, a) pair, where s is a state of the environment, and a is an action that the agent can perform when in state s. The agent updates the Q values at each time step based on the reinforcement it has received. The update formula is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \tag{1.1}$$

where the agent has just been in state s_t, has performed action a_t, has observed reward r_t, and it has transitioned to the new (current) state s_{t+1}; α is the learning rate, and γ is a discount factor for incorporating future rewards into the utility estimate. Action selection is usually based on a stochastic process; popular choices include the ε-greedy exploration (select the best action with probability $1-\varepsilon$, or a random action otherwise) and the Boltzmann exploration (select action a in state s proportionately to $e^{\frac{Q(s,a)}{\tau}}$, where τ is a "temperature" parameter that controls the greediness of the selection process). Q-learning has certain guarantees of convergence under theoretical conditions, which include performing each action an infinite number of times in each state, as well as proper settings for the learning rate (Watkins and Dayan, 1992; Singh *et al.*, 2000). For more information on Q-learning and other reinforcement learning techniques, good starting points include (Sutton and Barto, 1998; Barto *et al.*, 1990; Kaelbling *et al.*, 1996; Sutton, 1988).

1.2.3 Cooperative Coevolutionary Algorithms

Coevolution is an intuitive and easy-to-study extension of evolutionary algorithms for domains with multiple learning agents. Cooperative coevolutionary algorithms (CCEAs) are a popular approach for domains where agents succeed or fail together (Husbands and Mill, 1991; Potter and De Jong, 1994). Here, each agent is given its own population of individuals, each such individual representing a possible behavior for that agent. CCEAs usually assume that only the performance of the entire team can be assessed; as a consequence, one individual (behavior) for each of the team members is required for evaluation. Fitness assessment works as follows: for each individual *i* in an agent's population, one or more tuples are formed using *i* and individuals (termed collaborators) from the other agents' populations (either from the current generation, or from the previous one). These collaborators can be sampled randomly or selected based on performance (if the collaborators were evaluated in the previous generation, the better ones might be used preferentially). The fitness of *i* is then computed based on the performance obtained from evaluating the teams with the behaviors indicated by these tuples. As a consequence, the fitness assessment is context-sensitive and subjective. Aside from evaluation, the learning processes are independent of one another. In this thesis, I assume that the populations evolve concurrently, which is closer to the asynchronous nature of multiagent systems. For additional details on CCEAs, the readers are referred to (Potter, 1997; Wiegand, 2004).

1.2.4 Multiagent Q-learning

Multiagent Q-learning represents a straightforward extension of Q-learning to domains involving multiple agents (Claus and Boutilier, 1998). Here, each agent is given its own Q-value function for (state,action) pairs. All agents choose their actions independently and concurrently, perform them in parallel, and all observe the same reward associated with the joint action. In contrast to single-agent Q-learning, but similar to CCEAs, the information an agent observes is subjective — it depends on the actions chosen by other agents. The reward information is then used by each agent to update its Q-values. Usually, agents cannot perceive which actions their teammates have chosen.

1.3 Significance of Research

Multiagent systems are gradually becoming the main solution to a variety of real-world problems where centralized solutions have proven infeasible or suboptimal. For example, reconnaissance missions might be approached at comparable costs with either a large, complex and expensive plane, or a swarm of cheap unmanned air vehicles: the swarm could succeed, even if enemies destroy half the agents in the swarm, while the mission might be jeopardized if a single missile hits the wing of the large plane. Other practical applications of multiagent systems include cooperative mapping and target observation (Bassett, 2002; Parker, 2000; Luke *et al.*, 2005), vehicle traffic management (Lesser *et al.*, 1987; Nunes and Oliveira, 2004; Dresner and Stone, 2004; Balan and Luke, 2006), air traffic control (Steeb *et al.*, 1988), network management and routing (Weihmayer and Velthuijsen, 1994; Boyan and Littman, 1994; Subramanian *et al.*, 1997; Wolpert *et al.*, 1998), electricity distribution management (Varga *et al.*, 1994; Schneider *et al.*, 1999), and meeting scheduling (Chalupsky *et al.*, 2002; Crawford and Veloso, 2004), to name but a few.

Many fields have finally adopted multiagent models and frameworks due to their inexpensive computational brawn. These include models of social, political, and economic behavior; population-, systems-, and cellular-biology models and genomics; and video-game and virtual-reality agents, and simulations for animation. Multiagent frameworks have come into their own with the onset of the Internet, cluster computing, and peer-to-peer architectures.

Along with multiagent systems in general, multiagent learning will be called upon to assist in the development of multiagent behaviors, particularly as the complexity of the environments and the emergent behaviors become such that hand-coded solutions are not easily obtainable. Take for example the emerging field of computational social sciences, which uses computer based multiagent models to gain insight into past, present, and future human societies. Research in this field relies on hardcoding agent behaviors to replicate inter-human social interactions, which is a

very challenging task. For example, how should agents think and act, and what existing conditions are necessary, for a state or an empire to emerge? Multiagent learning techniques could provide assistance with this process by automating much of the search for agent behaviors. This approach will hopefully speed progress in computational social sciences, as well as in many other research fields that have adopted multiagent systems.

Progress in multiagent learning can also provide valuable inspiration for other research areas where decompositions of the learning tasks are possible. This is because certain multiagent learning techniques, such as the ones discussed in this thesis, essentially address the problem of decomposing complex search spaces into simpler components that are tractable to existing machine learning techniques.

Last, and certainly not least, research in multiagent learning will provide further insight into the human learning process. I argue that only a small part of human learning is in isolation, usually because there are limits to what one can solve alone. Most often, learning is performed through interactions with teachers, colleagues, or coworkers, and the range of problems that can be solved together is significantly broader. It has been noted before that there are certain features that distinguish multiagent learning methods from the traditional learning techniques in single-agent domains. As such, research in multiagent learning will provide unique and valuable information about specific characteristics of our own learning mechanisms.

1.4 Summary of Research Contributions

This thesis makes four significant contributions to the multiagent learning research. They are summarized below.

- **It establishes theoretical guarantees of convergence to the global optimum for certain multiagent learning algorithms.** I claim that multiagent learning might drift to suboptimal solutions simply because each learner has a poor estimate for the quality of its possible behaviors, and I extend certain theoretical models to account for increasingly accurate estimates. Using these models, I prove that multiagent learning algorithms are guaranteed to converge to the global optimum, if properly set and if given enough resources. This result extends prior theoretical research, showing that claims on the existence of uncontrollable pathologies that plague multiagent learning can be simply attributed to improper settings. Additionally, the theoretical analysis indicates that the key to improving multiagent learning algorithms is to make sure each learning process has an accurate estimate for the quality of its behaviors.

- **It demonstrates the advantages of biasing cooperative coevolutionary algorithms.** I propose and analyze several approaches to combine past and current information in order

to improve a learner's estimation for the quality of its behaviors. I show that some biasing schemes might be very sensitive to different settings, while others are not. I then show that several multiagent learning algorithms can be enhanced with a biasing mechanism to achieve significantly better performance.

- **It details a novel multiagent learning algorithm that benefits form simpler, functionally-equivalent estimates that can be obtained with reduced computational costs.** In addition, learners take an active approach to improve each other's estimates for the quality of their behaviors, and as a consequence, they are able to shift more computational resources to searching for better behaviors. I provide empirical results that demonstrate the advantages of this approach in domains where each agent has an infinite set of actions to choose from.

- **It introduces a novel multiagent learning algorithm for stochastic domains.** I address the more difficult task of achieving good quality estimates in domains where the reward information is noisy. In particular, I show that learners can benefit from showing lenience to one another, especially at early stages of learning. I propose a lenient multiagent learning algorithm that significantly outperforms other state-of-the-art algorithms, particularly as the level of noise increases.

The thesis also includes two other important contributions:

- **It provides a broad survey of the cooperative multiagent learning literature.** Previous surveys of this area have largely focused on issues common to specific subareas (for example, reinforcement learning or robotics). These communities have usually worked on similar issues, with little or no cross-fertilization of ideas. The thesis provides an extensive survey of multiagent learning work in a spectrum of areas, including reinforcement learning, evolutionary computation, game theory, complex systems, agent modeling, and robotics. I identify two broad categories of multiagent learning approaches, each with its own special issues: applying a single learner to discover joint solutions to multiagent problems, or using multiple concurrent learners. I also expose critical open issues that the research community needs to address before multiagent learning methods achieve widespread applicability to real-world problems.

- **It provides graphical illustrations of why concurrent learners might perform suboptimally.** I present intuitive graphical illustrations to support the claim that certain algorithmic decisions might lead to learners having poor quality estimates for their behaviors. This argument is supported further by the visualization of the basins of attraction to optimal and suboptimal solutions for learners with increasingly accurate estimates.

1.5 Dissertation Layout

The remainder of the dissertation is structured as follows. Chapter 2 provides a comprehensive survey of the state of the art in cooperative multiagent learning. I identify two primary classes of multiagent learning approaches: team learning and concurrent learning. The former applies a single learner to discover behaviors for all agents, while the latter employs multiple learners in parallel.

Chapter 3 presents a theoretical analysis of why certain concurrent learning algorithms may fail to converge to the optimal solution, even when given unlimited resources. I claim that such problems might arise from individual learners having poor estimates for the quality of their behaviors. I then employ a formal mathematical model to analyze the impact of increasingly accurate estimates onto the overall performance of the system.

The following three chapters describe novel multiagent learning algorithms that benefit from accurate estimates. Chapter 4 presents a biasing approach to improve quality estimates for cooperative coevolutionary algorithms by combining past and present information. Chapter 5 details another novel multiagent learning algorithm that benefits from the identification and exploration of informative actions — actions that, when chosen by a learner, provide the others with valuable information on how to accurately estimate the quality of various areas of their search spaces. In Chapter 6, I present a natural extension of these ideas to environments where noisy information provides an additional challenge to learners' having accurate estimates of which actions are better than others.

Finally, Chapter 7 summarizes the thesis and suggests directions for future work.

Chapter 2: The State of the Art in Cooperative Multi-Agent Learning

Researchers have traditionally approached multiagent learning from a number of different directions: temporal difference learning, evolutionary computation and stochastic optimization, cellular automata and other "swarm complexity" models, robotics, and game theory. This has led to non-interacting communities, and advances in one area has usually had little impact on learning methods in other areas. In this chapter, I provide a comprehensive overview of cooperative multiagent learning, and I identify major open research issues for the field. An extended version of this survey is available in (Panait and Luke, 2005a).

I argue that there are two major categories of cooperative multiagent learning approaches. The first one, *team learning*, applies a single learner to search for behaviors for the entire team of agents. Techniques in this category are more along the lines of traditional machine learning, but they may have scalability problems as the number of agents increases. To keep the search space manageable, team learning techniques might assign identical behaviors to multiple agents in the team.

A second category of techniques uses multiple concurrent learning processes. I term this set of approaches *concurent learning*. Concurrent learning approaches typically employ a learner for each team member, thus reducing the joint space by projecting it into N separate spaces. However, the presence of multiple concurrent learners makes the environment non-stationary, which poses problems to many machine learning techniques.

To illustrate the difference between the two, consider the application of evolutionary computation to the disaster search and rescue scenario. A team learning approach using evolutionary computation encodes the behavior of the entire team in each genetic individual, and the fitness evaluates how good the team behavior is according to the performance measure provided as feedback. A concurent learning approach using evolutionary computation (i.e. a cooperative coevolutionary approach) employs N populations, each searching for a single agent's behavior. However, the performance measure must now be apportioned among the various agents (for example, dividing the team reward equally among the team members, or on a per-merit basis — exactly how much each agent has contributed to the search and rescue operation). The agents will improve their behaviors independent of one another, but have little or no control over how the other agents decide to behave.

When a single learner is employed, most research has focused on the representation of candidate solutions that the learner is developing: in particular, the degree of heterogeneity among

the team members. Section 2.1 covers this research. Following, Section 2.2 focuses on concurrent learning. Research in this area concerns with how to divide team reward to each learner, the analysis of the dynamics of concurrent learning processes, and the benefits of modeling other agents in order to improve cooperation.

Communication adds an entirely new level of sophistication and difficulty to multiagent learning research. As such, I dedicate Section 2.3 to only discuss the connection between inter-agent communication and learning. The chapter ends with a discussion of three areas in multiagent learning, namely adaptive dynamics, problem decomposition, and scalability, which present significant open research questions that have so far been insufficiently addressed by the community. This is covered in Section 2.4.

2.1 Team Learning

Team learning techniques apply a single learner to discover behaviors for all the agents in the team. Such an approach makes most machine learning techniques readily available for their application to multiagent systems. Team learning is additionally interesting in that because the agents interact with one another, the joint behavior arising from these interactions can be unexpected: this notion is often referred to as the emergent complexity of multiagent systems. There is another advantage of team learning, although it might become more evident after discussing the difficulties of concurrent learning techniques: as it involves a single agent, team learning approaches sidestep the difficulties arising from the co-adaptation of several learners (a major issue for concurrent learning).

Team learning has some disadvantages as well. A major problem is the large state space for the learning process. For example, if agent A can be in any of 100 states and agent B can be in any of another 100 states, the team formed by the two agents can be in as many as $10,000$ states. This explosion in the state space size can be overwhelming for learning methods that explore the space of state utilities (such as Q-learning), but it may not as drastically affect techniques that explore the space of behaviors (such as evolutionary computation) (Iba, 1999; Salustowicz *et al.*, 1997; Salustowicz *et al.*, 1998; Sen and Sekaran, 1996). A second disadvantage is the centralization of the learning algorithm: all information and resources need to be available in a single place, where the learning is performed. This can be burdensome in domains where data is inherently distributed, especially when privacy constraints on sharing the data are present.

Team learning may be split into two broad categories: *homogeneous* and *purely-heterogeneous* team learning. Homogeneous learners develop a single agent behavior which is used by every team member: this reduces the search space (a unique behavior is represented, as opposed to one for each agent), but it might prevent agents from specializing to particular tasks. Purely-heterogeneous team learners develop a unique behavior for each agent: they must cope with a

larger search space, but hold the promise for better solutions through agent specialization. There exist middle-ground approaches between these two categories. For example, dividing the team into groups, with group mates sharing the same behavior. Such approaches, which I term *hybrid* team learning methods, reduce the number of behaviors that need to be learned (as opposed to purely-heterogeneous teams), while still allowing for agent specialization (in contrast to homogeneous teams).

To illustrate the differences among these approaches, consider possible applications of team learning to the disaster search and rescue domain. The single learning process is concerned with improving the performance of the entire team of agents. As such, it needs to somehow encode the agents' behaviors — how each agent will behave in any situation it might encounter. Given the fact that the learning algorithm needs to search for these agent behaviors, large teams of agents result in enormous, usually intractable, search spaces. As such, a purely-heterogeneous team learning approach might only be used with very small numbers of agents. Alternatively, a homogeneous team learning technique assumes that all agents have the same behavior, thus the learner needs to only encode this single behavior. This results in a relatively small search space. However, this reduction might eliminate good solutions from the space that the learning algorithm is actually searching. Indeed, it may be helpful for some agents to act as scouts and search for survivors, while the other agents specialize at only assisting with the rescue operations. As a homogeneous team learning approach cannot represent such solutions, it will not discover them either. As mentioned earlier, this team decomposition could be automatically taken advantage of by a hybrid team learning approach with agents grouped as either scouts or rescuers.

Choosing among these approaches depends on whether specialists are needed in the team or not. Experiments reported in (Balch, 1998; Bongard, 2000; Potter *et al.*, 2001) address exactly this issue, and I discuss their findings here, even though some of them fall in the area of concurrent learning. Balch suggests that domains where single agents can perform well (for example, cooperative foraging) are particularly suited for homogeneous learning, while domains that require task specialization (such as robotic soccer) are more suitable for heterogeneous approaches (Balch, 1998) . His argument is bolstered by the report that heterogeneity may be better in inherently decomposable domains (Bongard, 2000). Experiments with increasingly difficult versions of a multiagent herding domain obtained by adding predators, as reported in (Potter *et al.*, 2001), indicate that domain difficulty might not be a determinant factor requiring a heterogeneous approach. Instead, the results show that heterogeneous behaviors might be preferable when solving the domain requires more than one skill.

2.1.1 Homogeneous Team Learning

In homogeneous team learning, all agents are assigned identical behaviors, even though the agents may not be identical (for example, different agents might take a different amount of time to complete the same task). Because all agents have the same behavior, the search space for the learning process is drastically reduced. The appropriateness of homogeneous learning depends on the problem: some problems do not require agent specialization to achieve good performance. For other problem domains, particularly ones with very large numbers of agents ("swarms"), the search space is simply too large to use heterogeneous learning, even if heterogeneity would ultimately yield the best results.

In straightforward examples of successful homogeneous team learning, the evolution of behaviors for a predator-prey pursuit domain is reported in (Haynes *et al.*, 1996; Haynes and Sen, 1995; Haynes *et al.*, 1995a; Haynes *et al.*, 1995b; Haynes *et al.*, 1995c). When using fixed algorithms for the prey behavior, the results indicate that the evolved behaviors have similar performance as the best human-coded greedy predator algorithms. However, learning can also discover the behavior for a prey that evades all previously reported hand-coded, greedy, and evolved predators.

Agents can act heterogeneously even in homogeneous team learning, if the homogeneous behavior specifies sub-behaviors that differ based on an agent's initial condition (location, etc.) or its relationship with other agents ("always follow agent 2"). For example, the use of evolutionary computation techniques for a team formation problem is examined in (Quinn *et al.*, 2002). Here, three agents start from random positions but close enough to sense one another. They are required to move the team centroid a specific distance while avoiding collisions and remaining within sensor range. The authors investigate the roles of team members by removing the agents one at a time. They conclude that the rear agent is essential to sustain locomotion, but it is not essential to the other two agents' ability to maintain formation. The middle agent is needed to keep the two others within sensor range, and the front agent is crucial for team formation. Therefore, even though the agents are homogeneous, they specialize (based on their relative positions) to perform better as a team.

Other research compares different machine learning methods as applied to homogeneous learning; for example, the PIPE and CO-PIPE algorithms (variations of evolutionary algorithms) are compared to Q-learning (with all agents using the same Q-table) in (Salustowicz *et al.*, 1997; Salustowicz *et al.*, 1998). The results indicate that Q-learning has serious learning problems, attributed by the authors to the algorithm's need to search for a value function. On the other hand, both PIPE and CO-PIPE search directly in the policy space and show good performance. A contradicting result is reported in (Wiering *et al.*, 1999), where a modified Q-learning outperforms both PIPE and CO-PIPE. Which machine learning technique is better, and under what

circumstances, is still an open question.

A cellular automaton (CA) is an oft-overlooked paradigm for homogeneous team learning (only a few CA papers examine heterogeneous agents). A CA consists of a neighborhood (often a row or grid) of agents, each with its own internal state, plus a state-update agent behavior (the rule) applied to all the agents synchronously. This rule is usually based on the current states of an agent's neighbors. CAs have many of the hallmarks of a multiagent system: interactions and communications are local, and behaviors are performed independently. A good survey of existing work in learning cellular automata rules is presented in (Mitchell *et al.*, 1996).

One common CA problem, the Majority (or Density) Classification task, asks for an update rule which — given initial configurations of agents, each agent with an internal state of 1 or 0 — correctly classifies as many of them as possible based on whether the initial configuration had more 1s than 0s or more 0s than 1s. This is done by repeatedly applying the update rule for several iterations; if the agents have converged to all 1s, the rule is said to have classified the initial configuration as majority-1s (similarly for 0s). If it has not converged, the rule has not classified the initial configuration. The goal is to discover a rule that classifies most configurations correctly given specific standard settings (in terms of number of agents, size of neighborhood, etc). Due to the complexity of the emergent behavior, it is exceptionally difficult to create good performing solutions by hand. For a benchmark problem using a specific setting, the best human-coded result, of 82.178% accuracy, was proposed in (Das *et al.*, 1994). Much better results have been produced with evolutionary computation methods: a rule with accuracy 82.326% obtained by using genetic programming is reported in (Andre *et al.*, 1996). At present the best known result, with accuracy 86.3%, was obtained via competitive coevolution (Juille and Pollack, 1998).

Finally, many homogeneous team learning papers are concerned the increased power added by indirect and direct communication abilities. I discuss such literature in Section 2.3.

2.1.2 Purely-Heterogeneous Team Learning

In purely-heterogeneous team learning, a single learner tries to improve the team as a whole by learning a unique behavior for each of the agents. This approach allows for more diversity in the team at the cost of increasing the search space (Good, 2000). The bulk of research in heterogeneous team learning has concerned itself with the requirement for or the emergence of specialists.

Several approaches to evolving teams of predator agents chasing a prey are analyzed in (Luke and Spector, 1996). Here, the authors compare homogeneity, heterogeneity with restricted breeding (agents could only breed with like agents from other teams), and heterogeneity with no breeding restrictions. The authors suggest that the restricted breeding works better than having no restrictions for heterogeneous teams, which may imply that the specialization allowed by the heterogeneous team representation conflicts with the inter-agent genotype mixture allowed by the

free interbreeding. Similarly, genetic programming is applied to develop a team of soccer playing agents for the RoboCup simulator in (Andre and Teller, 1999). The individuals encode eleven different behaviors (one for each player). The authors also mention that the crossover operator (between teams of agents) was most successful when performing restricted breeding.

The evolution of homogeneous and heterogeneous teams for the predator-prey pursuit domain is investigated in (Haynes and Sen, 1996b; Haynes and Sen, 1997a; Haynes and Sen, 1997b). The authors present several crossover operators that may promote the appearance of specialists within the teams. The results indicate that team heterogeneity can significantly help despite apparent domain homogeneity. The authors suggest that this may be due to deadlocks generated by identical behaviors of homogeneous agents when positioned in the same location. Haynes and Sen report that the most successful crossover operator allows arbitrary crossover operations between behaviors for different agents, a result that contradicts the findings described in (Luke and Spector, 1996; Andre and Teller, 1999).

The so-called "one-population" coevolution can be used with heterogeneous team learning in an unusual way: a single population is evolved using an ordinary evolutionary algorithm, but agents are tested by teaming them up with partners chosen at random from the same population, to form a heterogeneous team (Bull, 1997). In (Miconi, 2001), members of teams selected from such a coevolving population are evaluated based on how much worse the team performs when that member is not in it. The author reports the appearance of squads and "subspecies" within the team. The results also indicate that the overall performance of this approach is superior to that of a homogeneous team learning algorithm (this is supported by the experiments in (Quinn, 2001a)). It is not clear, however, where one-population cooperative coevolution should be categorized. It is clearly a single learner (one EA population), yet the method exhibits many of the game-theoretic oddities discussed in Section 2.2.

Unfortunately, scaling heterogeneous team learning to large numbers of agents is extremely difficult, especially for large swarms of agents. Searching for more than a dozen agent behaviors is usually intractable with current computational capabilities. Certain simplifying assumptions are required to reduce the complexity of the search; for example, although all agents may not share the same behavior, large numbers of them might. Such approaches are covered next.

2.1.3 Hybrid Team Learning

In hybrid team learning, the set of agents is divided into several groups, with each agent belonging to exactly one group. All agents in a group have the same behavior. One extreme (a single group), is equivalent to homogenous team learning, while the other extreme (one agent per group) is equivalent to heterogeneous team learning. Hybrid team learning thus permits the experimenter to achieve some of the advantages of each method.

In (Luke, 1998; Luke *et al.*, 1997), the authors report on evolving soccer teams for the RoboCup competition, and mention that the limited amount of time available before the competition diminished the probability of obtaining good heterogeneous teams. Instead, they compare the homogeneous approach with a hybrid combination that divides the team into six groups of one or two agents each, and then evolves six behaviors, one per group. The authors report that homogeneous teams performed better than the hybrid approach, but mention that the latter exhibited initial offensive-defensive group specialization and suggest that hybrid teams might have outperformed the homogeneous ones, if learning was given more time.

Faced with the specialization/search-space tradeoff inherent in heterogeneity, Hara and Nagao present an automated grouping technique called Automatically Defined Groups (ADG) (Hara and Nagao, 1999). In ADG, the team of agents is composed of several groups of homogeneous agents (similar to (Luke, 1998; Luke *et al.*, 1997)); however, ADG automatically discovers a good number of groups and their compositions. Additionally, the acquired group structure may give insights into the cooperative behavior that solves the problem. The authors successfully apply this technique to a simple load transportation problem and to a modified tile-world domain. A similar approach is reported in (Bongard, 2000), where GP individuals contain partitioning instructions for the creation of groups.

2.2 Concurrent Learning

The alternative to team learning in cooperative multiagent systems is concurrent learning, where multiple learning processes attempt to concurrently improve parts of the team. Most often, each agent has its own unique learning process to modify its behavior. There are other degrees of granularity of course: the team may be divided into "groups", each with its own learner (for example, in (Potter *et al.*, 2001)).

Concurrent learning and team learning each have their champions and detractors. Concurrent learning might outperform both homogeneous and heterogeneous team learning, as shown in (Bull and Fogarty, 1994; Iba, 1996; Iba, 1998); in contrast, team learning is preferable in certain other situations (Miconi, 2003). When then would each method be preferred over the other? In (Jansen and Wiegand, 2003), the authors argue that concurrent learning may be preferable in those domains for which some decomposition is possible and helpful, and when it is useful to focus on each subproblem to some degree independently of the others. The reason for this is that concurrent learning projects the large joint team search space onto separate, smaller individual search spaces. If the problem can be decomposed such that individual agent behaviors are relatively disjoint, then this can result in a dramatic reduction in search space and in computational complexity. A second, related advantage is that breaking the learning process into smaller chunks permits more flexibility

in the use of computational resources to learn each process because they may, at least partly, be learned independently of one another.

The central challenge for concurrent learning is that each learner is adapting its behaviors in the context of other co-adapting learners over which it has no control. In single-agent scenarios (where traditional machine learning techniques are applicable), a learner explores its environment, and while doing so, improves its behavior. Things change with multiple learners: as the agents learn, they modify their behaviors, which in turn can ruin other agents' learned behaviors by making obsolete the assumptions on which they are based. One simplistic approach to deal with this is to treat the other learners as part of a dynamic environment to which the given learner must adapt. This idea was used in early multiagent learning literature (Schmidhuber, 1996; Schmidhuber and Zhao, 1996; Zhao and Schmidhuber, 1996). But things are more complicated in concurrent learning: the other agents are not merely changing, but are in fact co-adapting to the originial learner's adaptation to them. Therefore, the agents' adaptation to the environment can change the environment itself in a way that makes that very adaptation invalid. This is a significant violation of the basic assumptions behind most traditional machine learning techniques.

The concurrent learning literature breaks down along different lines than the team learning literature. This is because each agent is free to learn separately, and thus the heterogeneity of the team becomes an emergent aspect rather than a design decision in concurrent learning. I argue that there are instead three main thrusts of research in the area of concurrent learning. First, there is the *credit assignment* problem, which deals with how to apportion the reward obtained at a team level to the individual learners. Second, there are challenges in the *dynamics of learning*. Such research aims to understand the impact of co-adaptation on the learning processes. Third, some work has been done on *modeling other agents* in order to improve the collaboration with them.

2.2.1 Credit Assignment

When dealing with multiple learners, one is faced with the task of divvying up among them the reward received for their joint actions. The simplest solution is to split the team reward equally among each of the learners, or in a larger sense, divide the reward such that whenever a learner's reward increases (or decreases), all learners' rewards increase (decrease). This credit assignment approach is usually termed global reward.

There are many situations where it might be desirable to assign credit in a different fashion, however. Clearly if certain agents did the lion's share of the task, it might be helpful to specially reward them for their actions, or to punish others for laziness. Similarly, global reward might not scale well to increasingly difficult problems because the learners do not have sufficient feedback tailored to their own specific actions (Wolpert and Tumer, 2001). In other situations credit assignment must be done differently because global reward cannot be efficiently computed,

particularly in distributed environments. For example, in a robotics foraging domain, it may not be easy to gather the global information about all items discovered and foraged.

If we're not to equally divide the team reward among the agents, what options are there, and how do they impact on learning? One extreme is to assess each agent's performance based solely on its contribution. This approach discourages laziness because it rewards agents only for those tasks they have actually accomplished. However, agents do not have any rational incentive to help other agents, and greedy behaviors may develop. I call this approach local reward.

In (Balch, 1997; Balch, 1999), the author experiments with different credit assignment policies to explain when one works better than the others. He argues that local reward leads to faster learning rates, but not necessarily to better results than global reward. For one problem (foraging), local reward produces better results, while in another (robotic soccer) global reward is better. The author suggests that using local reward increases the homogeneity of the learned teams, which in turn indicates that the choice of credit assignment strategy should depend on the degree of specialization that is desired for the resulting team.

A few other credit assignment schemes have been proposed as well. In (Mataric, 1994a), the agents' separate learning processes are improved by combining individual local reinforcement with types of social reinforcement: observational reinforcement is obtained when observing and imitating behaviors of other agents; agents might receive vicarious reinforcements whenever other agents are directly rewarded. In (Chang *et al.*, 2003), each agent assumes the observed reward signal is a sum of the agent's direct contribution and some random Markov process that estimates the contributions of teammates. Each agent employs a Kalman filter to separate the two terms and to compute its true contribution to the global reward. The authors show that this reward filtering approach provides a better feedback for learning in simple cooperative multiagent domains. Yet another credit assignment mechanism approximates an agent's contribution to the team reward based on how the team would have fared differently were that agent not present. This is termed the Wonderful Life Utility, and it has been shown superior to both local and global reward, particularly when scaling to large numbers of agents (Tumer *et al.*, 2002; Wolpert and Tumer, 2001).

The wide variety of credit assignment methods have a significant impact on the coverage of research in the dynamics of learning, which follows in the next section. The initial focus will be on the study of concurrent learning processes in fully cooperative scenarios (when using global reward). But as just mentioned, global reward may not always work best for concurrent learning agents: instead, various other schemes have been proposed. However, these credit assignment schemes may run counter the researchers' intention for the agents to cooperate, resulting in dynamics resembling those in general-sum scenarios, which are also discussed.

To understand why credit assignment policies can complicate the dynamics of learning,

suppose two foraging agents are rewarded solely based on the speed at which they are gathering items. A narrow bridge separates the source of items from the location where items are to be deposited. If both robots arrive at the bridge at the same time, what is the rational thing for them to do? If they have a local credit assignment procedure, neither is willing to let the other go first: that would cost time and would decrease its own credit. This is reminiscent of social scientists' notion of being in the individual interest of none (none of the robots wants to let the other go first), but in the collective interest of all agents (both robots loose time if they struggle to get on the bridge first) (Lichbach, 1996).

2.2.2 The Dynamics of Learning

When applying single-agent learning to stationary environments, the agent experiments with different behaviors until hopefully discovering a globally optimal one. In dynamic environments, the agent may at best try to keep up with the changes in the environment and constantly track the shifting optimal behavior. Things are even more complicated in multiagent systems, where the agents may adaptively change each others' learning environments.

The range of tools to model and analyze the dynamics of concurrent learners is unfortunately very limited. I believe two tools have the potential to provide further understanding on such issues. The first one, evolutionary game theory (EGT), models the learners as concurrent decision processes in an ideal scenario where they are given unlimited resources (Maynard Smith, 1982; Hofbauer and Sigmund, 1998). EGT was successfully used to study the properties of cooperative coevolution (Wiegand, 2004), and to study trajectories of concurrent Q-learning processes (Tuyls *et al.*, 2003; 't Hoen and Tuyls, 2004). I employ the EGT model extensively in Chapter 3. The other tool is a framework to model and predict the behavior of concurrent multiagent learners (Vidal and Durfee, 1998; Vidal and Durfee, 2003). The system uses parameters such as rate of behavior change per agent, learning and retention rates, and the rate at which other agents are learning as well. Combining this information permits approximating the error in the agents' decision function during the learning process.

Many studies in concurrent learning have investigated the problem from a game-theoretic perspective. An important concept for these investigations is that of Nash equilibrium, which is a joint strategy (one strategy for each agent) such that no single agent has any rational incentive (in terms of better reward) to change its strategy away from the equilibrium. As the learners do not usually have control over each others' behaviors, creating alliances to escape this equilibrium is not trivial. For this reason, many of the investigations of concurrent learning are concerned with proving that the algorithms converge to Nash equilibrium points. However, especially in approaches specific to cooperative multiagent systems, it is also helpful to ask whether the agents actually found a globally-optimal collaboration, or only a suboptimal Nash equilibrium.

If one uses only global reward, the rewards received by agents are correlated, and so increasing one's reward implies increasing everybody else's reward. In such cases, it is relatively straightforward to check that the concurrent learning approach has converged to the globally optimal team. I discuss this literature next. Following, I cover learning in more general settings where the relationship among rewards received by learners is less clear. Such research is particularly apropos to learning in combination with credit assignment schemes other than using the global reward.

Fully Cooperative Scenarios

Theoretical investigations of the dynamics of learning are usually performed on very simple multiagent domains, employing a few (usually two) agents, a limited number of actions for each agent, and environments with only a few (if any) states. However, the convergence of concurrent Q-learners to global optimum is not always achieved even when the agents can immediately perceive the actions of all other agents in the environment (Claus and Boutilier, 1998). This result is disturbing, given the fact that agents usually do not have such complete information about the team and the environment. Follow-up research has proposed a number of alterations to concurrent multiagent Q-learning to increase the probability of finding the global optimum. For example, estimates of the highest reward observed for each action, as well as the frequency of observing those rewards in the past, can be used to update an agent's policy (Lauer and Riedmiller, 2000), or its exploration strategy (Kapetanakis and Kudenko, 2002a; Kapetanakis and Kudenko, 2002b). Alternatively, a stochastic sampling technique that is guaranteed to converge to optimal Nash equilibria in deterministic domains is introduced in (Brafman and Tennenholtz, 2002); however, the algorithm is polynomial in the number of actions of the agents, and it also assumes certain a priori coordination of the agents' learning processes (the agents agree to a joint exploration phase, followed by jointly exploiting the action that yielded maximum reward).

Scaling up to environments with states is computationally demanding as well. The Optimal Adaptive Learning algorithm is guaranteed to converge to optimal Nash equilibria if there are a finite number of actions and states (Wang and Sandholm, 2002). Unfortunately, the optimality guarantees comes at a cost in scalability: the overall learning task is decomposed into a set of simpler components that is usually exponential in the number of agents. Environments where the state can only be partially observed (usually due to the agents' limited sensor capabilities) represent even more difficult (also more realistic) settings. The task of finding the optimal policies in partially observable Markov decision process is PSPACE-complete (Papadimitriou and Tsitsiklis, 1987), and it becomes NEXP-complete for decentralized partially observable Markov decision processes (Bernstein *et al.*, 2000). Preliminary research on concurrent learning algorithms for such domains is presented in (Peshkin *et al.*, 2000; Nair *et al.*, 2003).

Research in cooperative coevolutionary algorithms has also addressed similar problems, although usually in the context of making the learning task tractable by decomposing the search space in multiple, simpler subspaces. Some work here has been done on tuning parameters of the system in order to increase the performance of the algorithm (Bull, 1997; Bull, 1998; Wiegand *et al.*, 2001). Recent work has analyzed the conditions under which coevolutionary systems gravitate towards suboptimal Nash equilibria, rather than providing globally optimal solutions for the team as a whole. For example, certain cooperative coevolutionary algorithms may converge to suboptimal solutions even when given infinite computational resources (Wiegand, 2004; Wiegand *et al.*, 2002a; Wiegand *et al.*, 2002b). This pathology, termed "relative overgeneralization", is due to the fact that each of the concurrent learners tends to prefer solutions (agent behaviors) that work reasonably-well in general, rather than solutions that form globally optimal team behaviors.

Several variations attempt to sidestep such difficulties. For example, the coevolutionary algorithm might be augmented with a "hall of fame" repository consisting of the N best teams discovered so far (Gordin *et al.*, 1997; Puppala *et al.*, 1998). In this case, individuals are evaluated by pairing them up with teammates chosen from this hall of fame; this approach generally outperforms ordinary cooperative coevolution methods. A related approach is reported in (Blumenthal and Parker, 2004), where learners periodically provide representative behaviors for the other learners to use for training.

To summarize, concurrent learning in fully cooperative scenarios is extremely difficult, even in very simple problem domains. Many of the algorithms often converge to suboptimal solutions, either because they perform heuristic searches in NP-complete problems, or simply because of certain unfortunate pathologies such as the relative overgeneralization. One approach to design better concurrent learning algorithms is to restart them multiple times, and to only take the best solution. This approach is explored in (Verbeeck *et al.*, 2003; Verbeeck *et al.*, 2005), where coordinated restarts of suboptimal learning algorithms are combined with action exclusions (similar to tabu search) to guarantee convergence to the globally optimal solution. Unfortunately, the restarts may require a significant amount of time.

General Sum Games

Due to unequal-share credit assignment, increasing the reward of an agent may not necessarily result in increasing the reward of all its teammates. Indeed, such credit assignment can inadvertently create highly non-cooperative scenarios. For such reasons, general sum games are applicable to the cooperative learning paradigm, even though in some situations such games may not be in any way cooperative. Following the early work in (Littman, 1994), there has been significant research in concurrent learning for general-sum stochastic games. Most work in the area has been in normal-form games where players act simultaneously, which I cover next.

Hu and Wellman introduce a distributed reinforcement learning algorithm where agents do not learn just a single table of Q-values, but also tables for all other agents (Hu and Wellman, 1998a) (revised and extended in (Bowling, 2000; Hu and Wellman, 2003)). This extra information is used later to approximate the actions of the other agents. An alternative approach is presented in (Nagayuki *et al.*, 2000), where agents approximate the policies, rather than the tables of Q-values, of the other agents. Yet another extension, EXORL, is introduced in (Suematsu and Hayashi, 2002) to learn optimal response policies to both adaptive and fixed policies for the other agent. The Friend-or-Foe Q-learning algorithm (Littman, 2001) attempts to find either a coordination equilibrium (with cooperative agents) or an adversarial one (with competitive agents). Correlated-Q (Greenwald and Hall, 2003) relies on the "correlated equilibrium" solution concept, which assumes additional coordination among agents via a mechanism to specify what action each agent should play.

As argued in (Bowling and Veloso, 2001; Bowling and Veloso, 2002), learning agents should have two desirable properties, namely rationality (the agent should converge optimally when the other agents have converged to stationary strategies) and convergence (under specified conditions, all agents should converge to stationary strategies). The authors then present the WoLF learning algorithm that exhibits these two properties. Additionally, WoLF varies the learning rate from small and cautious values when winning, to large and aggressive values when losing to the other agents.

In (Nowe *et al.*, 2001), two agents participate in a repeated game with two Nash equilibria. In one Nash equilibrium, one agent receives more reward than the other, and the situation is reversed in the other equilibrium. The authors propose a "homo-equalis" reinforcement approach, where communication is used to alternate between the two unfair optimal points in order to fairly share the rewards received by the two agents. An extension of this algorithm for games with multiple states is described in (Peeters *et al.*, 2004).

2.2.3 Teammate Modeling

A final area of research in concurrent learning is teammate modeling: learning about other agents in the environment so as to make good guesses of their expected behavior, and to be able to cooperate with them more effectively. Such an approach is used in (Boutilier, 1996; Chalkiadakis and Boutilier, 2003), where a Bayesian learning method is employed for updating models of other agents. A similar agent modeling approach consisting of learning the beliefs, capabilities and preferences of teammates, is presented in (Suryadi and Gmytrasiewicz, 1999). As the correct model cannot usually be computed, the system stores a set of such models together with their probability of being correct, given the observed behaviors of the other agents.

As other agents are likely modeling *you*, modeling *them* in turn brings up the spectre of infinite recursion: "Agent A thinks that agent B thinks that agent A thinks that agent B thinks that ...". This must be rationally sorted out in finite time. One approach is to categorize agents based on the complexity they assume for their teammates (Vidal and Durfee, 1997). A 0-level agent does not perform any direct reasoning about its teammates (they are rather treated as part of the environment). A 1-level agent models its teammates as 0-level agents. In general, an N-level agent models its teammates as (N-1)-level agents.

The use of 0-level, 1-level and 2-level modeling agents is investigated in (Mundhe and Sen, 2000). The authors report a very good performance for the 0-level learners, suggesting that teammate modeling may not be necessary in some domains. Similar results showing good coordination without modeling other agents are reported in (Sen and Sekaran, 1998; Sen *et al.*, 1994), where two robots learn to cooperatively push a box without either being aware of the other's presence. Experiments in which 1-level agents model each others' action probability distributions are presented in (Banerjee *et al.*, 2000; Mukherjee and Sen, 2001); by creating models of one another, the agents develop a beneficial form of mutual trust. The advantages of modeling other agents and recursively tracking their decisions (for either agents or groups of agents) are also briefly discussed in (Tambe, 1995).

When dealing with larger numbers of teammates, a simpler modeling approach is to presume that the entire team consists of agents identical to the modeling agent. This approach, complemented with a case-based learning mechanism for dealing with exceptions from this assumed behavior, is detailed in (Haynes *et al.*, 1996; Haynes and Sen, 1996a; Haynes and Sen, 1996c).

When learning models of other agents in a multiagent learning scenario, the resulting behaviors are highly sensitive to the agents' initial beliefs (Hu and Wellman, 1996; Wellman and Hu, 1998). Depending on these initial beliefs, the final performance may be better or even worse than when no teammate modeling is performed at all — agent modeling may prevent agents from converging to optimal behaviors. A similar conclusion is reported in (Hu and Wellman, 1998b): the authors suggest that the best policy for creating learning agents is to minimize the assumptions about the other agents' policies. An alternative is to automatically discover if the other agents in the environment are cooperating with you, competing with you, or are in some other relationship to you. The application of reciprocity concepts to multiagent domains where not enough information on the intentions of other agents is known a priori is investigated in (Sekaran and Sen, 1995; Sen and Sekaran, 1995). The authors apply a stochastic decision-making algorithm that encourages reciprocity among agents while avoiding being taken advantage of by unfriendly agents. The results indicate that those agents that prefer not to cooperate end up with worse performance. As pointed out in (Nowak and Sigmund, 1998), there are two types of reciprocity: direct (agent A

helps another agent B and so expects B's help in the future) and indirect (an agent helps another in return for future help from other agents). Such reciprocity improves an agent's "reputation". The authors argue that a necessary condition for cooperation is that the "degree of acquaintanceship" (the probability that an agent knows another agent's reputation) should exceed the ratio of cost of altruistic help relative to the benefit to the recipient. Similar analysis can be done without the need to model reputation per se: in (Ito, 1997), game-players are pitted against one another, where each agent had access to a history of his opponent's previous actions against other players.

I conclude this section with work in agent modeling under communication. A communication protocol for agents to subcontract subtasks to other agents is analyzed in (Ohko *et al.*, 1997). Here, each agent tries to decompose tasks into simpler subtasks and broadcasts announcements about the subtasks to all other agents in order to find "contractors" who can solve them more easily. When agents are capable of learning about other agents' task-solving abilities, communication is reduced from broadcasting to everyone to communicating exact messages to only those agents that have high probabilities to win the bids for those tasks. A related approach is presented in (Bui *et al.*, 1999; Bui *et al.*, 1998), where Bayesian learning is used to incrementally update models of other agents to reduce communication load by anticipating their future actions based on previous ones. Case-based learning has also been used to develop successful joint plans based on one's historical expectations of other agents' actions (Garland and Alterman, 2004).

2.3 Learning and Communication

Communication is a necessity for some problems; for others, communication may nonetheless increase agent performance. I define communication very broadly: altering the state of the environment such that other agents can perceive the modification and decode information from it. Among other reasons, agents communicate in order to coordinate more effectively, to distribute more accurate models of the environment, and to learn subtask solutions from one another.

But are communicating agents really multiagent? Stone and Veloso argue that unrestricted communication reduces a multiagent system to something isomorphic to a single-agent system (Stone and Veloso, 2000). They do this by noting that without any restriction, the agents can send complete external state information to a "central agent", and execute its commands in lock-step, in essence acting as effectors for the central agent. A central agent is not even necessary: as long as agents can receive all the information they need to known about the current states of all the other agents, they can make independent decisions knowing exactly what the other agents will do, in essence enabling a "central controller" on-board each individual agent, picking the proper sub-action for the full joint action. Thus, a true multiagent problem necessitates restrictions on communication. At any rate, while full, unrestricted communication can orthogonalize the learning

problem into a basic single-agent problem, such an approach requires very fast communication of large amounts of information. Real-time applications instead place considerable restrictions on communication, in terms of both throughput and latency.

Explicit communication can also significantly increase the search space for the learning method, both by increasing the size of the external state available to the agent (it now knows state information communicated from other agents), and by increasing the agent's available choices (perhaps by adding a "communicate with agent *i*" action). As noted in (Durfee *et al.*, 1987), this increase in search space can hamper learning by more than communication itself may help. Thus, even when communication is required for optimal performance, the learning method must disregard communication, or hard-code it, in order to simplify the learning process for many applications. When learning in a predator-prey pursuit domain, for example, experiments in (Luke and Spector, 1996) assume that predators can sense each other's position no matter what distance separates them, while a blackboard communication scheme is used instead in (Berenji and Vengerov, 2000b) to allow the agents to know other agents' locations.

2.3.1 Direct Communication

Many agent communication techniques employ, or assume, an external communication method by which agents may share information with one another. The method may be constrained in terms of throughput, latency, locality, agent class, etc. Examples of direct communication include shared blackboards, signaling, and message-passing. The literature has examined both hard-coded communication methods and learned communication methods, and their overall effects on learning.

Cooperating learners can use communication in a variety of ways in order to improve team performance, as indicated in (Tan, 1993). For example, agents can inform others of their current state by sharing immediate sensor information. Another approach is for agents to share information about past experiences in the form of episodes (sequences of ⟨state,action,reward⟩ previously encountered by the agent) that others may not have experienced yet. Yet another alternative is for agents to share knowledge related to their current policies (for example, cooperating Q-learning agents might communicate ⟨state,action,utility⟩ values).

Some research, particularly in Q-learning, has simply presumed that the agents have access to a joint utility table or to a joint policy table to which each may contribute in turn, even though the agents are separate learners. For example, a cooperative learning setting where multiple agents employ communication to use and update the same policy is investigated in (Berenji and Vengerov, 2000a). The authors suggest that the simultaneous update of a central policy reduces early tendencies for convergence to suboptimal behaviors. I argue that this is an implicit hard-coded communication procedure: the learners are teaching each other learned information.

Much of the remaining research provides the agents with a communication channel but does not hard-code its purpose. In a simple situation, agents' vocabulary may consist of a single signal detectable by other agents (Wagner, 2000). In other work, mobile robots in a cooperative movement task are endowed with a fixed but undefined communication vocabulary (for example, in (Yanco and Stein, 1993)) . The robots learn to associate meanings with words in various trials. When circumstances change, the robots learn to adjust their communication language as appropriate. A similar approach is reported in (Jim and Giles, 2000) to learn a language for communication in a predator-prey domain. The authors use a blackboard communication scheme and present a rule for determining a pessimistic estimate on the minimum language size that should be used for multiagent problems. The emergence of a spontaneous coherent lexicon that may adapt to cope with new meanings during the lifetime of the agents is reported in (Steels, 1996a). Further, agents are able to create general words through collective agreement on their meanings and coverage (Steels and Kaplan, 1999). Similar approaches to evolving communication languages are presented in (Cangelosi, 2001; de Boer, 1997; Saunders and Pollack, 1996; Steels, 1995; Steels, 1996b; Steels, 1996c; Steels, 1997; Steels, 2000; Williams, 2004).

Many such methods provide an incentive to the agents to communicate (perhaps by sharing the resulting reward). However, reward incentives are not necessary for the agents to communicate (Ackley and Littman, 1994).

2.3.2 Indirect Communication

Indirect communication involves the implicit transfer of information from agent to agent through modifications of the external world. Examples of indirect communication include: leaving footsteps in snow, leaving a trail of bread crumbs in order to find one's way back home, and providing hints through the placement of objects in the environment (perhaps including the agent's body itself).

Much of the indirect communication literature has drawn inspiration from social insects' use of pheromones to mark trails or to recruit other agents for tasks (Hölldobler and Wilson, 1990). Pheromones are chemical compounds whose presence and concentration can be sensed by fellow insects (Bonabeau *et al.*, 1999), and like many other media for indirect communication, pheromones can last a long time in the environment, though they may diffuse or evaporate. In some sense, pheromone deposits may be viewed as a large blackboard or state-utility table shared by all the agents; but they are different in that pheromones can only be detected locally.

Several pheromone-based learning algorithms have been proposed for foraging problem domains. A series of reinforcement learning algorithms have adopted a fixed pheromone laying procedure, and use current pheromone amounts for additional sensor information while

exploring the space or while updating the state-action utility estimates (Leerink *et al.*, 1995; Monekosso *et al.*, 2002; Monekosso and Remagnino, 2001; Monekosso and Remagnino, 2002). Evolutionary algorithms have also been applied to learn exploration/exploitation strategies using pheromones deposited by hard-coded mechanisms. For example, EAs can tune the agents' policies in an application involving multiple digital pheromones (Sauter *et al.*, 2002; Sauter *et al.*, 2001). A similar idea applied to network routing is presented in (White *et al.*, 1998).

Another research direction is concerned with studying whether agents can learn not only to use pheromone information but to deposit the pheromones in a rational manner. This question was first examined in AntFarm, a system that combines communication via pheromones and evolutionary computation (Collins and Jefferson, 1992; Collins and Jefferson, 1991). AntFarm ants use a single pheromone to mark trails toward food sources, but use a compass to point themselves along the shortest path back to the nest. Related algorithms that exhibit good performance at various foraging tasks are detailed in (Panait and Luke, 2004a; Panait and Luke, 2004c). These algorithms use multiple pheromones, and in doing so they eliminate the explicit requirement for agents to precisely know the direction towards the nest from any location. Rather, the agents use pheromone information to guide themselves to the food source and back to the nest. Agents using these algorithms are able to optimize paths, and also to cope with obstacles and dynamic environments. Finally, evolutionary computation is shown to automatically discover such algorithms in (Panait and Luke, 2004b).

Another approach to indirect communication among agents involves using the agents' body positions themselves to convey meaning. As argued in (Wilson, 1975; Quinn, 2001b), communication "is neither the signal by itself, nor the response, [but] instead the relationship between the two", and as such, communication does no require a dedicated channel. One form of indirect communication involves foraging robots that use their bodies to mark the path that other robots should follow (Werger and Mataric, 1999). Alternatively, robots in a collaborative movement scenario might coordinate by communicating their adopted roles of leader or follower via sequences of moves; for example, after initial phases of alignment, the robots use rotation and oscillatory back-and-forth movements to decide who leads the way and who follows (Quinn, 2001b).

2.4 Open Research Issues

Multi-agent learning is a new field and as such its open research issues are still very much in flux. This section singles-out three important open questions that need to be addressed in order to make multiagent learning more broadly successful as a technique.

2.4.1 Adaptive Dynamics and Nash Equilibria

Multi-agent systems are typically dynamic environments, with multiple learning agents vying for resources and tasks. This dynamism presents a unique challenge not normally found in single-agent learning: as the agents learn, their coadaptation to one another effectively changes the learning goalposts. How can agents cope with an environment where the goalposts are constantly and adaptively being moved?

Much of the past literature has concerned itself with concurrent learning algorithms that converge. However, such convergence might be in many cases not to optimal but to suboptimal solutions. For example, the theoretical analysis in (Wiegand, 2004) reports on the existence of undesirable pathologies that plague concurrent learning algorithms and make them drift away from optimal solutions. Additionally, the research in (Lichbach, 1996; Shoham *et al.*, 2004) suggest that it might be the very rationality of the learners that makes them prefer suboptimal solutions.

These issues are critical for significant advancements in the multiagent learning field. What are the underlying causes for pathologies reported to haunt concurrent learning? Moreover, how can such pathologies be eliminated to allow concurrent learning algorithms to search for the optimal solutions? Can such learners exhibit rationality, or should they be created to exhibit some degree of irrationality?

The research in this thesis centers on addressing these very questions. In Chapter 3, I present a theoretical analysis revealing that concurrent learners might drift away from optimal solutions simply because the learners might incorrectly estimate that actions associated with optimal solutions are in fact worse than actions associated with suboptimal solutions. Following, Chapters 4–6, I detail novel concurrent learning algorithms that benefit from correcting the learners' estimates. Interestingly, one primary characteristic of these corrections is in some sense "irrational": I show that agents are better off if they *ignore* certain information.

2.4.2 Problem Decomposition

The state space of a large, joint multiagent task can be overwhelming. As a consequence, multiagent learning algorithms might be applied to explore gigantic search spaces. This poses a significant challenge for the learning task. How can the search space be reduced to render it tractable for multiagent learning algorithms?

An obvious way to tackle this is to use domain knowledge to simplify the state space, often by providing a smaller set of more "powerful" actions customized for the problem domain. For example, an application of Q-learning to select from hand-coded reactive behaviors for robot foraging tasks (such as avoid, head-home, search or disperse) is presented in (Mataric, 1994b; Mataric, 1998). Another alternative has been to reduce complexity by heuristically decomposing

the problem, and hence the joint behavior, into separate, simpler behaviors for the agents to learn. Such decomposition may be done at various levels (decomposing team behaviors into sub-behaviors for each agent; decomposing an agents' behavior into sub-behaviors; etc.), and the behaviors may be learned independently, iteratively (each depending on the earlier one), or in a bottom-up fashion (learning simple behaviors, then grouping into "complex" behaviors).

One such approach is to learn basic behaviors first, then set them in stone and learn more complex behaviors based on them. This method is commonly known as layered learning, and was proposed in (Stone, 1997; Stone, 1998) and successfully applied to robotic soccer. Further applications of the technique to keep-away soccer are reported in (Gustafson, 2000; Gustafson and Hsu, 2001; Hsu and Gustafson, 2002; Whiteson and Stone, 2003). Another approach, shaping, gradually changes the reward function from favoring easier behaviors to favoring more complex ones based on those easy behaviors. Balch uses a shaped reinforcement reward function (earlier suggested in (Mataric, 1997)) which depends on the number of partial steps fulfilled towards accomplishing the joint task (Balch, 1999). The author shows that using either shaped or local reward leads to similar results, but the former drastically reduces the learning time as well. A related method, fitness switching, adaptively changes the reward function to favor those behaviors the learners have made the least progress on (Zhang and Cho, 1998).

Shaping, layered learning, and fitness switching, are not multiagent learning techniques, though they have often been applied in such a context. Less work has been done on formal methods of decomposing tasks (and behaviors) into subtasks (sub-behaviors) appropriate for multiagent solutions, how agents' sub-behaviors interact, and how and when learning of these sub-behaviors may be parallelized. Consider robot soccer as an example: while it is true that agents must learn to acquire a ball and to kick it before they can learn to pass the ball, their counterparts must also have learned to receive the ball, and to ramp up difficulty, opponents may simultaneously co-adaptively learn to intercept the ball. Few papers have examined how to form these "decomposition dependency graphs", much less have the learning system develop them automatically. Yet, understanding these interactions is important in order to parallelize the learning process, to simplify the search space, and to produce more robust multiagent behaviors. One notable exception is reported in (Guestrin *et al.*, 2002): here, the authors note that the actions of some agents may be independent in specific situations. They suggest taking advantage of this fact by partially decomposing the joint Q-values of agents based on a coordination graph that heuristically spells out which agents must interact in order to solve the problem. This partial decomposition represents a middle-ground between learning a full joint utility table and learning separate independent tables. Also, a different hierarchical approach to simplifying the inter-agent coordination task is suggested in (Makar *et al.*, 2001; Ghavamzadeh and Mahadevan, 2004), where agents coordinate their high-level behaviors, rather than each primitive action they may perform.

Concurrent learning techniques provide an intuitive divide-and-conquer approach to simplifying the multiagent learning task: the overall search space is divided into multiple simpler subspaces, and each of them is explored in parallel. The novel concurrent learning algorithms that are detailed in this thesis represent significant contributions that help this problem decomposition technique work better.

2.4.3 Scalability

Scalability is a problem for many learning techniques, but especially so for multiagent learning. The dimensionality of the search space grows rapidly with the number of agents involved, the number and complexity of agent behaviors, and the size of the network of interactions between them. This search space grows so rapidly that one cannot learn the entire joint behavior of a large, heterogeneous, strongly intercommunicating multiagent system. Effective learning in an area this complex requires some degree of sacrifice: either by isolating the learned behaviors among individual agents, by reducing the heterogeneity of the agents, or by reducing the complexity of the agents' capabilities. Techniques such as learning hybrid teams, decomposition, or partially restricting the locality of reinforcement provide promising solutions in this direction, but it is not well understood under which constraints and for which problem domains these restricted methods will work best.

Scaling multiagent learning techniques to large teams and complex problem domains is a challenging task indeed. And while the research in this thesis does not go beyond artificial problem domains involving only two agents, I argue that it has significant impact on scaling multiagent learning techniques to more complex scenarios. My argument relies on the observation that hybrid team learning provides a decomposition of the overall learning task that appears susceptible to further parallelization: one learning process might be associated with each group to search for the behavior that all agents within that group share, and these learning processes could proceed in parallel. I believe that the advancements in the design of concurrent learning algorithms as presented in these thesis provide a significant boost to scaling such a concurrent hybrid learning approach to complex problem domains and large teams of agents.

Chapter 3: A Theoretical Analysis of Concurrent Learning

In a single-agent scenario, the learner evaluates different behaviors, each indicating what action (or sequence of actions) the agent should perform in specific situations. Upon performing such actions, the learner receives rewards that are used to update its quality assessment associated with one or multiple behaviors. For example, an agent employing evolution computation uses individuals to encode different behaviors, and then updates the fitness of each individuals based on rewards received upon acting according to that individual. Similarly, an agent employing Q-learning (or any other temporal difference learning method) uses the rewards received upon performing actions in the environment in order to update the utilities it has associated for immediate and future rewards that would be received upon performing that action. The objective of the learner is to identify a behavior that maximizes the rewards the agent receives. For simplicity, consider that each behavior represents a single action; consequently, the learner needs to identify the action that receives maximum reward.

Many such single-agent learning algorithms have guarantees of convergence to the global optimum under idealized conditions (which usually include proper settings and enough resources). For example, formal models employing infinite populations have been used to establish such guarantees for evolutionary algorithms (Vose, 1999). Similarly, Q-learning (and most temporal difference learning methods in general) will converge to the optimal policy when using proper learning rates and exploration policies (Watkins and Dayan, 1992; Singh *et al.*, 2000).

As discussed in Chapter 2, these guarantees of optimality are unfortunately lost for straightforward extensions of many single-agent learning algorithms to concurrent multiagent learning. Such problems were often reported in empirical experimentation. In addition, previous theoretical analysis suggested that concurrent learning algorithms might drift towards suboptimal solutions due to undesirable pathologies, even when these algorithms are given unlimited resources (Wiegand, 2004).

What drives learning algorithms away from global optima in multiagent domains, given their convergence guarantees in single agent scenarios? Answering this question is the focus of the present chapter. Section 3.1 provides intuitive illustrations of how a learner might estimate the quality of its actions in a simple two-agent coordination problem. These illustrations indicate that learners might incorrectly estimate that actions associated with optimal solutions are inferior to other actions. These poor estimates are the primary cause for concurrent learning drifting away from optima. The key to providing convergence guarantees for these algorithms becomes a matter

of providing each learner with accurate estimates for the quality of its actions. In Sections 3.2 – 3.3, I present a theoretical analysis of the impact of accurate estimates onto the convergence of concurrent learning algorithms. Using an abstract mathematical framework, I prove that improving the learners' estimates results in an increase in the probability that they will converge to the optimum. Interestingly, learners can achieve more accurate estimates if they simply ignore certain reward received upon performing their actions.

3.1 A Learner's Estimation for the Quality of Its Actions

Imagine a simple scenario where two agents learn to coordinate. If X is the set of actions that the first agent can choose from, and Y is the set of actions available to the second agent, the task is for the agents to independently choose one action each (say x and y) such as to maximize the joint reward $f(x,y)$ that they receive. The learning task proceeds as follows: agents independently and concurrently choose an action each, and they both receive the same reward information without knowing which action the other agent has selected. The agents use this information to assess the quality of the actions and to decide which actions they should explore next. The learning proceeds over multiple repetitions of this process involving concurrent action selection, reward observation, and belief update.

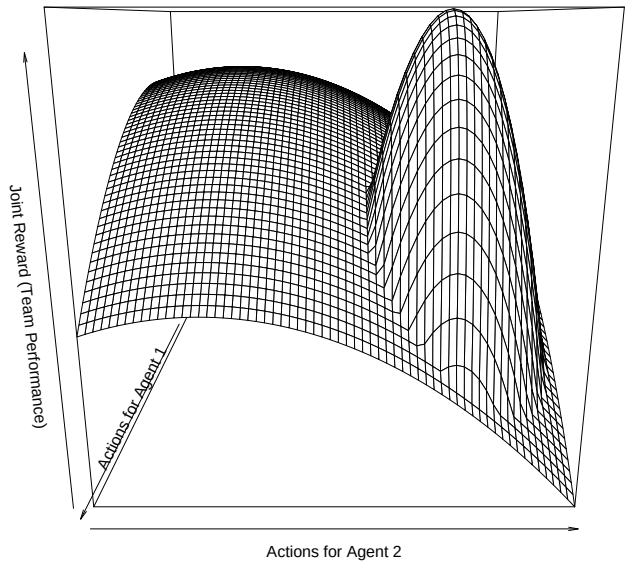

Figure 3.1: A bimodal joint search space for two concurrent learners.

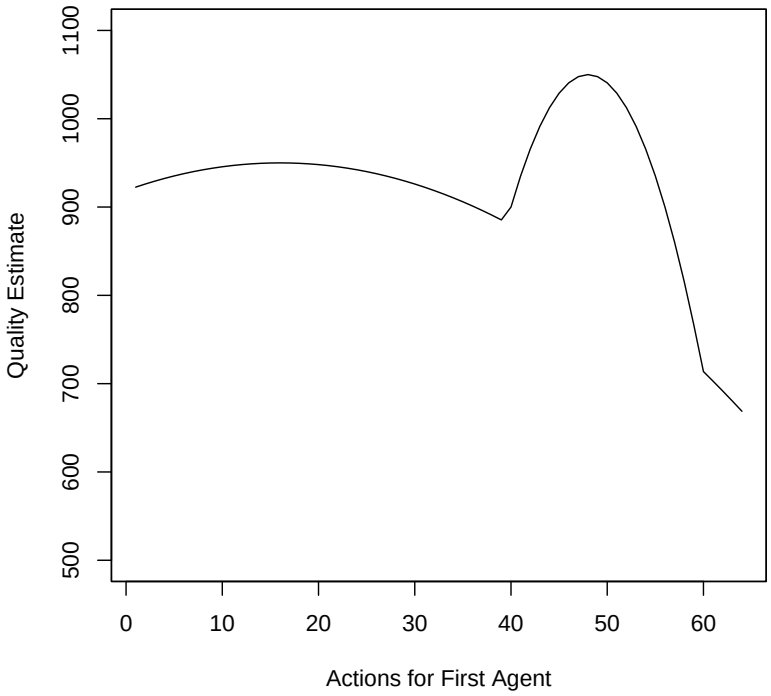

Figure 3.2: A desirable estimation provides precise qualitative information about the location of the optimal peak.

Figure 3.1 illustrates a search space of joint rewards for a simple coordination game. The figure shows two peaks of different sizes. The lower peak represents a suboptimal solution, and the wide coverage of that peak implies that solution quality changes only a little when either agent chooses a slightly different action. The higher peak represents the global optimum; its smaller coverage implies that the solution quality may change rapidly if either agent chooses a slightly different action. Both peaks are Nash equilibria — modifying either the x or the y value (but not both) would lead to a decrease in the function value. This implies that if both agents decide to choose actions corresponding to a Nash equilibrium, neither of them has a rational incentive to choose any other action on its own.

In concurrent learning, the reward an agent receives is based not just on the action it has chosen, but also on other, likely unknown, actions made by its teammate. This other learner is also receiving reward that is partially dependent on the actions of the first learner. Thus, each learner cannot make an objective quality assessment of a given action, because it would depend on the action of the other learners. Instead the learner might make a subjective estimation of the quality of that action based on the rewards that learner has received for that action. As a consequence, an agent's estimation changes with time: some actions might appear better or worse than others, depending on which actions the teammate is exploring at the same time.

Imagine if the learners were permitted to test their actions in combination with every possible

action for the other learner. It turns out that that if each learner bases his estimation on the maximum reward receivable over this large (possibly infinite) set of actions, then their quality estimation provides accurate information about the location of the optimal peak: the actions corresponding to this optimal peak are assigned higher quality than any other action. Figure 3.2 illustrates such a desirable estimation (computed as computed as $g(x) = \max_{y \in Y} f(x,y)$) that an agent might have for its actions in the two-agent coordination game depicted in Figure 3.1. Based on this illustration, I hypothesize that concurrent learners, each employing this desirable quality estimate, will independently converge to actions corresponding to the global optimum, if there is only one optimum in the joint action space. Section 3.2.1 provides the formal proofs for this hypothesis for cooperative coevolutionary algorithms.

However, this desirable estimate requires an exhaustive exploration of the joint search space. In practice, an agents' estimate depends only on a limited number of rewards it has received. The most common approach involve an agent updating the estimate based on a single reward. For example, an agent employing the Q-learning algorithm will update the utility of its actions every time a reward is observed following the selection of that action. This is similar to a CCEA where the fitness of an individual (action) equals the reward received with only a single collaborator (actions for teammates).

Estimating the quality of an action based on a single reward is problematic for two primary reasons. First, the estimation might be very noisy, given that widely different rewards can be received for the same action depending on the actions chosen by the teammate. Second, miscoordination penalties might significantly lower the average (expected) reward for actions corresponding to the global optimum. For these reasons, estimating the quality of an action based on multiple rewards might be a better approach. For example, agents might randomly select their actions several times, then estimate their quality based on some aggregation of the observed rewards. Three aggregation methods are discussed in (Wiegand *et al.*, 2001): using the minimum (a very conservative approach), the maximum (a very optimistic approach), and the average (an intermediate setting between the two extremes).

Figure 3.3 illustrates a learner's quality estimates when using these three aggregation methods. Here, 5, 20, and 100 actions for the teammate are randomly selected with replacement using the uniform distribution (this is a good approximation for the behavior of many learning algorithms at early stages of learning, assuming no information about the search space is available a priori). Given the non-determinism in the selection process, I repeat it three times to have some information about the amount of noise (there are three curves on each of the graphs).

All information about the global optimum completely vanishes when using the minimum (Figure 3.3(a)): actions corresponding to this peak appear worse than those corresponding to the suboptimal peak. Were the agents to learn using this poor estimate, it would be expected

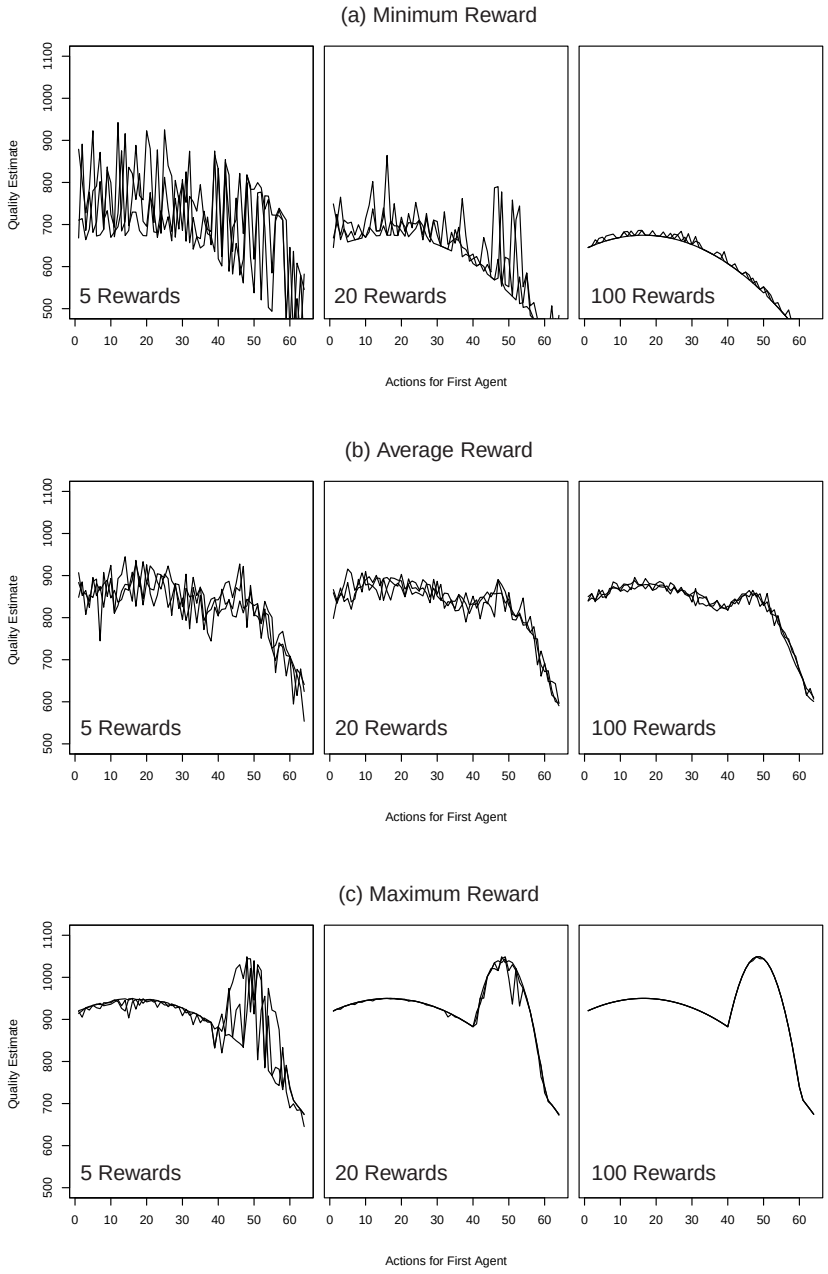

Figure 3.3: One learner's estimation for the quality of its actions, using the (a) minimum, (b) average, and (c) maximum of 5, 20, and 100 joint rewards, respectively.

that they converged to the suboptimal solution, simply because each of them estimates the actions corresponding to this solution as being better than any other actions the agents can choose. In fact, it would be surprising if the learners having this estimate converged to the globally optimal peak!

The situation is fairly similar when using the average to aggregate the rewards (Figure 3.3(b)). This time, the learner's estimate contains some information about the presence of both peaks in the joint search space. However, the actions corresponding to the global optimum still appear slightly

worse than the ones corresponding to the top of the suboptimal peak. It would again be surprising if learners using this quality estimate converged to the global optimum. Note, however, that many theoretical models of multiagent learning employ only such estimates (for example, the theoretical model of cooperative coevolutionary algorithms detailed in (Wiegand, 2004), as well as that of multiagent Q-learning in (Tuyls *et al.*, 2006)). As a consequence, it should be expected that poor quality estimates cause drift towards suboptimal solutions for these theoretical models, as well as for the concurrent learning algorithms that they approximate.

Finally, Figure 3.3(c) illustrates quality estimate of an agent that assesses the quality of an action as the maximum of multiple rewards received for that action. This estimates allow an agent to discern that the actions corresponding to the global optimum are preferable over other actions. Using more rewards decreases the noise in the estimate, making it more accurate. This contrasts the situation involving rewards aggregated via the minimum or the average: more rewards also decreased the noise in the estimate in those cases, but the quality of actions associated with the suboptimal peak remained higher (Figure 3.3(a,b)).

Note also that using the maximum reward resembles in some sense the notion of *lenience*: the agent ignores the lower utilities that it observes, and only uses the maximum one. I argue that this is common when people learn. Consider a simple situation where John and David learn to play soccer together. Let's assume that they do not have any a priori information about each other's skills. At some point, John attempts a pass that allows David to receive the ball in a very favorable position (for example, alone with the goalie). However, David is not able to receive the ball properly because he was not expecting the pass. Given the failure of their collaboration in this particular instance, should John assume that the pass was bad in general and that it should be avoided? Or should John be lenient with his teammate and hope that he will perform better once they learn more about each other's skills? I argue that the second alternative is much more common, and, as will be shown throughout this thesis, more beneficial to agents that learn concurrently.

To summarize, concurrent learning algorithms might drift away from optimal solutions simply because learners have poor estimates for the quality of its actions. However, the graphical illustrations of these estimates can be used only to a certain extent due to their significant limitations. One primary limitation is that they fail to capture the dynamics of the multiagent learning process: the teammates are assumed to select actions at random, rather than via their own learning process. Another limitation is that these illustrations do not represent rigorous tools to analyze multiagent learning in general, although they are certainly very intuitive. In the remainder of this chapter, I overcome these limitations by formalizing the impact that increasingly accurate quality estimates have on concurrent learning algorithms. To this end, I employ the evolutionary game theory toolkit (Maynard Smith, 1982; Hofbauer and Sigmund, 1998). EGT is an abstract mathematical framework based on dynamical systems that provides

elegant models for the dynamics of certain concurrent learning processes, and has been previously employed to analyze undesired pathologies in cooperative coevolutionary systems (Wiegand, 2004), and to visualize the learning trajectories for multiagent Q-learning ('t Hoen and Tuyls, 2004; Tuyls *et al.*, 2006). I extend both of these models to provide each learner with more accurate quality estimates, and I analyze the guarantees and limitations of the resulting algorithms. Section 3.2 presents the enhanced EGT models for cooperative coevolutionary algorithms, while Section 3.3 covers the extended EGT model for multiagent Q-learning.

3.2 EGT Models for CCEAs

I begin with an overview of the EGT model for cooperative coevolutionary algorithms detailed in (Wiegand, 2004). Consider the application of CCEAs to solve a simple two-agent coordination game, such as the one in Figure 3.1. Here is how one could apply CCEAs to address the coordination game:

- Each agent is assigned a population of individuals.
- Individuals in one population represent actions that that specific agent can perform in the environment.
- A genotype is equivalent to a specific action (e.g. *move-left* or *shoot-the-ball*). Each individual represents a genotype, but multiple individuals in the same population might represent the same individual (in that case, the agent is more likely to perform that specific action).
- The fitness of an individual reflects the quality estimate that the agent assigns to that action. Remember that the estimate is subjective – it reflects the quality of that action when in combination with other actions of the teammate.

The EGT model assumes each agent has only a finite set of actions to choose from (the set of genotypes is finite). However, the EGT model assumes that the two populations are infinite, and it computes the proportions of genotypes in the populations at each generation. If the first agent has a finite number n of distinct actions (genotypes) to choose from, its population at each generation is an element of the set $\Delta^n = \{x \in [0,1]^n \mid \sum_{i=1}^{n} x_i = 1\}$. A higher proportion x_i indicates that the agent is more likely to choose action i. Given that the second agent might have a different set (of size m) of actions to choose from, its population is an element of the set $\Delta^m = \left\{y \in [0,1]^m \mid \sum_{j=1}^{m} y_j = 1\right\}$. The fitness computation in the EGT model involves the game payoff matrix A, where the a_{ij} element represents the reward for the joint action (i, j). Assuming

both agents are equally rewarded, A is used to compute the fitnesses in one population, and the transpose of A is used for the other population. Equations 3.1–3.4 describe the EGT model for CCEAs, as expressed in (Wiegand, 2004):

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} y_j^{(t)} \tag{3.1}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} x_i^{(t)} \tag{3.2}$$

$$x_i^{(t+1)} = \left(\frac{u_i^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}} \right) x_i^{(t)} \tag{3.3}$$

$$y_j^{(t+1)} = \left(\frac{w_j^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}} \right) y_j^{(t)} \tag{3.4}$$

where $x^{(t)}$ and $y^{(t)}$ represent the proportions of genotypes (actions) in the two populations at generation t, and $x^{(t+1)}$ and $y^{(t+1)}$ represent the new proportions at the next generation $t+1$. For simplicity, the equations only model the dynamics of cooperative coevolutionary systems under the pressure of selection.

The EGT model can be separated into two phases. First, the fitness of each genotype is computed for the two populations (Equations 3.1 and 3.2). The fitness of an individual with genotype i is estimated as the mean payoff over pairwise collaborations with every individual in the other population. Note that the genotype proportions are used for this computation, given that the populations are infinite. Second, the distributions of the two populations at the next generation are computed (Equations 3.3 and 3.4 model the effects of fitness proportional selection).

Equations $3.1-3.4$ describe a deterministic model for CCEAs: given the composition of the two populations, the model can be used to compute the precise distributions of genotypes in the populations at the next generation. Using this model, it can be shown that pure Nash equilibria are stable, attracting fixed points (Wiegand *et al.*, 2002b). This implies that starting from certain initial configurations and iterating the model until convergence may result in convergence to suboptimal Nash equilibria. This pathology (termed relative overgeneralization in (Wiegand, 2004)), is problematic, especially given the theoretical assumption of infinite populations.

One limitation of this EGT model is that the use of fitness proportional selection poses difficulties when dealing with payoff matrices that contain negative rewards. Unfortunately, some of the most common benchmark problem domains in the multiagent learning literature have negative rewards for miscoordination penalties. For consistency throughout this thesis, I

propose a slightly different EGT model that uses tournament selection of size H instead of fitness proportional selection. The updated EGT model is:

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} y_j^{(t)} \tag{3.5}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} x_i^{(t)} \tag{3.6}$$

$$x_i^{(t+1)} = \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u_i^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \leq u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k^{(t)} \right)^H \right) \tag{3.7}$$

$$y_j^{(t+1)} = \frac{y_j^{(t)}}{\sum_{k:w_k^{(t)}=w_j^{(t)}} y_k^{(t)}} \left(\left(\sum_{k:w_k^{(t)} \leq w_j^{(t)}} y_k^{(t)} \right)^H - \left(\sum_{k:w_k^{(t)} < w_j^{(t)}} y_k^{(t)} \right)^H \right) \tag{3.8}$$

Deriving Equations 3.7 and 3.8 is relatively straightforward. Given the symmetry, I detail only how to derive Equation 3.7. For each genotype i, I start by computing the probability that the best individual of H random ones (selected with replacement from the current population) has the same fitness as an individual with genotype i. Given that $u_k^{(t)}$ is the fitness of each individual with genotype k, this probability equals $\left(\left(\sum_{k:u_k^{(t)} \leq u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k^{(t)} \right)^H \right)$. The only thing remaining is to account for identical fitnesses for different genotypes. Tournament selection works as follows: if multiple individuals have entered the tournament with equally-maximal fitness, one of them is chosen at random. If the selection of individuals is performed sequentially, this is also equivalent to simply choosing the first one of them. As a consequence, Equation 3.7 contains an extra term that indicates the probability that an individual with genotype i is chosen at random from the first population out of all the individuals that have the same fitness as that individual. This term essentially indicates that the ratios of equally-fit individuals with respect to each other remain constant from one generation to the next.

Observe that the fitness estimation in Equations 3.1–3.2 (and also in Equations 3.5–3.6) bears a striking resemblance to aggregating multiple rewards by using the average, as detailed in Section 3.1. However, Figure 3.3 indicated that averaging the rewards might result in learners having poor estimates for the quality of their actions. Next, I enhance the EGT model to account for learners with accurate estimates. I achieve this solely by modeling other approaches to computing the fitness.

3.2.1 An EGT Model with a Desirable Estimation of Fitness

First, I extend the EGT model to provide each learner with a desirable estimate for the quality of actions (similarly to the one illustrated in Figure 3.2 for the joint space in Figure 3.1). Theorem 1 proves that this model is guaranteed to converge to the global optimum, if a unique such optimum exists. This result was only very briefly sketched in (Wiegand, 2004).

I precede the theorem with a short lemma.

Lemma 1. *Assume the populations for the EGT model are randomly initialized based on a uniform distribution over all possible initial populations. Then,*

$$\begin{aligned} P\left(\min_{i=1..n} x_i^{(0)} = 0\right) &= 0 \\ P\left(\max_{i=1..n} x_i^{(0)} = 1\right) &= 0 \\ P\left(\min_{j=1..m} y_j^{(0)} = 0\right) &= 0 \\ P\left(\max_{j=1..m} y_j^{(0)} = 1\right) &= 0 \end{aligned}$$

In other words, the probability that the initial populations contain extreme values (either 0 *or* 1*) for some proportion of genotypes is* 0*.*

Proof. One method to sample the simplex Δ^n uniformly is described in (Devroye, 1986) (pages 568 – 569): take $n-1$ uniformly distributed numbers in $[0,1]$, then sort them, and finally use the differences between consecutive numbers (also, the difference between the smallest number and 0, and the difference between 1 and the largest number) as the coordinates for the point. The probability that any of these differences is 0 or 1 is 0 (the probability of randomly generating a specific value, or a finite set of specific values at random using the uniform distribution over $[0,1]$ is 0). □

Theorem 1. *If payoff matrix A has a unique element $a_{i^\star j^\star}$ with maximal value, then the EGT model in Equations 3.9 – 3.12 will converge upon iteration to the $(i^\star, j^\star)$ Nash equilibrium with probability* 1*.*

$$u_i^{(t)} = \max_{j=1..m} a_{ij} \tag{3.9}$$

$$w_j^{(t)} = \max_{i=1..n} a_{ij} \tag{3.10}$$

$$x_i^{(t+1)} = \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u_i^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \leq u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k^{(t)} \right)^H \right) \tag{3.11}$$

$$y_j^{(t+1)} = \frac{y_j^{(t)}}{\sum_{k:w_k^{(t)}=w_j^{(t)}} y_k^{(t)}} \left(\left(\sum_{k:w_k^{(t)} \leq w_j^{(t)}} y_k^{(t)} \right)^H - \left(\sum_{k:w_k^{(t)} < w_j^{(t)}} y_k^{(t)} \right)^H \right) \tag{3.12}$$

Proof. I begin the proof with several observations. First, Equations 3.11 and 3.12 are identical to Equations 3.7 and 3.8: the new model changes only the mechanism for computing the fitness of an individual. Second, $u^{(t)}$ and $w^{(t)}$ are constant over time. Thus, the EGT model can be implemented by pre-computing $u^{(0)}$ and $w^{(0)}$, followed by only the iteration of Equations 3.11 and 3.12. Finally, the update of $x^{(t+1)}$ only depends on $x^{(t)}$ and on the constant vector $u^{(t)} = u^{(0)}$; as a consequence, the computation of $x^{(t+1)}$ can be iterated independently of the state of the other population (and similarly for $y^{(t+1)}$). This implies that the two populations are completely decoupled, and each of them evolves on its own as in a traditional evolutionary algorithm. I will prove that $x_{i^\star}^{(t)}$ converges to 1 upon iterating the model ($t \to \infty$). The proof that $\lim_{t\to\infty} y_{j^\star}^{(t)} = 1$ is similar, and it is omitted for brevity.

Given that $u_{i^\star}^{(t)} > u_i^{(t)}$ for all $i \neq i^\star$, it follows that $x_{i^\star}^{(t+1)} = 1 - \left(1 - x_{i^\star}^{(t)}\right)^H$, which is equivalent to $1 - x_{i^\star}^{(t+1)} = \left(1 - x_{i^\star}^{(t)}\right)^H$. In general, it is straightforward to show by induction that $1 - x_{i^\star}^{(t)} = \left(1 - x_{i^\star}^{(0)}\right)^{H^t}$.

Suppose the populations for the EGT model are initialized at random based on a uniform distribution over all possible initial populations. Lemma 1 indicates that the initial $x_{i^\star}^{(0)}$ is greater than 0 with probability 1. Thus, with probability 1, it follows that

$$\lim_{t\to\infty} x_{i^\star}^{(t)} = \lim_{t\to\infty} \left(1 - \left(1 - x_{i^\star}^{(0)}\right)^{H^t}\right) = 1 - \lim_{t\to\infty} \left(1 - x_{i^\star}^{(0)}\right)^{H^t} = 1$$

□

As a side note, it can be shown that the system converges to a mixed equilibrium if there are multiple global optima. This proof is useful to justify certain settings for the methods proposed in

Chapter 4. The next corollary establishes this result.

Corollary 1.1. *If the payoff matrix A has multiple elements with maximum value, then the EGT model in Equations 3.9 – 3.12 will converge upon iteration to a mixed equilibrium with probability* 1.

Proof. Let $u^\star = max_{ij} a_{ij}$, and let me define $z^{(t)} = \sum_{i:u_i^{(t)}=u^\star} x_i^{(t)}$. It follows that

$$
\begin{aligned}
1 - z^{(t+1)} &= 1 - \sum_{i:u_i^{(t)}=u^\star} x_i^{(t+1)} \\
&= 1 - \sum_{i:u_i^{(t)}=u^\star} \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u_i^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k^{(t)} \right)^H \right) \\
&= 1 - \sum_{i:u_i^{(t)}=u^\star} \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u^\star} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u^\star} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \right) \\
&= 1 - \sum_{i:u_i^{(t)}=u^\star} \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u^\star} x_k^{(t)}} \left(1 - \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \right) \\
&= 1 - \left(1 - \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \right) \sum_{i:u_i^{(t)}=u^\star} \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u^\star} x_k^{(t)}} \\
&= 1 - \left(1 - \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \right) \\
&= \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \\
&= \left(1 - z^{(t)} \right)^H \\
&\cdots \\
&= \left(1 - z^{(0)} \right)^{H^{t+1}}
\end{aligned}
$$

As before, consider that the populations for the EGT model are initialized at random based on a uniform distribution over all possible initial populations. It follows that $z^{(0)} < 1$ with probability 1 (from Lemma 1). Thus, $lim_{t\to\infty} z^{(t)} = 1$. The resulting equilibrium is mixed because for every i and j such that $u_i^{(t)} = u_i^{(0)} = u_j^{(t)} = u_j^{(0)} = u^\star$, the ratio $\frac{x_i^{(t)}}{x_j^{(t)}}$ is constant over time:

$$\begin{aligned}
\frac{x_i^{(t+1)}}{x_j^{(t+1)}} &= \frac{\frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u_i^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k \right)^H \right)}{\frac{x_j^{(t)}}{\sum_{k:u_k^{(t)}=u_j^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u_j^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_j^{(t)}} x_k^{(t)} \right)^H \right)} \\
&= \frac{\frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u^\star} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u^\star} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \right)}{\frac{x_j^{(t)}}{\sum_{k:u_k^{(t)}=u^\star} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u^\star} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u^\star} x_k^{(t)} \right)^H \right)} \\
&= \frac{x_i^{(t)}}{x_j^{(t)}}
\end{aligned}$$

□

Corollary 1.2. *The theoretical cooperative coevolutionary algorithm* Theoretical-CCEA *will converge to the global optimum with probability* 1*, if a unique global optimum exists.*

Theoretical-CCEA

Note: All populations are assumed infinite

For each individual i in the first population

Randomly select an infinite set $\{j_k\}_{k\ge1}$ *of collaborators from the second population*

$Fitness(i) = max_{k\ge1} a_{ij_k}$

For each individual j in the second population

Randomly select an infinite set $\{i_k\}_{k\ge1}$ *of collaborators from the first population*

$Fitness(j) = max_{k\ge1} a_{i_k j}$

Create next populations via tournament selection of size H

Proof. I show that the Theoretical-CCEA implements the EGT model in Equations 3.9 – 3.12. Observe that both of them employ infinite populations and tournament selection. As a consequence, it suffices to show that the Theoretical-CCEA and the EGT model assign the same fitness to the individuals (this implies that both of them will create identical populations at the next generation, were they applied to the same current populations).

Indeed, the fitnesses are identical with the Theoretical-CCEA and the EGT model, as shown next. Given an individual with genotype i in the first population, its expected fitness with the theoretical EGT model is equal to $max_{j=1..m}a_{ij}$. The fitness of an individual with genotype i in the Theoretical-CCEA is $max_{k\geq 1}a_{ij_k}$. Let $j^\star \in \{1,...,m\}$ be the index such that $a_{ij^\star} = max_{j=1..m}a_{ij}$. From Lemma 1, the proportion of $j^\star$ is non-zero with probability 1 in the initial random population. The use of tournament selection also implies that the proportion of $j^\star$ is non-zero thereafter (it might be very small, but it is not zero). As a consequence, the infinite set of random collaborators $\{j_k\}_{k\geq 1}$ contains $j^\star$ with probability 1. Thus, $Fitness(i) = max_{j=1..m}a_{ij}$. Similarly, the individuals in the second population are assigned identical fitness under both models. □

As an observation, the simple EGT framework in Equations 3.5–3.8 assumes that an individual's fitness is the mean payoff over pairwise collaborations with *every* member of the cooperating population. This has two possible interpretations. First, the framework might approximate (as the population size approaches infinity) a real CCEA where the fitness of an individual is computed as the average payoff over pairwise collaborations with each and every individual in the other population. Under this interpretation, the Theoretical-CCEA improves upon the EGT model in Equations 3.5 – 3.8 by providing each learner with a better quality estimate at the same computational cost (Figures 3.3 (a) and (c)). Corollary 1.2 establishes that this improved estimate allows the learners to converge to the global optimum with probability 1.

A second interpretation of the complete mixing assumption is that it is computing an individual's fitness as the expected payoff with a single random collaborator from the other population. From this perspective, the Theoretical-CCEA is a generalization of the EGT framework in Equations 3.5–3.8 that involves an infinite number of random collaborators. In the next section, I provide a novel EGT model that bridges these two extremes by employing an arbitrary finite number of collaborators.

3.2.2 An EGT Model with a Practical Estimation of Fitness

The primary problem with the Theoretical-CCEA is that it requires an infinite number of collaborators. One class of approximations for this model involves only a finite number of random collaborators. Although this idea is not new in practical approaches to cooperative coevolutionary algorithms, its theoretical study has been very limited. In this section, I extend the EGT model in Equations 3.5 – 3.8 to account for multiple collaborators, and I analyze the theoretical convergence guarantees and limitations for this new model.

First, I estimate the expected fitness for an individual if computed as the maximum reward obtained for that individual when in combination with multiple collaborators from the other population. The following theorem establishes this result.

Theorem 2. *Let a_{ij} be the payoff for individual i when teamed with an individual j, and $(p_j)_{j\in 1..n}$ be the probability distribution for the individuals in the population of collaborators for i (here, n is the number of distinct genotypes in the population of collaborators). The expected maximum payoff for i over N pairwise combinations with random collaborators $j_1 \ldots j_N$ chosen with replacement from the other population is*

$$\sum_{j=1}^{n} a_{ij} \frac{p_j}{\sum_{k:a_{ik}=a_{ij}} p_k} \left(\left(\sum_{k:a_{ik}\le a_{ij}} p_k \right)^N - \left(\sum_{k:a_{ik}< a_{ij}} p_k \right)^N \right).$$

Proof. Clearly, the expected maximum payoff of *i* over *N* pairwise combinations can be expressed as a weighted sum of all possible payoffs a_{ij} that *i* can receive with different types of collaborators. The weight of each term a_{ij} equals the probability h_j that a_{ij} is the maximum payoff observed over *N* trials. This computation can also be modeled as follows: given an infinite population with the distribution of individuals indicated by $x_j = p_j$ and with fitness $f_j = a_{ij}$, what is an estimation h_j for the maximum fitness of *N* randomly selected individuals (with replacement)? Interestingly, this is the same problem as that of modeling tournament selection in the EGT model. As described in Equations 3.7 and 3.8, it follows that

$$h_j = \frac{x_j}{\sum_{k:f_k=f_j} x_k} \left(\left(\sum_{k:f_k\le f_j} x_k \right)^N - \left(\sum_{k:f_k< f_j} x_k \right)^N \right)$$

The remainder of the proof is only a matter of changing notations. □

Given this result, I extend the traditional EGT model in Equations 3.5–3.8 to replace complete mixing with the assessment of fitness based on the maximum of rewards with multiple collaborators.

Theorem 3. *The EGT model in Equations 3.13 – 3.16 represents a theoretical model for a cooperative coevolutionary algorithm where the fitness of an individual is computed as the maximum reward with N collaborators.*

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} \frac{y_j^{(t)}}{\sum_{k:a_{ik}=a_{ij}} y_k^{(t)}} \left(\left(\sum_{k:a_{ik} \leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik} < a_{ij}} y_k^{(t)} \right)^N \right) \tag{3.13}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} \frac{x_i^{(t)}}{\sum_{k:a_{kj}=a_{ij}} x_k^{(t)}} \left(\left(\sum_{k:a_{kj} \leq a_{ij}} x_k^{(t)} \right)^N - \left(\sum_{k:a_{kj} < a_{ij}} x_k^{(t)} \right)^N \right) \tag{3.14}$$

$$x_i^{(t+1)} = \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u_i^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \leq u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k^{(t)} \right)^H \right) \tag{3.15}$$

$$y_j^{(t+1)} = \frac{y_j^{(t)}}{\sum_{k:w_k^{(t)}=w_j^{(t)}} y_k^{(t)}} \left(\left(\sum_{k:w_k^{(t)} \leq w_j^{(t)}} y_k^{(t)} \right)^H - \left(\sum_{k:w_k^{(t)} < w_j^{(t)}} y_k^{(t)} \right)^H \right) \tag{3.16}$$

Proof. Equations 3.15–3.16 preserve the tournament selection procedure of the standard EGT model (Equations 3.7–3.8). The fitness of each individual is estimated as the expected maximum reward with N random collaborators from the other population (as established in Theorem 2). □

Note that this new model is equivalent to the previous EGT model in Equations 3.5–3.8 when a single collaborator is used ($N = 1$). More importantly, the new EGT model approximates the Theoretical-CCEA as N approaches ∞. It would therefore be expected that the new EGT model is more likely to converge to the global optimum as N increases. The proof that that is the case relies on the following lemmas, which establish bounds on the probabilities that the initial populations have some extremely large or small proportions of certain genotypes.

Lemma 2. *Assume the populations for the EGT model are initialized at random based on a uniform distribution over all possible initial populations. Then, for any $\varepsilon > 0$, there exists $\theta_\varepsilon > 0$ such that*

$$P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_\varepsilon \right) < \varepsilon \tag{3.17}$$

$$P\left(\max_{i=1..n} x_i^{(0)} \geq 1 - \theta_\varepsilon \right) < \varepsilon \tag{3.18}$$

$$P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_\varepsilon \right) < \varepsilon \tag{3.19}$$

$$P\left(\max_{j=1..m} y_j^{(0)} \geq 1 - \theta_\varepsilon \right) < \varepsilon \tag{3.20}$$

Proof. As described in Lemma 1, $\left(x_i^{(0)}\right)_{i=1..n}$ can be generated as the difference between $n-1$ numbers generated uniformly in $[0,1]$. It follows that $\min_{i=1..n} x_i^{(0)}$ is the closest distance between two such numbers (and possibly the boundaries 0 and 1).

Suppose $\gamma > 0$ is a small number. I iterate over the $n-1$ uniformly-distributed random numbers that are needed to generate an initial population $\left(x_i^{(0)}\right)_{i=1..n}$. The probability that the first number is not within γ of the boundaries 0 and 1 is $1-2\gamma$. The probability that the second number is not within γ of the boundaries or of the first number is less than or equal to $1-4\gamma$. In general, the probability that the kth number is not within γ of the boundaries or of the first $k-1$ numbers is less than or equal to $1-2k\gamma$. Given that the numbers are generated independently of one another, the probability that the closest pair of points (considering the boundaries) is farther apart than γ is equal to

$$\begin{aligned} P\left(\min_{i=1..n} x_i^{(0)} \leq \gamma\right) &= 1 - P\left(\min_{i=1..n} x_i^{(0)} > \gamma\right) \\ &= 1 - \prod_{i=1}^{n-1} (1-2i\gamma) \quad \leq \quad 1-(1-2(n-1)\gamma)^{n-1} \end{aligned}$$

Inequalities 3.17 and 3.19 are symmetric, and as such, the proof that

$$P\left(\min_{j=1..m} y_j^{(0)} \leq \gamma\right) \leq 1-(1-2(m-1)\gamma)^{m-1}$$

is very similar. Given that

$$\begin{aligned} \lim_{\gamma\to 0} 1-(1-2(n-1)\gamma)^{n-1} &= 0 \\ \lim_{\gamma\to 0} 1-(1-2(m-1)\gamma)^{m-1} &= 0 \end{aligned}$$

it follows that for any $\varepsilon > 0$ there exists $\theta_\varepsilon > 0$ such that $P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_\varepsilon\right) < \varepsilon$ and $P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_\varepsilon\right) < \varepsilon$.

To prove Inequality 3.18, consider that $\max_{i=1..n} x_i^{(0)} \geq 1-\theta_\varepsilon$ implies that all other x_i ratios except for the maximum are smaller to θ_ε, which, as proven above, occurs with probability smaller than ε. The proof for Inequality 3.20 is similar. □

Lemma 3. *Assume the populations for the EGT model are initialized at random based on a uniform distribution over all possible initial populations. Then, for any $\varepsilon > 0$, there exists $\eta_\varepsilon > 0$ such that*

$$P\left(\min_{i=1..n} x_i^{(0)} > \eta_\varepsilon \wedge \max_{i=1..n} x_i^{(0)} < 1-\eta_\varepsilon \wedge \min_{j=1..m} y_j^{(0)} > \eta_\varepsilon \wedge \max_{j=1..m} y_j^{(0)} < 1-\eta_\varepsilon\right) \geq 1-\varepsilon$$

In other words, there is an arbitrarily probability that the initial populations contain reasonable values (not too close to either 0 *or* 1*) for all proportions of genotypes.*

Proof. I apply Lemma 2 for $\frac{1-\sqrt{1-\varepsilon}}{2}$, which is greater than 0. The specific value of η_ε for this proof equals the value of $\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}$ from Lemma 2. It follows that:

$$\begin{aligned}
&P\left(\min_{i=1..n} x_i^{(0)} > \eta_\varepsilon \wedge \max_{i=1..n} x_i^{(0)} < 1-\eta_\varepsilon \wedge \min_{j=1..m} y_j^{(0)} > \eta_\varepsilon \wedge \max_{j=1..m} y_j^{(0)} < 1-\eta_\varepsilon\right) \\
&\quad = P\left(\min_{i=1..n} x_i^{(0)} > \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \wedge \max_{i=1..n} x_i^{(0)} < 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right) \times \\
&\qquad P\left(\min_{j=1..m} y_j^{(0)} > \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \wedge \max_{j=1..m} y_j^{(0)} < 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right) \\
&\quad = \left(1-P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \vee \max_{i=1..n} x_i^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right) \times \\
&\qquad \left(1-P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \vee \max_{j=1..m} y_j^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right) \\
&\quad \geq \left(1-\left(P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)+P\left(\max_{i=1..n} x_i^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right)\right) \times \\
&\qquad \left(1-\left(P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)+P\left(\max_{j=1..m} y_j^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right)\right) \\
&\quad \geq \left(1-2\frac{1-\sqrt{1-\varepsilon}}{2}\right) \times \left(1-2\frac{1-\sqrt{1-\varepsilon}}{2}\right) \\
&\quad = 1-\varepsilon
\end{aligned}$$

□

Next, I use the new EGT model described by Equations 3.13 – 3.16 to analyze the properties of cooperative coevolutionary algorithms. Theorem 4 shows that properly implemented CCEAs *will converge to the global optimum* in any problem domain, if given *enough* resources. The term "enough" indicates two types of resources that CCEAs need: large populations to explore the space (here, the populations are in fact assumed infinite for simplicity), and large numbers of collaborators to provide accurate fitness estimates for all populations.

Theorem 4. *Given a joint reward matrix A with a unique global optimum $a_{i^\star j^\star}$, for any $\varepsilon > 0$ and any $H \geq 2$, there exists a value $N_\varepsilon \geq 1$ such that the theoretical CCEA model in Equations 3.13–3.16 converges to the global optimum with probability greater than $(1-\varepsilon)$ for any number of collaborators N such that $N \geq N_\varepsilon$.*

Proof. I only use ε as a guarantee for the worst case scenario for the proportions of individuals in the initial populations. From Lemma 3, it follows that there exists $\eta_\varepsilon > 0$ such that with probability at least $1-\varepsilon$, it holds that $\eta_\varepsilon < x_i^{(0)} < 1-\eta_\varepsilon$ and $\eta_\varepsilon < y_j^{(0)} < 1-\eta_\varepsilon$ for $i, j \in \{1,2,3\}$. In other words, with probability ε, the initial populations will not have any proportion of individuals that cover more than $1-\eta_\varepsilon$, nor cover less than η_ε of the entire population.

I will prove that there exists $N_\varepsilon \geq 0$ such that the EGT model converges to the global optimum for any $N \geq N_\varepsilon$ and for all initial populations that satisfy $\eta_\varepsilon < x_{i^\star}^{(0)} < 1-\eta_\varepsilon$ and $\eta_\varepsilon < y_{j^\star}^{(0)} < 1-\eta_\varepsilon$. To this end, let α be the second highest element of the joint payoff matrix A ($\alpha < a_{i^\star j^\star}$). It follows that $u_i^{(t)} \leq \alpha$ for all $i \neq i^\star$, and similarly $w_j^{(t)} \leq \alpha$ for all $j \neq j^\star$. Given the symmetry, I only prove the first (by refining Equation 3.13):

$$\begin{aligned} u_i^{(t)} &\leq \sum_{j=1}^{m} \alpha \frac{y_j^{(t)}}{\sum_{k:a_{ik}=a_{ij}} y_k^{(t)}} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} y_k^{(t)} \right)^N \right) \\ &\leq \alpha \sum_{j=1}^{m} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} y_k^{(t)} \right)^N \right) \leq \alpha \end{aligned}$$

Next, I work on identifying a lower bound for $u_{i^\star}^{(t)}$:

$$
\begin{aligned}
u_{i^\star}^{(t)} &= \sum_{j=1}^{m} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&= a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&= a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star \wedge a_{i^\star j}\ge 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N
\end{aligned}
$$

Given that $\sum_{k=1}^{m} y_k^{(t)} = 1$, I further refine the previous inequality:

$$
\begin{aligned}
u_{i^\star}^{(t)} &\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(1 - y_{j^\star}^{(t)} \right)^N \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \left(1 - y_{j^\star}^{(t)} \right)^N \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \\
&= a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t)} \right)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \right) \qquad (3.21)
\end{aligned}
$$

The inequalities $\eta_\varepsilon < y_{j^\star}^{(0)} < 1-\eta_\varepsilon$ hold for the initial populations, as inferred earlier from Lemma 3. It follows from Equation 3.21 that

$$u_{i^\star}^{(0)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \tag{3.22}$$

However,

$$\lim_{N\to\infty} a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) = a_{i^\star j^\star} \tag{3.23}$$

Given that $a_{i^\star j^\star} > \alpha$, Equation 3.23 implies that there exists $N_1 \geq 1$ such that

$$a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) > \alpha \tag{3.24}$$

for all $N \geq N_1$. From Equations 3.21 and 3.24, it follows that $u_{i^\star}^{(0)} > \alpha$ for all $N \geq N_1$. Observe that N_1 does not depend on the initial populations $x^{(0)}$ and $y^{(0)}$. Similarly, it is straightforward to prove that

$$w_{j^\star}^{(t)} \geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{i j^\star} < 0} a_{i j^\star} \right) \tag{3.25}$$

$$w_{j^\star}^{(0)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{i j^\star} < 0} a_{i j^\star} \right) \tag{3.26}$$

and that there exists $N_2 \geq 1$ such that $w_{j^\star} > \alpha$ for all $N \geq N_2$.

Let $N_\varepsilon = max\,(N_1, N_2)$, and let $N \geq N_\varepsilon$. Next, I show by induction by t (the number of iterations of the model, i.e. the number of generations) that the following four inequalities hold:

$$
\begin{aligned}
u_{i^\star}^{(t)} &\geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\
w_{j^\star}^{(t)} &\geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{i j^\star} < 0} a_{i j^\star} \right) \\
x_{i^\star}^{(t+1)} &\geq x_{i^\star}^{(t)} \\
y_{j^\star}^{(t+1)} &\geq y_{j^\star}^{(t)}
\end{aligned}
$$

At the first generation ($t = 0$), the first two inequalities hold (Equations 3.22 and 3.26). Combining these with the definition of N, it follows that $u_{i^\star}^{(0)} > u_i^{(0)}$ for all $i \neq i^\star$ (and similarly, $w_{j^\star}^{(0)} > w_j^{(0)}$ for all $j \neq j^\star$). As a consequence, $x_{i^\star}^{(1)} = 1 - \left(1 - x_{i^\star}^{(0)}\right)^H > x_{i^\star}^{(0)}$ (from Equation 3.13), and similarly $y_{j^\star}^{(1)} = 1 - \left(1 - y_{j^\star}^{(0)}\right)^H > y_{j^\star}^{(0)}$ (from Equation 3.14).

To prove the inductive step, it follows from Equation 3.21 and from the inductive hypothesis that

$$
\begin{aligned}
u_{i^\star}^{(t+1)} &\geq a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t+1)}\right)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\
&\geq a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\
&\cdots \\
&\geq a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(0)}\right)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\
&\geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right)
\end{aligned}
$$

Given the definitions of N and α, this also implies that $u_{i^\star}^{(t+1)} > \alpha > u_i^{(t+1)}$ for all $i \neq i^\star$. As a consequence, $x_{i^\star}^{(t+1)} = 1 - \left(1 - x_{i^\star}^{(t)}\right)^H \geq x_{i^\star}^{(t)}$ (from Equation 3.13). Similarly, Equation 3.25 and

the inductive hypothesis imply that

$$\begin{aligned} w_{j^\star}^{(t+1)} &\geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t+1)}\right)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{ij^\star} < 0} a_{ij^\star}\right) \\ &\geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{ij^\star} < 0} a_{ij^\star}\right) \\ &\cdots \\ &\geq a_{i^\star j^\star} - (1 - \eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{ij^\star} < 0} a_{ij^\star}\right) \end{aligned}$$

Again, given the definition of N and α, this implies $w_{j^\star}^{(t+1)} > \alpha > w_j^{(t+1)}$ for all $j \neq j^\star$. As a consequence, $y_{j^\star}^{(t+1)} = 1 - \left(1 - y_{j^\star}^{(t)}\right)^H \geq y_{j^\star}^{(t)}$ (from Equation 3.14).

Having shown that both $x_{i^\star}^{(t)}$ and $y_{j^\star}^{(t)}$ are monotonically increasing, and given that they are bounded between 0 and 1, it follows that they converge to some value. Given that $u_{i^\star}^{(t)} > u_i^{(t)}$ for all $i \neq i^\star$ at each iteration, it follows that $x_{i^\star}^{(t+1)} = 1 - \left(1 - x_{i^\star}^{(t)}\right)^H$ at each iteration as well. If $\bar{x}$ is the limit of the $x_{i^\star}^{(t)}$ values when t goes to ∞, then $\bar{x} = 1 - (1 - \bar{x})^H$, which implies that $\bar{x}$ is either 0 or 1. I can rule out the 0 limit because the values of $x_{i^\star}^{(t)}$ are monotonically increasing and $x_{i^\star}^{(0)} > \eta_\varepsilon$. Thus, $x_{i^\star}^{(t)}$ converges to 1. The proof that $y_{j^\star}^{(t)}$ also converges to 1 is similar and is omitted for brevity. □

Theorem 4 indicates that CCEAs should converge to the global optimum if properly set (large population sizes and large numbers of collaborators). However, one may ask what are their limitations in practice. I claim there are two types of limitations that might affect the performance of CCEAs. First, populations will only have finite numbers of individuals in practical applications. In this case, the limitations of CCEAs will be primarily due to insufficient exploration in deceptive domains. Given that these limitations are similar to local convergence problems in standard EAs, I rather focus on a second type of limitations, namely those due to poor fitness estimates resulting from insufficient collaborators. The next theorem establishes that no finite number of collaborators is sufficient for CCEAs to have good performance across all possible problem domains. Rather, for any number of collaborators, there exists problem domains that can make CCEAs converge suboptimally in most situations.

Theorem 5. *For any $\varepsilon > 0$, any $N > 1$ and any $H \geq 2$, there exists a joint reward matrix A with a unique global maximum such that the theoretical CCEA model in Equations 3.13–3.16 converges to a globally optimal equilibrium with probability lower than ε.*

Proof. Suppose $\varepsilon > 0$, $N > 1$, and $H \geq 2$. Consider the joint reward matrix defined by

$$A: \begin{bmatrix} 2 & -\rho & 0 \\ -\rho & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\rho > 0$. Here, the optimal joint action $(1,1)$ receives a reward of 2, but there may be severe penalties associated with miscoordination for joint actions $(1,2)$ and $(2,1)$. The proof proceeds as follows. First, I identify threshold values for x_3 and y_3 such that, once their values exceed those thresholds, the system is pulled toward the suboptimal equilibrium $(3,3)$. Second, I make the miscoordination penalty ρ severe enough as to move the system towards the suboptimal equilibrium $(3,3)$ until both x_3 and y_3 exceed the thresholds identified earlier.

As before, I use ε to provide some guarantees on the distribution of the initial populations. Arbitrarily small values of ε are especially interesting: I want to show that, for some fixed values of N and H, there are coordination games for which the theoretical CCEA model converges to the suboptimal solution with probability 99%. For the same fixed values of N and H, there are other more difficult coordination games for which the model converges to the suboptimal solution with probability 99.999%. And so on. Thus, I will show that the CCEA model converges to the suboptimal solution $(3,3)$ with probability higher than $1-\varepsilon$, as $\varepsilon \to 0$. Lemma 3 indicates that for any $\varepsilon > 0$ there exists $\eta_\varepsilon > 0$ such that the initial populations will not have any proportion of individuals cover more than $1-\eta_\varepsilon$, nor cover less than η_ε of the entire population, with probability higher than $1-\varepsilon$.

I start by expressing the expected fitnesses of individuals from one of the populations, given the game defined earlier. As before, $x_i^{(t)}$ and $y_j^{(t)}$ denote the values of x_i and y_j after t iterations. Using Equations 3.13–3.14, it follows that

$$
\begin{aligned}
u_1^{(t)} &= 2\left(\left(y_1^{(t)}+y_2^{(t)}+y_3^{(t)}\right)^N-\left(y_2^{(t)}+y_3^{(t)}\right)^N\right)-\rho\left({y^{(t)}}_2^N-0^N\right) \\
&= 2\left(1-\left(1-y_1^{(t)}\right)^N\right)-\rho{y^{(t)}}_2^N \qquad (3.27) \\
u_2^{(t)} &= -\rho{y^{(t)}}_1^N \qquad (3.28) \\
u_3^{(t)} &= 1\left(\left(y_1^{(t)}+y_2^{(t)}+y_3^{(t)}\right)^N-\left(y_1^{(t)}+y_2^{(t)}\right)^N\right) \\
&= 1-\left(1-y_3^{(t)}\right)^N \qquad (3.29)
\end{aligned}
$$

Obviously, $u_3^{(t)} > 0 > u_2^{(t)}$ at each generation t. Given that $lim_{y_3^{(t)}\to 1} u_3^{(t)} = 1$ and $lim_{y_3^{(t)}\to 1} u_1^{(t)} = 0$, it follows that there exists a $0 < \bar{y} < 1$ such that $u_3^{(t)} > u_1^{(t)}$ for all $y_3^{(t)} > \bar{y}$. In other words, there exists $0 < \bar{y} < 1$ such that $u_3^{(t)} > \max\left(u_1^{(t)}, u_2^{(t)}\right)$ for all $y_3^{(t)} > \bar{y}$. Similarly, there exists $0 < \bar{x} < 1$ such that $w_3^{(t)} > \max\left(w_1^{(t)}, w_2^{(t)}\right)$ for all $x_3^{(t)} > \bar{x}$. Once the EGT model reaches a configuration with $x_3^{(t)} > \bar{x}$ and $y_3^{(t)} > \bar{y}$, the values of $x_3^{(t')}$ and $y_3^{(t')}$ will increase monotonically at each subsequent generation $t' > t$. This is the case because $u_3^{(t')}$ and $w_3^{(t')}$ (which are monotonically increasing as functions of $y_3^{(t')}$ and $x_3^{(t')}$, respectively) will always be greater than both $u_1^{(t')}$ and $u_2^{(t')}$, and $w_1^{(t')}$ and $w_2^{(t')}$, respectively. As a consequence, the EGT model will converge to the suboptimal equilibrium $(3,3)$ once it reaches a configuration with $x_3^{(t)} > \bar{x}$ and $y_3^{(t)} > \bar{y}$.

The main idea of the proof is to set ρ arbitrarily large such that $u_3^{(t)} > \max\left(u_1^{(t)}, u_2^{(t)}\right)$ and $w_3^{(t)} > \max\left(w_1^{(t)}, w_2^{(t)}\right)$ until at least $x_3^{(t)} > \bar{x}$ and $y_3^{(t)} > \bar{y}$. In order to set ρ, I analyze the worst case scenario for the number of generations that $x_3^{(t)}$ and $y_3^{(t)}$ would require to exceed the threshold. As shown in Theorem 3, it holds that

$$x_3^{(t)} = 1-\left(1-x_3^{(0)}\right)^{H^t}$$

I further refine this equation to identify a lower bound on the number of iterations t such that $x_3^{(t)}$ exceeds the bound $\bar{x}$ after those t iterations:

$$x_3^{(t)} > \bar{x} \quad \Leftrightarrow \quad 1-\left(1-x_3^{(0)}\right)^{H^t} > \bar{x} \Leftrightarrow \left(1-x_3^{(0)}\right)^{H^t} < 1-\bar{x}$$

$$\Leftrightarrow \quad H^t \ln\left(1-x_3^{(0)}\right) < \ln\left(1-\bar{x}\right) \Leftrightarrow H^t > \frac{\ln\left(1-\bar{x}\right)}{\ln\left(1-x_3^{(0)}\right)}$$

$$\Leftrightarrow t \quad \ln H > \ln \frac{\ln\left(1-\bar{x}\right)}{\ln\left(1-x_3^{(0)}\right)} \Leftrightarrow t > \frac{\ln \frac{\ln(1-\bar{x})}{\ln\left(1-x_3^{(0)}\right)}}{\ln H}$$

Given that $\eta_\varepsilon < x_3^{(0)} < 1-\eta_\varepsilon$, it follows that a worst-case number of iterations necessary for $x_3^{(0)}$ to surpass $\bar{x}$ is t_1 such that

$$t_1 > \frac{\ln \frac{\ln(1-\bar{x})}{\ln \eta_\varepsilon}}{\ln H} > \frac{\ln \frac{\ln(1-\bar{x})}{\ln\left(1-x_3^{(0)}\right)}}{\ln H}$$

The worst-case number of iterations necessary for $y_3^{(0)}$ to surpass $\bar{y}$ can be computed similarly, and it is t_2 such that

$$t_2 > \frac{\ln \frac{\ln(1-\bar{y})}{\ln \eta_\varepsilon}}{\ln H} > \frac{\ln \frac{\ln(1-\bar{y})}{\ln\left(1-y_3^{(0)}\right)}}{\ln H}$$

Thus, in the worst-case, it takes the system at most $\bar{t} = \max(t_1, t_2)$ iterations such that both $x_3^{(\bar{t})}$ and $y_3^{(\bar{t})}$ exceed their thresholds $\bar{x}$ and $\bar{y}$, respectively. Note that $\bar{t}$ does not depend on the initial populations, but rather only on H and ε (through η_ε).

Next, I identify a value for ρ such that $u_3^{(t)} > u_1^{(t)}$ and $w_3^{(t)} > w_1^{(t)}$ during the first $\bar{t}$ iterations. Consider the conditions under which $u_3^{(t)} > u_1^{(t)}$ (the $w_3^{(t)} > w_1^{(t)}$ case is similar). Observe that $u_3^{(t)} = 1-\left(1-y_3^{(t)}\right)^N$. Given that $y_3^{(t)}$ will be monotonically increasing, and given that its initial value is greater than η_ε, it follows that $u_3^{(t)} > 1-(1-\eta_\varepsilon)^N$. I want $1-(1-\eta_\varepsilon)^N > u_1^{(t)}$ during the first $\bar{t}$ iterations, which would be enough for $x_3^{(t)}$ to surpass $\bar{x}$. Given that $u_1^{(t)}$ increases when $y_1^{(t)}$ increases, and it decreases when $y_2^{(t)}$ increases, the worst case scenario is when $u_3^{(t)} > u_1^{(t)} > u_2^{(t)}$ and $w_3^{(t)} > w_1^{(t)} > w_2^{(t)}$ for the first $\bar{t}$ iterations. In this case, tournament selection will lead to

$$\begin{aligned} x_2^{(t+1)} &= \left(x_2^{(t)}\right)^H \\ y_2^{(t+1)} &= \left(y_2^{(t)}\right)^H \end{aligned}$$

It follows from Equation 3.27 that

$$\begin{aligned} u_1^{(t)} &= 2\left(1-\left(1-y_1^{(t)}\right)^N\right) - \rho {y_2^{(t)}}^N \\ &\leq 2 - \rho {y_2^{(t)}}^N \end{aligned} \tag{3.30}$$

The worst-case scenario for the proof is when $w_2^{(t)} < w_1^{(t)}$ and $w_2^{(t)} < w_3^{(t)}$ for $t \leq \bar{t}$. This is the worst case because it implies that $y_2^{(t)}$ will decrease fastest, leading to an exponential decrease in the impact of miscoordination penalties onto $u_1^{(t)}$. I need to choose a value of ρ that is high enough to compensate for a potential exponential decrease in the value of $y_2^{(t)}$. After $\bar{t}$ iterations, the value of $y_2^{(\bar{t})}$ is greater or equal to $\left(y_2^{(0)}\right)^{H^{\bar{t}}}$, which is the value $y_2^{(\bar{t})}$ would have if $w_2^{(t)} < w_1^{(t)}$ and $w_2^{(t)} < w_3^{(t)}$ during all these $\bar{t}$ iterations. If follows from Equation 3.30 that

$$\begin{aligned} u_1^{(\bar{t})} &\leq 2 - \rho {y_2^{(\bar{t})}}^N \\ &\leq 2 - \rho \left(\left(y_2^{(0)}\right)^{H^{\bar{t}}}\right)^N \\ &= 2 - \rho \left(y_2^{(0)}\right)^{NH^{\bar{t}}} \end{aligned} \tag{3.31}$$

Now, remember that ε was used to obtain some bounds on the initial populations. One such set of bounds indicated that $\eta_\varepsilon < y_2^{(0)} < 1-\eta_\varepsilon$. Using this condition in combination with Equation 3.31, it follows that

$$\begin{aligned} u_1 &\leq 2 - \rho \left(y_2^{(0)}\right)^{NH^{\bar{t}}} \\ &\leq 2 - \rho {\eta_\varepsilon}^{NH^{\bar{t}}} \end{aligned}$$

I set $\rho > \frac{2}{\eta_\varepsilon^{NH^t}}$, which implies $u_1^{(t)} < 0 < u_3^{(t)}$ during the first $\bar{t}$ iterations of the system. The proof that $w_3^{(t)} > 0 > w_1^{(t)}$ for $\rho > \frac{2}{\eta_\varepsilon^{NH^t}}$ follows similarly (observe that $u_1^{(t)}$, $u_2^{(t)}$, and $u_3^{(t)}$ as functions of $y_1^{(t)}$, $y_2^{(t)}$, and $y_3^{(t)}$ have the same form as $w_1^{(t)}$, $w_2^{(t)}$, and $w_3^{(t)}$ have as functions of $x_1^{(t)}$, $x_2^{(t)}$, and $x_3^{(t)}$). □

3.2.3 Basins of Attraction due to Estimations

Ideally, a properly implemented cooperative coevolutionary algorithm will find the global optimum (Theorem 4) if given enough resources. However, Theorem 5 indicates that insufficient resources (in particular, using only a few random collaborators) might lead to poor estimates that could result in drift towards suboptimal solutions. In this section, I describe an intuitive way to visualize the impact of improved estimates onto the performance of the system.

Given specific initial populations, the EGT model is completely deterministic. As shown in (Wiegand, 2004), the populations are expected to converge to Nash equilibrium points in the payoff matrix. As a result, it is straightforward to provide two-dimmensional visualizations of the basins of attraction for different Nash equilibria when each population contains only two genotypes: the two-dimmensional location of a pixel encodes uniquely the initial ratios of one of the genotypes for each population, and the color of the pixel encodes to which equilibrium the system converges upon iteration (see (Panait *et al.*, 2004) for details). However, such simple problem domains have severe limitations with respect to the range of challenges they might present to cooperative coevolutionary algorithms.

Here, I choose instead to illustrate the basins of attraction of Nash equilibria in 3*x*3 coordination games, which can stress the challenges posed by poor estimates of fitness. To this end, I employ two multiagent coordination problems: Climb and Penalty. Climb has two Nash equilibria, the optimum $(1,1)$ and the suboptimum $(2,2)$, and it is strongly deceptive. Penalty has three Nash equilibria: $(1,1)$, $(2,2)$, and $(3,3)$. Of them, $(2,2)$ is suboptimal, but is more forgiving if one or the other population deviates from the Nash equilibrium. The problem domains are similar to the ones introduced in (Claus and Boutilier, 1998) and used in previous investigations of multiagent Q-learning (Kapetanakis and Kudenko, 2002b; Lauer and Riedmiller, 2000) and of cooperative coevolution (Panait *et al.*, 2003). The joint payoff matrices for these coordination games are:

$$Climb: \begin{bmatrix} 11 & -\sigma & 0 \\ -\sigma & 7 & 6 \\ 0 & 0 & 5 \end{bmatrix} \qquad Penalty: \begin{bmatrix} 10 & 0 & -\sigma \\ 0 & 2 & 0 \\ -\sigma & 0 & 10 \end{bmatrix}$$

where σ is a penalty associated with miscoordinations. Unless stated otherwise, $\sigma = 10$.

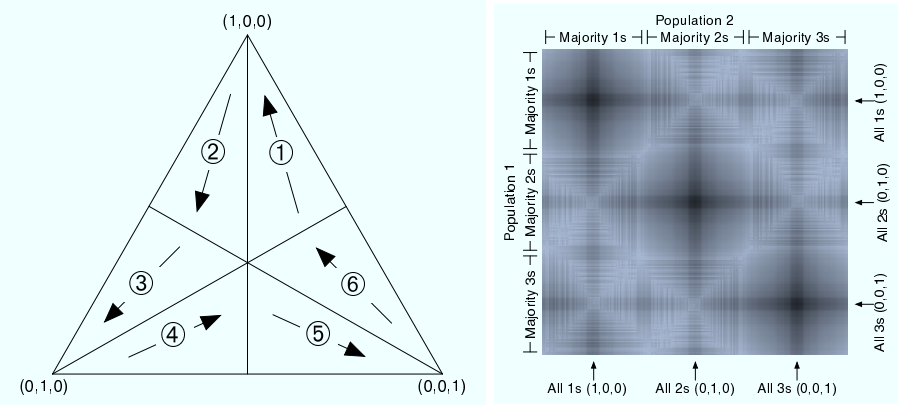

Figure 3.4: The projection of $\Delta^3 \times \Delta^3$ to $[0,1]^2$. (left): The projection divides the simplex Δ^3 into six equal-area triangles; arrows shows the direction for sorting points in each area. (right): Visualization of the cartesian product of two simplexes.

I apply the EGT model in Equations 3.13 – 3.16 to each of these two problem domains, and I iterate it for 100000 generations or until the proportion of one action in each population exceeds a threshold of $1-10^{-10}$. Given that tournament selection is a very strong selection method, approximation errors due to the fixed-sized computer representation of real-valued numbers may result to erroneous convergence to arbitrary solutions in certain situation, even when the tournament size is set to only 2. To remedy this situation, I employ a tournament selection size $H = 1.5$, estimated as half-way between random search and tournament selection with size 2 (similar to the implementation of tournament selection in the ECJ system (Luke, 2005)). I identify each of the possible end results (Nash equilibria) with a color. For consistency, black dots always indicate convergence to a suboptimal solution, while white and grey dots indicate converge to global optima. The projection of the $\Delta^3 \times \Delta^3$ is detailed next.

The search space (Δ^3) of the first population is projected along the vertical axis, while that of the second population is projected along the horizontal axis. The projection of Δ^3 to one dimension starts by dividing it into six equal-area triangles, as in Figure 3.4 (left). Initial populations in areas $1-2$ have a majority of 1s in the population, and similarly areas $3-4$ and $5-6$ have majorities of 2s and 3s. If $i < j$, all initial populations in area i are projected before those in area j. Inside each area, initial populations are ordered lexicographically in the direction of the arrow. More specifically, in regions $1-2$, the sorting is done primarily on p_1, and secondarily on p_2; for $3-4$, p_2 and p_3; for $5-6$, p_3 and p_1. Even-numbered regions are sorted ascending and odd-numbered regions are sorted descending. The objective of all these procedures is to group together regions of the space of initial populations that are expected to converge to the same Nash equilibrium. I sample 216 initial populations in the simplex: the six areas in Figure 3.4(left) are each divided

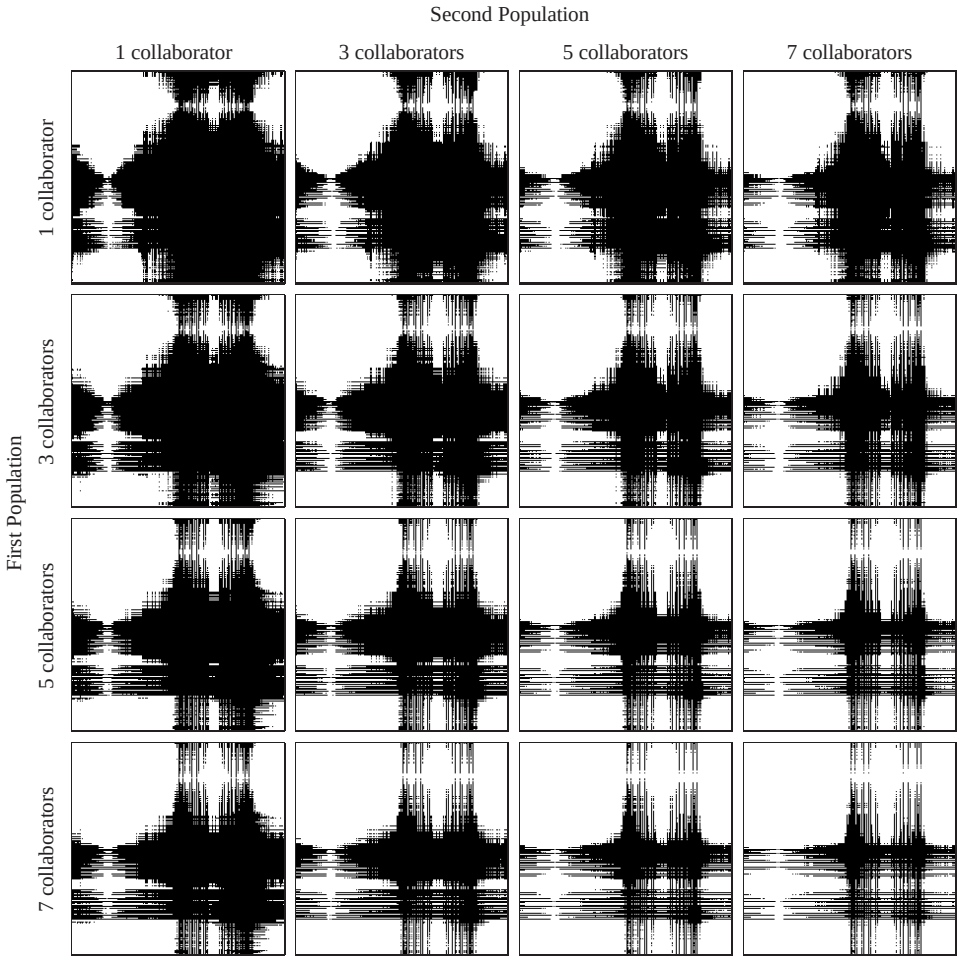

Figure 3.5: Basins of attraction for CCEA in the Climb problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

into six triangles, and each of them is further divided into six more triangles. The center of each resulting triangle corresponds to an initial population. I add random noise distributed uniformly between -0.00005 and 0.00005 to reduce certain artifacts due to identical distributions of the two populations. The sampling also does not cover initial populations on the edges or vertexes of the simplex, but the probability that an evolutionary algorithm starts from those initial populations is 0 anyway.

The right image in Figure 3.4 is an example of the resulting projection of $(\Delta^3)^2$ onto 2-D. Thanks to the sorting described above, certain regions reflect majority-1, majority-2, and majority-3 regions; and borders between those regions are the mixture of the two regions. Dark lines in the figure show locations that have high ratios of 1s, 2s, or 3s in one or the other population.

First, I visualize the impact of improved estimates due to increased numbers of collaborators

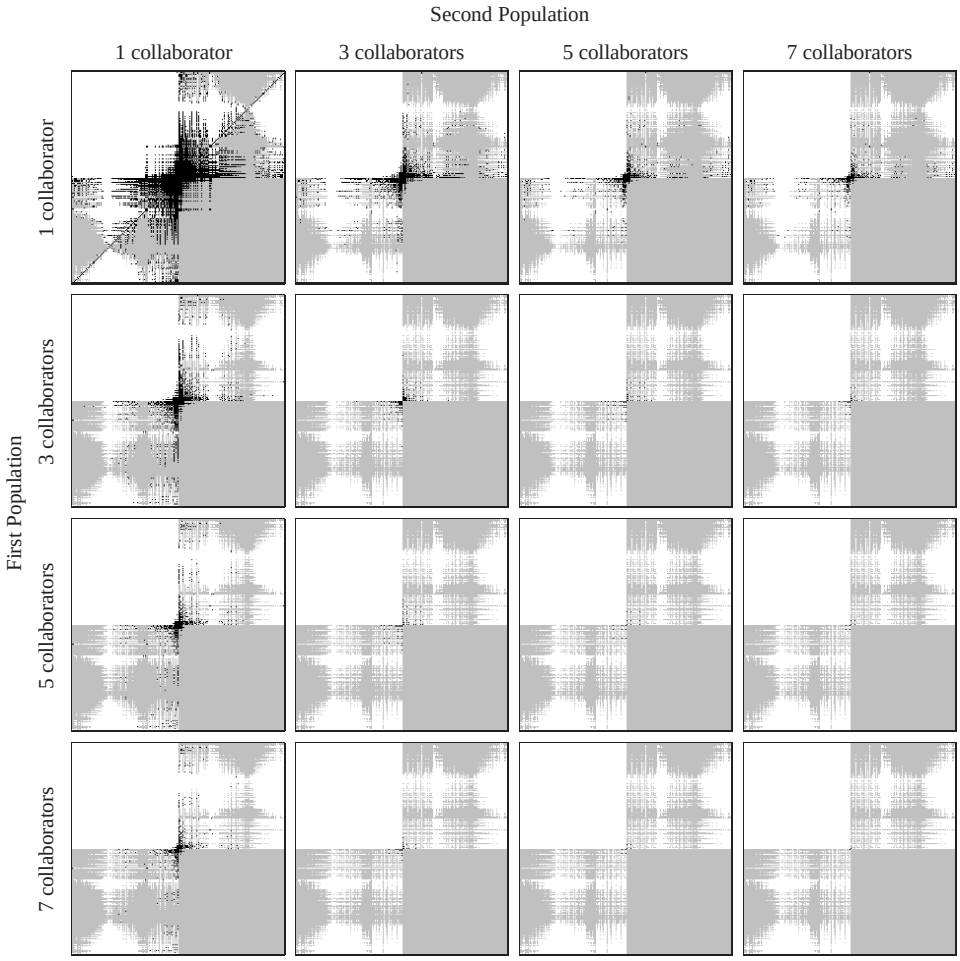

Figure 3.6: Basins of attraction for CCEA in the Penalty problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

for each population (to parallel Theorem 4). Figure 3.5 shows the basins of attraction in the Climb coordination game. The images show that the difficulty of the problem domain decreases as each population is provided with more accurate estimates for the fitness of individuals. When using a single collaborator, it appears that the coevolutionary search will find the optima if at least one of the populations starts with a large number of 1s. Even in this case, the system is most likely to converge to the global optimum if the ratio of 2s is relatively low. As each population gets a better estimate of fitness (via an increased number of collaborators), the basin of attraction for the suboptimal equilibria reduces to areas where at least one of the initial populations has a very large proportion of 2s or 3s: the more collaborators are used, the larger the proportion required to still converge to the sub-optimum.

Figure 3.6 presents the basins of attraction in the Penalty game. Observe that the two global

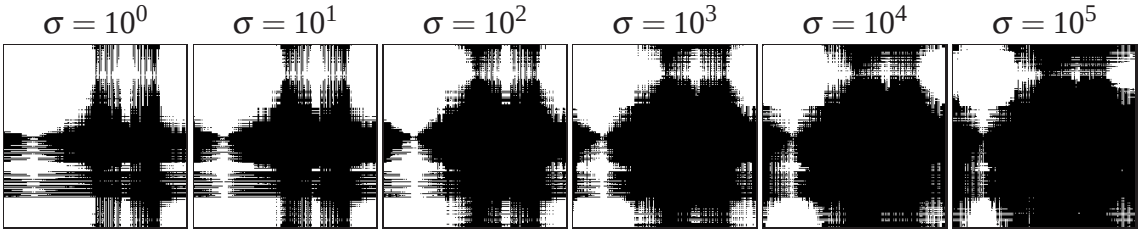

Figure 3.7: Basins of attraction for Nash equilibria for increasing miscoordination penalties σ in the Climb domain. Fitness of individuals in each population is estimated using 3 collaborators. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

optima cover most of the space even when a single collaborator is used; the suboptimal equilibria covers mainly areas where at least one of the population started with a high percentage of 2s, and the other population has the 1s and 3s equally distributed — this increases the percentage of miscoordinations. As the number of collaborators is increased, the basin of attraction for the (2,2) point reduces to only areas where *both* populations start with almost only 2s. The visualization of the basins of attraction suggests that Penalty is a much easier coordination game than Climb. Note also a thin diagonal line in the top-left graph of Figure 3.6. Interestingly, this is due to the fact that if the proportion of 1s in one population is about equal to the proportion of 3s in the other population, there are frequent miscoordinations that impact on the expected reward for these actions as estimated by the learners, and the system converges to the suboptimal (2,2) equilibrium.

Finally, I illustrate the effects of varying the miscoordination penalties onto the basins of attraction of Nash equilibria. For this purpose, I only use the Climb domain with different settings of σ, and I set the number of collaborators $N = 3$ for each population. The basins of attraction are illustrated in Figure 3.7. Observe that the basin of attraction for the optimal solution decreases with more severe miscoordination penalties, as suggested by Theorem 5. Interestingly, the decrease in the size of the basin of attraction for the global optimum does not match the exponential increase in the miscoordination penalty σ; in some sense, this parallels the setting $\rho > \frac{2}{\eta_\varepsilon{}^{NH^t}}$ at the end of Theorem 5.

3.3 EGT Models for Multiagent Q-learning

Q-learning (detailed in Section 1.2.2) is another machine learning techniques that is widely used in single-agent domains due to its well-known properties and convergence guarantees. However, straightforward applications of this technique to multiagent domains (as detailed in Section 1.2.4) have revealed problematic convergence problems to suboptimal solutions (for example, in (Claus and Boutilier, 1998)). In this section, I extend an existing replicator dynamics (RD) model (an EGT

tool) of multiagent Q-learning to show that these problems are also caused by learners having poor estimates for the quality (utility) of their actions. I also extend this model to account for accurate estimates, and I demonstrate the benefits of these changes to the performance of the algorithm. For consistency, the discussion in this section uses the same notations as in Section 3.2.

Consider the application of multiagent Q-learning to solve a simple two-agent coordination game, such as the one in Figure 3.1. Given that the coordination game was stateless, here is how the RD model might be applied:

- Each agent is assigned a population of individuals, where individuals represent actions that that agent can perform in the environment.
- A genotype is equivalent to a specific action (e.g. *move-left* or *shoot-the-ball*). Each individual represents a genotype, but multiple individuals in the same population might represent the same individual (in that case, the agent is more likely to perform that specific action). The total number of genotypes is assumed finite for each agent.
- The populations are assumed infinite, such as in the EGT model for CCEAs. The ratio of a genotype in a population (x_i or y_j, respectively) equals the probability that the agent will choose that specific action (i or j, respectively).
- The utility of an action reflects the quality estimate that the agent assigns to that action. Remember that the estimate is subjective – it reflects the quality of that action when in combination with other actions of the teammate.

To simplify the theoretical analysis, I assume that agents choose actions by using the Boltzmann selection: an action a_k is chosen with probability

$$P(a_k) = \frac{e^{\frac{Q(s,a_k)}{\tau}}}{\sum_i e^{\frac{Q(s,a_i)}{\tau}}} \tag{3.32}$$

where τ is a "temperature" parameter used to balance exploration and exploitation (the agent tends to select actions associated with higher utilities when τ is low). The underlying principle behind replicator dynamics is to compute a set of differential equations that describe the change of the system over time. In particular, the replicator dynamics model for a system with two concurrent learners consists of two differential equations, one for each of the learners. These equations must specify the change of the first learner ($\frac{dx_i}{dt}$), and the change of the second learner ($\frac{dy_i}{dt}$), over time. By combining Equations 1.1 and 3.32, the RD model for multiagent Q-learning derived in (Tuyls *et al.*, 2003; Tuyls *et al.*, 2006) is:

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(\sum_k a_{ik}y_k - \sum_k x_k \sum_j a_{kj}y_j\right) + \alpha\sum_k x_k \ln\frac{x_k}{x_i}$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau}\left(\sum_k a_{kj}x_k - \sum_k y_k \sum_i a_{ik}x_i\right) + \alpha\sum_k y_k \ln\frac{y_k}{y_j}$$

where $u_i = \sum_k a_{ik}y_k$ is the expected reward that would be observed by the first agent, given the other agent's current probabilities of selecting among its actions (and similarly for w_j). The RD model above has therefore the simpler form

$$u_i = \sum_k a_{ik}y_k \tag{3.33}$$

$$w_j = \sum_k a_{kj}x_k \tag{3.34}$$

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(u_i - \sum_k x_k u_k\right) + \alpha\sum_k x_k \ln\frac{x_k}{x_i} \tag{3.35}$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau}\left(w_j - \sum_k y_k w_k\right) + \alpha\sum_k y_k \ln\frac{y_k}{y_j} \tag{3.36}$$

3.3.1 Extending the Formal Model

The RD model in Equations 3.33 — 3.36 assumes that the agent will use all rewards to update the utility of its actions. It is therefore very similar to the EGT model of cooperative coevolutionary algorithms in Equations 3.1 – 3.4. As illustrated in Section 3.1 and demonstrated in Section 3.2, using the average reward usually results in poor estimates for the quality of actions, causing potential drift towards suboptimal solutions.

To remedy this situation, I extend the RD model such that each learner ignores lower rewards to improve its estimate. The key to this extension is in Theorem 2 in Section 3.2.2. The following RD model is a straightforward combination of these previous results.

$$u_i = \sum_{j=1}^{m} a_{ij} \frac{y_j}{\sum_{k:a_{ik}=a_{ij}} y_k} \left(\left(\sum_{k:a_{ik} \leq a_{ij}} y_k \right)^N - \left(\sum_{k:a_{ik} < a_{ij}} y_k \right)^N \right) \quad (3.37)$$

$$w_j = \sum_{i=1}^{n} a_{ij} \frac{x_i}{\sum_{k:a_{kj}=a_{ij}} x_k} \left(\left(\sum_{k:a_{kj} \leq a_{ij}} x_k \right)^N - \left(\sum_{k:a_{kj} < a_{ij}} x_k \right)^N \right) \quad (3.38)$$

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau} \left(u_i - \sum_k x_k u_k \right) + \alpha \sum_k x_k \ln \frac{x_k}{x_i} \quad (3.39)$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau} \left(w_j - \sum_k y_k w_k \right) + \alpha \sum_k y_k \ln \frac{y_k}{y_j} \quad (3.40)$$

3.3.2 Basins of Attraction due to Estimations

This section demonstrates the advantages of improving the learners' estimates for the quality of their actions. To this end, I employ the extended RD model of multiagent Q-learning to visualize the basins of attraction for suboptimal and optimal Nash equilibria in the Climb and Penalty domains.

The visualization method is similar to the one presented in Section 3.2.3. The main difference is that it employs the RD model in Equations 3.37–3.40, instead of the EGT model for CCEAs. Given that the RD model provides differential equations, the next state of the concurrent learning system is approximated by assuming the variations in the derivative are small over short periods of time θ:

$$x_i' = x_i + \theta \frac{dx_i}{dt}$$

$$y_j' = y_j + \theta \frac{dy_j}{dt}$$

I set $\theta = 0.001$, the learning rate $\alpha = 0.1$, and the exploration parameter $\tau = 0.01$. I iterate the RD model 100000 times, or until the proportion of one action in each population exceeds a threshold of $1 - 10^{-10}$.

The basins of attraction for the optimal $(1,1)$ and suboptimal $(2,2)$ equilibria in the Climb domain are visualized in Figure 3.8. Given that the new RD model reduces to the one described in (Tuyls *et al.*, 2006) when $N = 1$, the basin of attraction in the top-left graph indicates that a lot of trajectories for straightforward extensions of Q-learning to multiagent domains will often

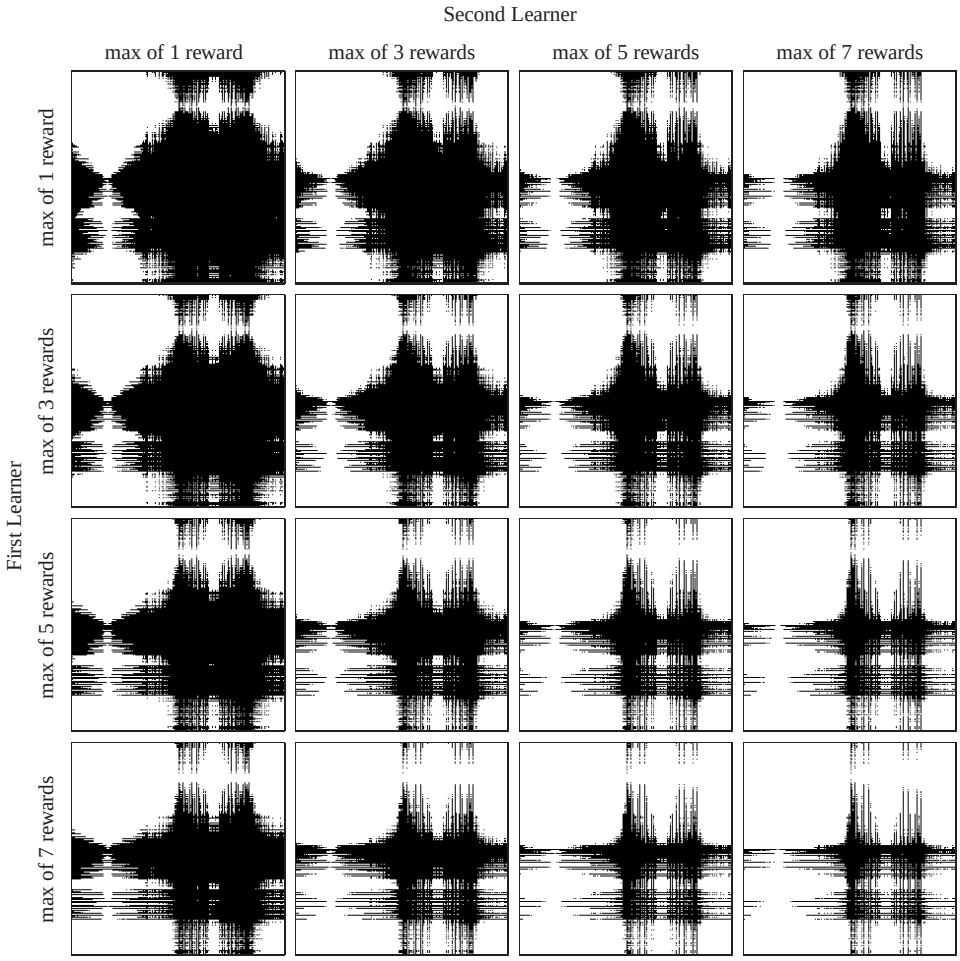

Figure 3.8: Basins of attraction in the Climb problem domain for multiagent Q-learning that updates the utility of an action based on the maximum of 1, 3, 5 and 7 of the rewards received when selecting that action. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

converge to the suboptimal solution. As the learners ignore more of the lower rewards, they improve their estimates for the quality of their actions, and are thus more likely to converge to the global optimum. The same behavior can be observed in the Penalty domain, as illustrated in Figure 3.9.

Interestingly, observe that the basins of attraction for multiagent Q-learning resemble the basins of attraction for cooperative coevolutionary algorithms. To highlight this similarity, Figure 3.10 samples the diagonal graphs in Figures 3.5 and 3.8: these are the cases where both agents ignore the same number of lower rewards. The shape and coverage of the basins of attraction are very similar indeed for each of the four settings. This is at least intriguing, given that the two algorithms have different properties and parameters.

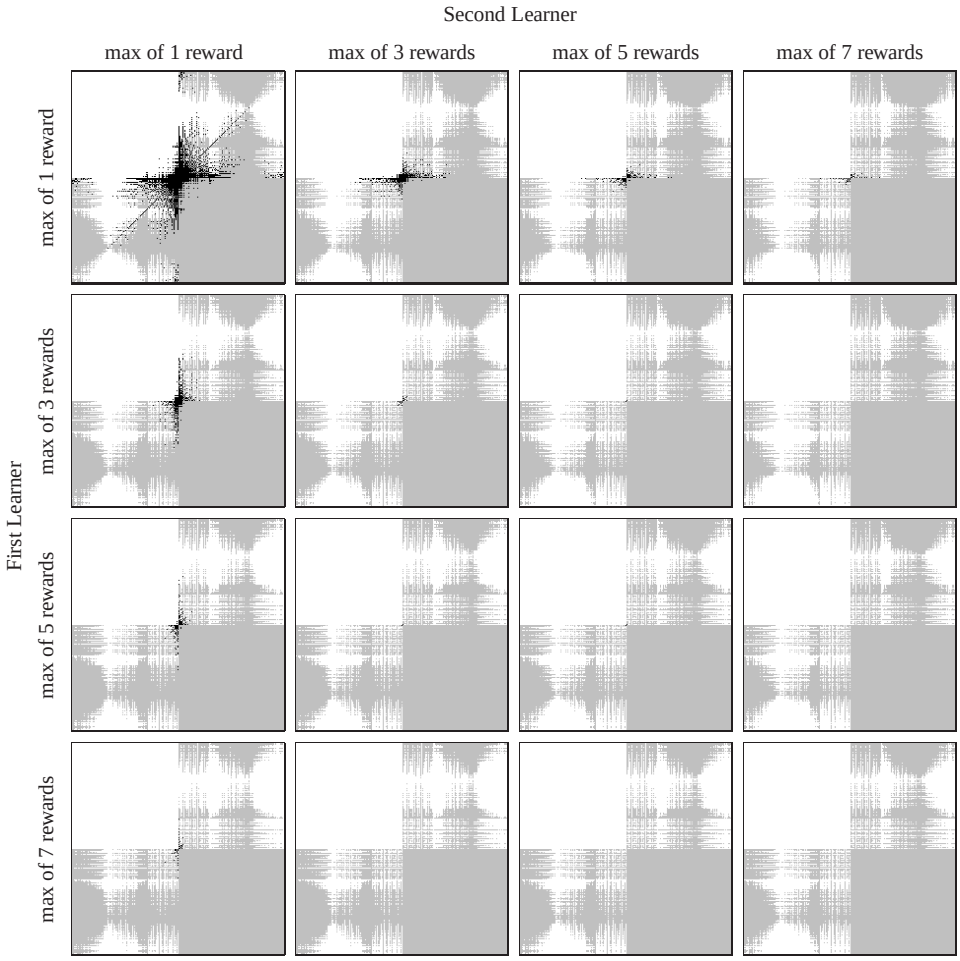

Figure 3.9: Basins of attraction in the Penalty problem domain for multiagent Q-learning that updates the utility of an action based on the maximum of 1, 3, 5 and 7 of the rewards received when selecting that action. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

The similarity of the basins of attraction for the two multiagent learning algorithms should be, however, not surprising at all. Remember that both evolutionary computation and Q-learning are guaranteed to converge to the global optimum in single-agent scenarios, if properly set and if given enough resources. This whole chapter argued that any drift of these algorithms away from the global optimum is only cause by poor estimates, and that it can be dealt with if learners simply ignore lower rewards. Figure 3.10 basically suggests that if the multiagent learning algorithm is drifting towards suboptimal solutions, it is rarely the fault of the learning algorithms. These algorithms do not start to suffer from undesirable pathologies only because they are used in multiagent scenarios. Rather, poor estimates are the primary cause for the drift. As also argued throughout this whole chapter, ignoring lower rewards improves the learners' estimate for the

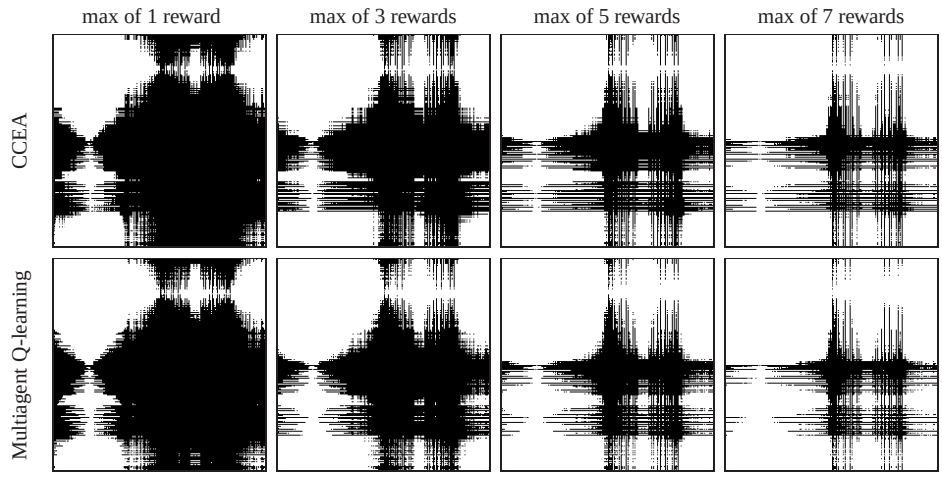

Figure 3.10: Basins of attraction in the Climb problem domain for cooperative coevolution and multiagent reinforcement learning, both ignoring all but the maximum of 1, 3, 5, and 7 rewards. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

quality of their behaviors, resulting in an increased probability that the multiagent learning system will converge to the global optimum.

Chapter 4: The Biased Cooperative Coevolutionary Algorithm

The research in Chapter 3 employed theoretical models of concurrent learning to demonstrate the advantages of accurate estimates for the fitness of individuals. These models involve large numbers of random collaborators to combat poor estimates. However, the analysis also revealed the significant computational resources that might be required to guarantee convergence to the global optimum for methods that rely solely on random collaborators.

This chapter proposes a new biased cooperative coevolutionary algorithm that combines the current joint rewards as observed during the evaluation process, with joint rewards that were observed in the past. This approach complements the use of random collaborators to improve the learners' estimates, while not adding a significant computational overhead. Parts of the material presented herein have been published in (Panait *et al.*, 2006).

The chapter is structured as follows. Section 4.1 details a simple approach to explicitly incorporate bias into CCEAs. Following, Section 4.2 provides theoretical justifications for the method. Section 4.3 details an empirical analysis of several biasing mechanisms, highlighting that one of them is very sensitive to a key parameter. Finally, Section 4.4 provides experimental results for the application of biased cooperative coevolutionary algorithms that employ a simple rote learning method to update the biasing information over the run. The results indicate that biasing enhances both traditional CCEAs, as well as spatially-embedded CCEAs.

4.1 A Naïve Biasing Approach

How might the CCEA be modified such that it is more suitable for learning? One problem with CCEAs is that the fitness of individuals depend only on the current evaluations with collaborators from the other populations. One approach to address this problem is to bias the search by computing an individual's fitness based on two components: its immediate reward when combined with collaborators in the other population, and a heuristic estimate for the reward it would have received had it interacted with its optimal collaborators. The first part of this reward will be called the underlying objective function, and the second will be called the optimal collaborator estimate. δ is a parameter that denotes the degree of emphasis that the fitness places on the optimal collaborator estimate.

Note that this notion of bias toward maximum possible reward has also been used in literature in subtly different ways than is used here. Maximum reward has been used in multiagent reinforcement learning in (Claus and Boutilier, 1998; Lauer and Riedmiller, 2000;

Kapetanakis and Kudenko, 2002b). To some extent, the "Hall of Fame" method introduced in (Rosin and Belew, 1997) for competitive coevolution is also related to biased coevolution; however, that technique samples randomly in the Hall of Fame to increase robustness, while the biased approach in this chapter tends to deterministically select the ideal partners.

Although several experiments in this chapter use an exact computation for the optimal collaborator estimate, I refer to a heuristic estimate because in practice it is highly unlikely that the algorithm will be able to easily compute the actual ideal collaborators. I envision a variety of approaches to computing a heuristic estimate. The estimate might be based on partnering with the most successful collaborators known so far in the population; or with collaborators chosen (or constructed) based on the success they have had with individuals that are structurally "similar" to the test individual; or with collaborators chosen based on past history with the individual's ancestors; and so forth. I reserve comparison of such approaches to future work. Here, I only concentrate on the foundations of the technique itself.

The most obvious approach to bias is through a modified fitness assessment method that is simply a weighted sum of the result of the collaborations and the result of the estimated maximum projection. Equation 4.1 below describes this idea mathematically. Here the fitness of argument x_a is being assessed by combining the result of the underlying objective function, f, with the optimal collaborator estimate, f'_a.

$$\overline{f}(x_1,\ldots,x_a,\ldots,x_k) = (1-\delta)\cdot f(x_1,\ldots,x_a,\ldots,x_k) + \delta \cdot f'_a(x_a) \tag{4.1}$$

At one extreme, when $\delta = 1$, the algorithm will trust only the estimate, and the states of the other populations are entirely irrelevant. It is no longer a coevolutionary system at all; there are rather k EAs searching the k-projected component spaces independently in parallel. At the other extreme, when $\delta = 0$, the algorithm trusts only the underlying objective function. This is the traditional CCEA.

4.2 An EGT Model of Biased CCEAs

This naïve biasing approach can be easily incorporated into the extended EGT model in Equations 3.13 – 3.16: the last two equations (creating the next population via tournament selection) remain unchanged, and the first two equations (computing the fitnesses of individuals) are slightly altered to incorporate the biasing term. The following EGT model of biased CCEAs results:

$$u_i^{(t)} = (1-\delta)\sum_{j=1}^{m} a_{ij} \frac{y_j \left(\left(\sum_{k:a_{ik} \le a_{ij}} y_k \right)^N - \left(\sum_{k:a_{ik} < a_{ij}} y_k \right)^N \right)}{\sum_{k:a_{ik}=a_{ij}} y_k} + \delta \max_{j=1..m} a_{ij}$$

$$w_j^{(t)} = (1-\delta)\sum_{i=1}^{n} a_{ij} \frac{x_i \left(\left(\sum_{k:a_{kj} \le a_{ij}} x_k \right)^N - \left(\sum_{k:a_{kj} < a_{ij}} x_k \right)^N \right)}{\sum_{k:a_{kj}=a_{ij}} x_k} + \delta \max_{i=1..n} a_{ij}$$

$$x_i^{(t+1)} = \frac{x_i^{(t)}}{\sum_{k:u_k^{(t)}=u_i^{(t)}} x_k^{(t)}} \left(\left(\sum_{k:u_k^{(t)} \le u_i^{(t)}} x_k^{(t)} \right)^H - \left(\sum_{k:u_k^{(t)} < u_i^{(t)}} x_k^{(t)} \right)^H \right)$$

$$y_j^{(t+1)} = \frac{y_j^{(t)}}{\sum_{k:w_k^{(t)}=w_j^{(t)}} y_k^{(t)}} \left(\left(\sum_{k:w_k^{(t)} \le w_j^{(t)}} y_k^{(t)} \right)^H - \left(\sum_{k:w_k^{(t)} < w_j^{(t)}} y_k^{(t)} \right)^H \right)$$

In this modified system, the tendency to optimize performance is clear when $\delta = 1$ and there is a unique global optimum. At each iteration of the model, the fitness of each component will be its best possible fitness. If there is a unique global maximum, the system will converge to it (Theorem 1). When there are multiple global maxima, setting $\delta = 1$ is not necessarily a good choice because it provides no incentive for the joint populations as a whole to converge to a single solution (Corollary 1.1). Furthermore, setting $\delta = 1$ may place too much faith on an inaccurate heuristic estimate for the optimal collaborators. When δ is set appropriately, however, biasing can have a considerable (positive) effect, as will be demonstrated next.

I use the visualization technique in Section 3.2.3 to illustrate the theoretical impact of the proposed biasing mechanism onto the basins of attraction of optimal and suboptimal Nash equilibria. For consistency with the illustrations in Section 3.2.3, I visualize the basins of attraction for biased CCEAs applied to the Climb and Penalty domains, and I keep all other settings unchanged. I set $\delta = 0.5$ to provide an intermediate setting between traditional and fully-biased CCEAs. The optimal collaborator is assumed known for any individual. Note that setting δ to 1.0 would cause the biased CCEA to converge to mixed equilibria in the Penalty domain due to the presence of two global optima.

Figures 4.1 and 4.2 show the basins of attraction for the equilibria. The graphs indicate that augmenting CCEAs with biasing can have a considerable positive effect on the resulting basins of attraction. Indeed, a brief visual comparison of Figures 4.1 and 4.2 with Figures 3.5 and 3.6 reveals that the augmented biased CCEA using a very small N is superior to a traditional CCEA using larger values for N. In fact, in the Penalty domain, the basins of attraction for the two globally optimal equilibria cover the entire space, even with a single collaborator.

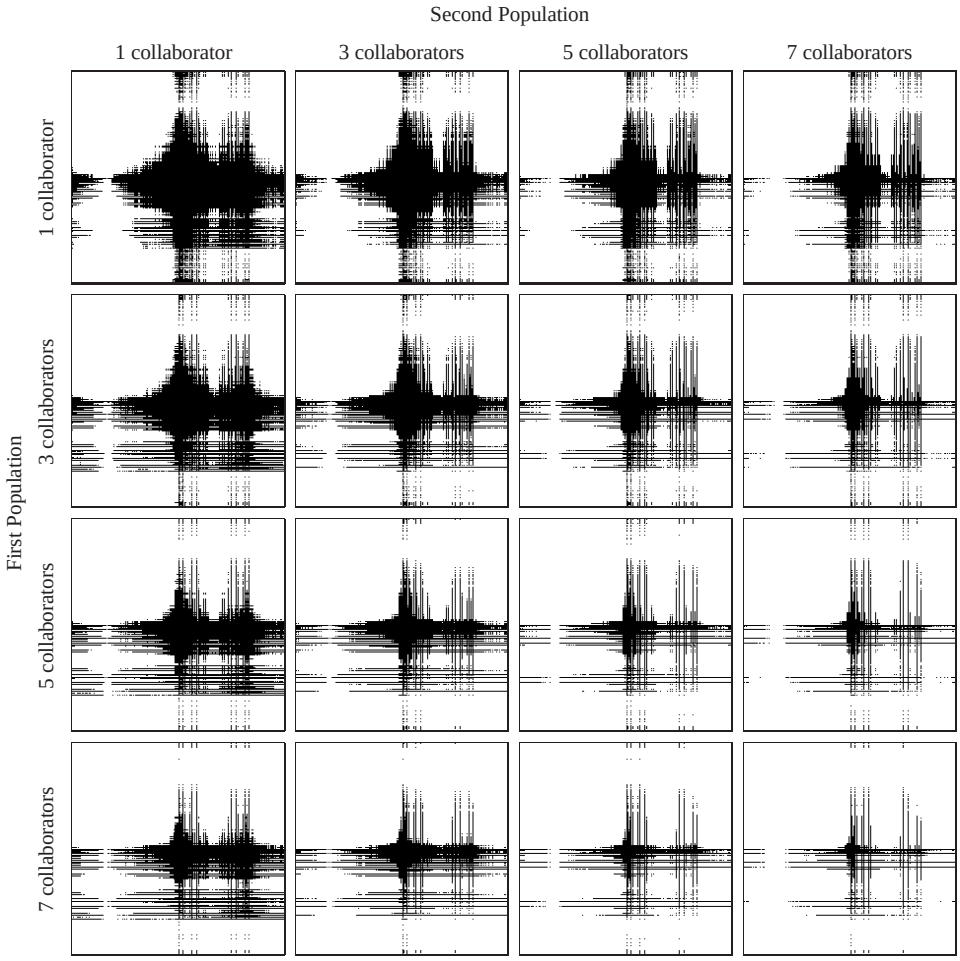

Figure 4.1: Basins of attraction for Biased CCEA in the Climb problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

4.3 Analysis of Sensitivity to the Biasing Rate

A crucial issue remains: what are the effects of choice of δ? As it turns out, naïve approaches to defining δ can result in high sensitivity to the exact value of δ. This is a serious problem if (as would usually be the case) the experimenter does not know the best value of δ beforehand, or chooses to adjust it dynamically during the run. Large amounts of bias may be unwise if optimal-collaborator estimates are poor, or if there are multiple optima. But depending on problem properties, small amounts of bias may have almost no effect. Further, certain system settings may decrease an algorithm's sensitivity to the degree of bias or exacerbate it. A smooth, relatively insensitive biasing procedure is necessary for the success of biased coevolution, as it mitigates radical, unexpected changes in the algorithm properties due to changes in choice of δ.

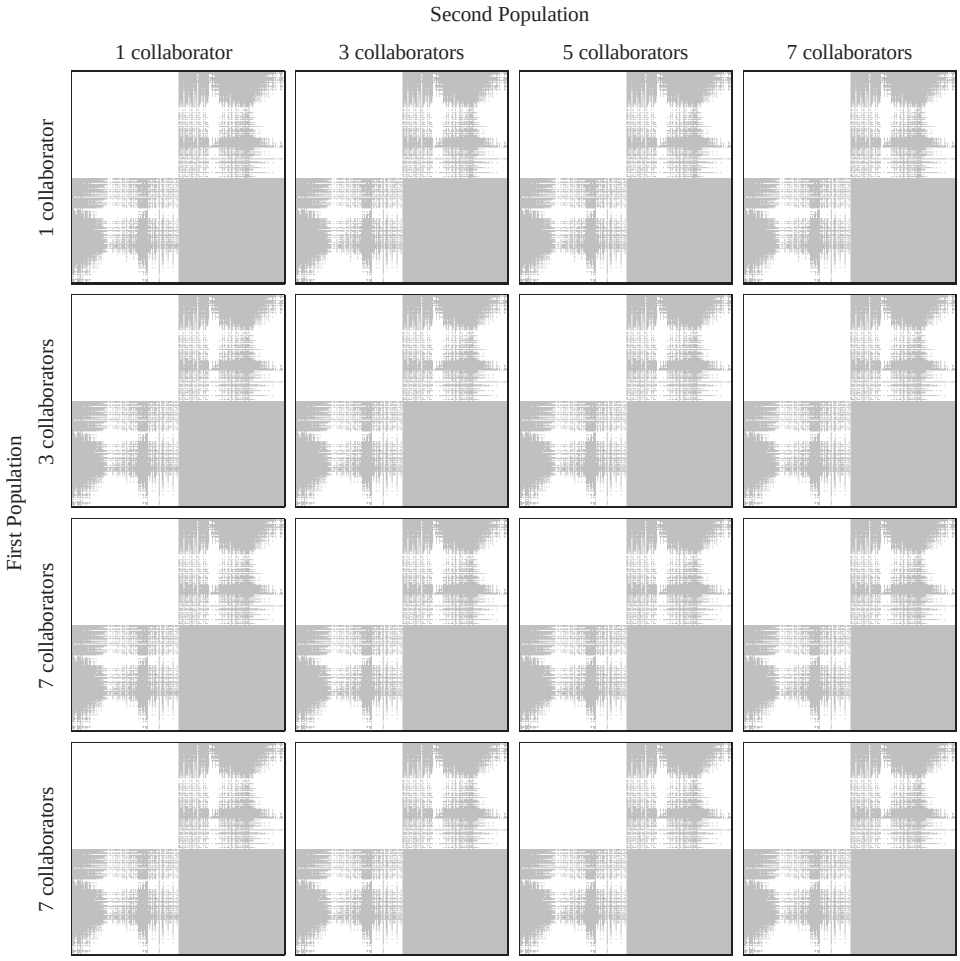

Figure 4.2: Basins of attraction for Biased CCEA in the Penalty problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

To examine this sensitivity to δ, I make two relatively straightforward and obvious simplifications. First, I consider only a static value for δ throughout a run and focus the attention on how different static values affect final runtime performance. Though a more realistic algorithm (such as in Section 4.4) would likely adjust the degree of bias dynamically throughout the run, it is unclear how best to do this. Second, I assume that the biasing information (the optimal collaborator for each individual) is known a priori. In other words, I assume that the function g'_a is known beforehand. This simplification is made in order to reduce the variables involved in the experiment, and it is reasonable because there remain important sensitivity issues to consider even with this simplification. The experiments detailed later in Section 4.4 will relax both of these simplifications to account for more realistic problems and algorithms.

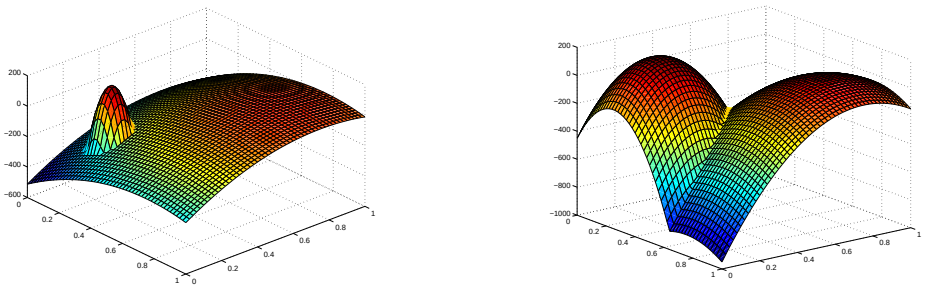

Figure 4.3: MTQ instances illustrating $(S_1 = 1.6, S_2 = 0.5)$ and $(S_1 = 0.5, S_2 = 0.5)$.

4.3.1 Problem and Algorithm Properties

Consider the maximum of two quadratics (MTQ) class of problem domains, which can offer a range from simple to very difficult instances (Wiegand, 2004). MTQ is a class of two-dimensional functions defined as the maximum of two quadratic polynomials. One advantage of the MTQ functions is that the maximum projection can be easily computed. The MTQ class is defined as:

$$MTQ(x,y) = \max \begin{cases} H_1 \times (1 - \frac{16\times(x-X_1)^2}{S_1} - \frac{16\times(y-Y_1)^2}{S_1}) \\ H_2 \times (1 - \frac{16\times(x-X_2)^2}{S_2} - \frac{16\times(y-Y_2)^2}{S_2}) \end{cases} \tag{4.2}$$

where x and y take values ranging between 0 and 1. Figure 4.3 illustrates some example MTQ problem instances. Different settings for H_1, H_2, X_1, Y_1, X_2, Y_2, S_1, and S_2 affect the difficulty of the problem domain in one of the following aspects:

Peak height H_1 and H_2 affect the heights of the two peaks. Higher peaks may increase the chances that the algorithm converges there.

Peak coverage S_1 and S_2 affect the area that the two peaks cover: a higher value for one of them results in a wider coverage of the specific peak. This makes it more probable that the coevolutionary search algorithm will converge to this (possibly suboptimal) peak.

Peak relatedness The values X_1, Y_1, X_2, and Y_2 affect the locations of the centers of the two peaks, which in turn affect their relatedness: similar values of the x or y coordinates for the two centers imply higher overlaps of the projections along one or both axes.

Aside from the impact of the properties of the problem domain, the sensitivity study targets three algorithmic settings:

Bias rate How does the performance degrade as the bias rate δ changes? The experiments report on the impact of different bias rates δ on the performance of the coevolutionary system.

Population size The population size affects how much the coevolutionary algorithm samples the search space. Coevolution using a bias rate δ of 1.0, combined with infinite populations and perfect knowledge of maximal projections, will converge to the unique optimum with probability 1 (Theorem 1). Large populations should achieve similar results as well, particularly in simple domains where the maximal projections can be easily approximated.

Collaboration scheme The CCEA attempts to simplify the search process by decomposing the candidate solutions into components and coevolving them in separate populations. The only information a population can get about the overall progress of the search process is through collaborators — samples that are usually representative of the status of the other populations. Varying the number of collaborators presents a tradeoff between computational complexity and the efficiency of the algorithm (Wiegand *et al.*, 2001; Bull, 1997): more collaborators induce an increased computational complexity, but the performance of the search might also be significantly improved.

4.3.2 Sensitivity Results

All experiments here use the MTQ class of problems. The coevolutionary search process employs two populations, one for each variable. Each population uses a real-valued representation, with individuals constrained to values between 0 and 1 inclusive. Parents are selected via tournament selection of size 2, and non-adaptive Gaussian mutation (mean 0 and standard deviation 0.05) is the only variational operator. The best individual in each population is automatically cloned to the next generation. The search lasts for 50 generations, after which the best individuals in each population are at, or very near, one of the two peaks. Each point in Figures 4.4–4.7 is computed over 250 independent runs. All experiments are performed with the ECJ system (Luke, 2005).

Other default settings are as follows. Each population consists of 32 individuals. The default collaboration scheme uses two collaborators from each population: the best individual in the previous generation is always selected, and the other individual is chosen at random[1]. The biasing method combines the a priori fitness with the better of the results obtained when the individual is teamed with each of the two collaborators. The default values of the parameters for the first (suboptimal) peak are H_1=50, X_1=$\frac{1}{4}$, Y_1=$\frac{1}{4}$, and S_1=1.6. The second (optimal) peak is characterized

[1]To reduce noise in the evaluation process, the experiments in this section employ the same random collaborators for all individuals in the population at that generation. While using different random collaborators for different individuals may result in slightly better performance due to better sampling of the search space, the sensitivity to δ is unaltered.

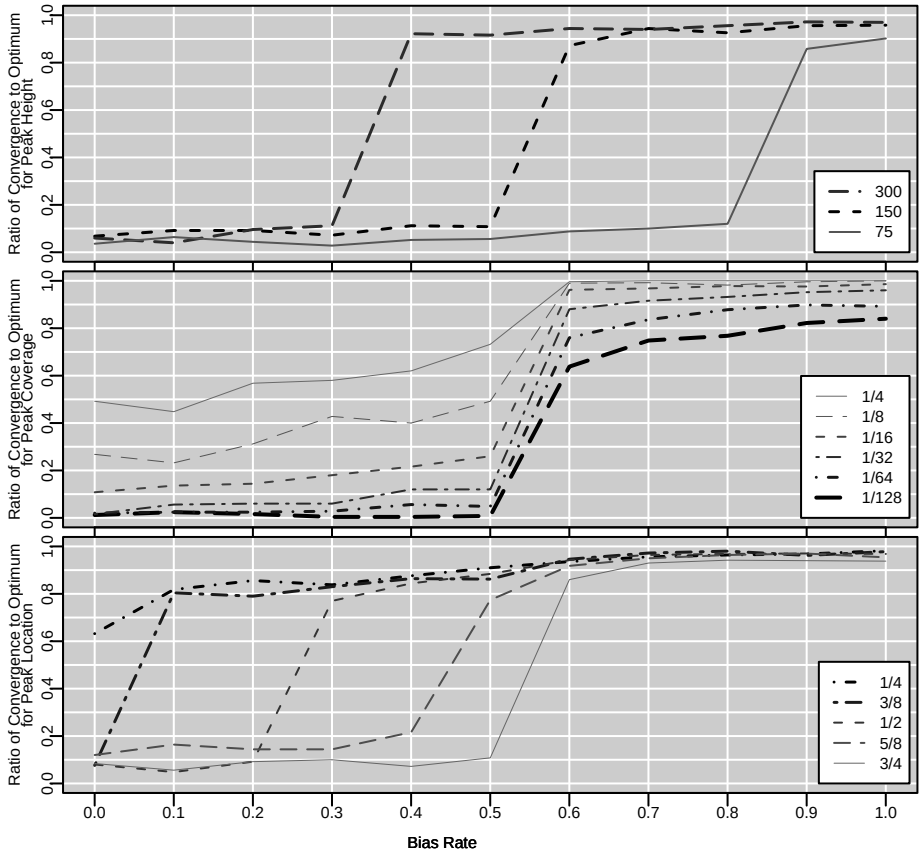

Figure 4.4: Convergence ratios to global optimum for peak height (top), peak coverage (center), and peak relatedness (bottom).

by H_2=150, X_2=$\frac{3}{4}$, Y_2=$\frac{3}{4}$, and S_2=$\frac{1}{32}$. With these settings, the two peaks are nearly at opposite corners of the domain space.

Biasing and Domain Features

The first set of experiments investigates the relationship between the bias rate and the three problem domain features described previously: the relative heights, coverages and locations of the peaks. There are 11 experimental groups for each property, one for each value of $\delta \in [0,1]$ in increments of 0.1. Figure 4.4 shows the mean final results of these 33 groups.

Peak height The experiments employ three settings for H_2: 75, 150, and 300. The results indicate that less than 10% of runs converged optimally when the bias rate is low, while the ratio increases to over 90% when using high bias rates. Unfortunately, there is no smooth transition between these two extremes: rather, small modifications to the bias rate can change the rate of

convergence to the optimum by as much as 70 – 80%. Moreover, the relative difference in peak height directly affects where these sudden jumps in performance appear. This suggests that the algorithm may not only be quite sensitive to δ with respect to changes in relative peak height, but also suggests that it may be difficult to predict *where* the sudden transitions occur.

Peak coverage I experiment with six values for S_2: $\frac{1}{128}$, $\frac{1}{64}$, $\frac{1}{32}$, $\frac{1}{16}$, $\frac{1}{8}$, and $\frac{1}{4}$. The results indicate that the location of the transition is more consistent among the various values, but the transitions themselves are still abrupt. It also appears that the relative peak coverages cause more variation in results when the bias rate is small, while the curves at the other extreme of the graph appear close together. The results indicate that the peak coverage will affect the algorithm's sensitivity to the δ parameter: the wider the peak, the more gradual the transition when varying the bias rate.

Peak relatedness The experiments employ five values for Y_2: $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{5}{8}$, and $\frac{3}{4}$. These settings gradually transition the relative peak positions from diagonally opposite locations to ones aligned along one axis. Similar to peak height, the peak relatedness has a significant effect on the ratio of runs that converge to the global optimum: the more related the peaks, the less biasing is required to assure good performance. However, the curves have an abrupt transition between lower and higher rates of convergence to the optimum. Moreover, the location of this transition depends on the actual degree of peak relatedness, which suggests that the algorithm may be highly sensitive to δ with respect to this parameter.

Biasing and Algorithm Features

A second set of experiments investigates the relationship between the bias rate and the population size and collaboration scheme. Again, there are 11 groups for each of these two parameters corresponding to each of the δ settings. Figure 4.5 details the results.

Population size I set the size of each of the two populations to 8, 16, 32, 64, 128, 256, and 512. As expected, extremely small populations are less likely to reach the optimum, even with high bias rates. The results suggest that increasing the population size does not necessarily alleviate algorithm sensitivity to δ: the graphs contain the same abrupt shift in performance as in the previous experiments.

Collaboration scheme Finally, I vary the number of random individuals from 0 to 4; the best individual from each population in the previous generation is always used. The bottom graph in Figure 4.5 shows that the collaboration scheme has some influence over the performance of the

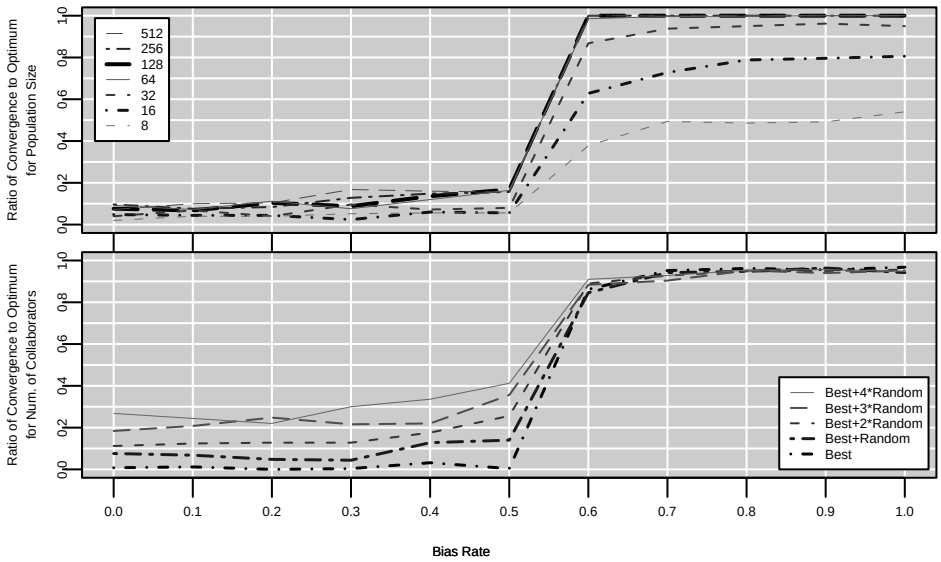

Figure 4.5: Convergence ratios to global optimum for population size (top) and collaboration scheme (bottom).

algorithm at low bias rates, but it has no effect when higher bias rates are used. Again, the abrupt change in performance indicates that the algorithm can be highly sensitive to the δ parameter, regardless of the collaboration methodology.

The results of nearly all of these experiments indicate that the naïve biased CCEA is very sensitive to δ, and that this problem cannot be alleviated by only adjusting the algorithmic parameters. As argued before, this is problematic. In the next section, I detail a nonparametric biasing method that is fairly robust to changes in δ.

4.3.3 An Alternative Stochastic Biasing Mechanism

To uncover a possible simple alternative that does not share this problem, recall that for larger differences in peak heights, a wider range of bias rates results in a high ratio of convergence to optimum; however, when one peak is only slightly higher than the other, the range of high convergence ratios is much smaller. The transition is abrupt, and the location of the transition shifts depending on the differences in peak height.

This extreme sensitivity of the algorithm to the biasing method with respect to the relative peak heights is caused by the linear combination of the two fitness components: the fitness when teamed with collaborators, and the fitness when in combination with the optimal collaborator. The higher the optimal peak, the lower the bias rate it needs to dominate the other term. However, if one peak is slightly higher than the other, the algorithm requires more biasing to locate the optimum.

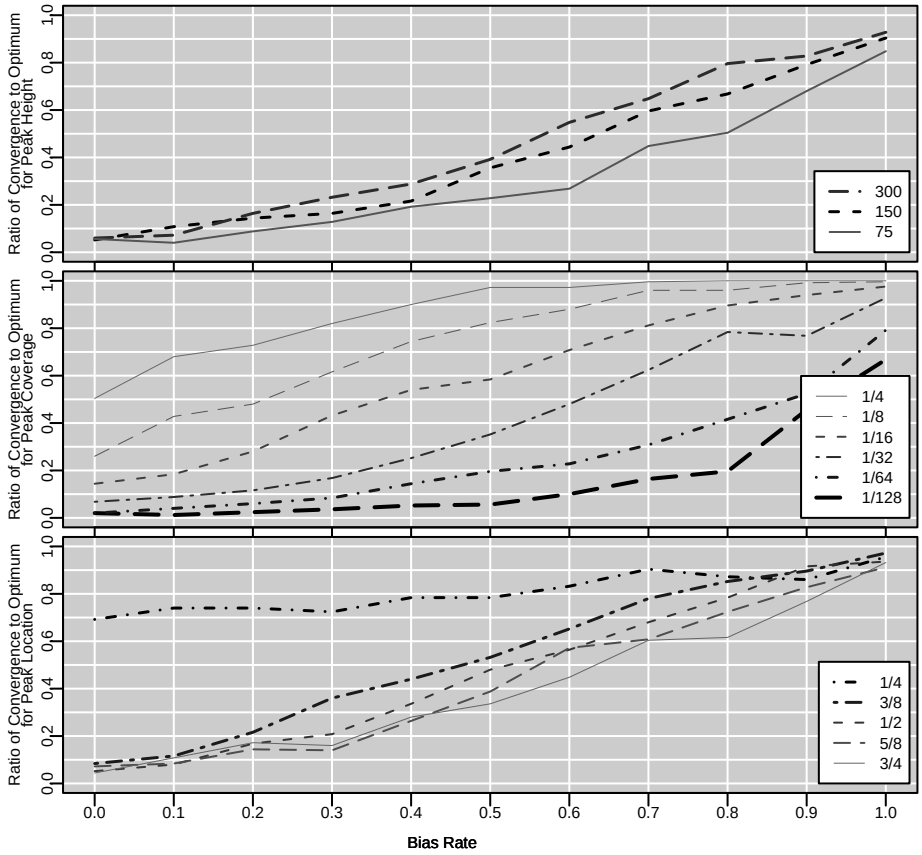

Figure 4.6: Convergence ratios to global optimum for stochastic biasing when varying peak height (top), peak coverage (center) and peak relatedness (bottom).

To counter this, I propose a nonparametric comparison method which considers the relative order of two components rather than their exact values. The justification for this technique is similar to that of nonparametric selection methods such as tournament selection (Goldberg and Deb, 1990), rank selection (Whitley, 1989), and truncation selection (Mühlenbein and Schlierkamp-Voosen, 1993). The new nonparametric biasing technique works as follows: each individual is assigned two fitnesses: the underlying objective one when combined with the collaborators from other populations, and another one indicating the performance of the individual when in combination with its optimal collaborator. When two individuals are compared to select a parent, they are compared based on the first "fitness" with probability δ, and on the second "fitness" with probability $1 - \delta$.

Figures 4.6 and 4.7 detail the results of a similar sensitivity study for the nonparametric biasing technique. Observe that the new algorithm does not exhibit the sudden jumps in performance that the original did.

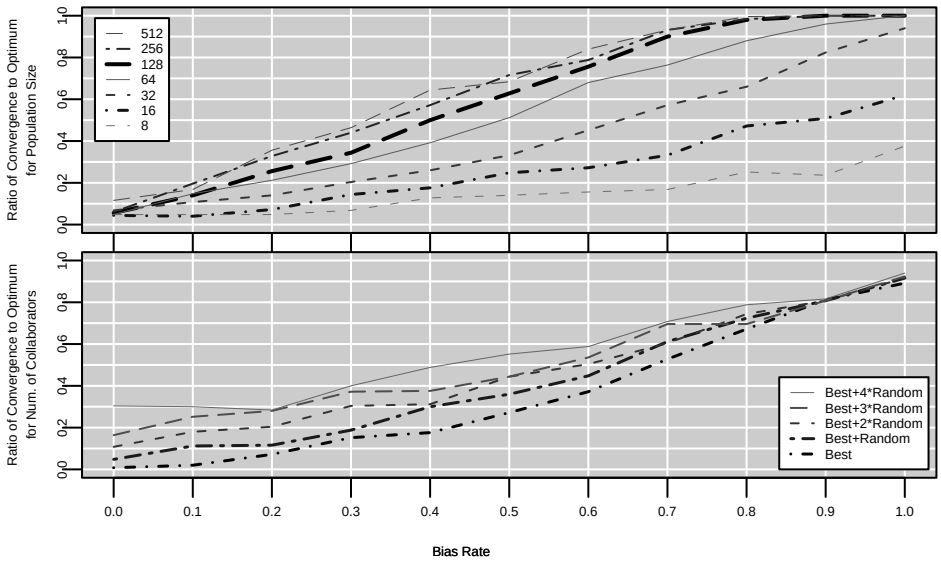

Figure 4.7: Convergence ratios to global optimum for stochastic biasing when varying population size (top) and collaboration scheme (bottom).

4.4 Comparing Realistic Implementations of Traditional and Biased CCEAs

While the previous theoretical discussion, as well as the sensitivity analysis just discussed, helps justify the intuition for biasing the fitness evaluation, neither is immediately applicable to real problems. In a more realistic setting, simplifying model assumptions such as infinite populations, lack of variational operators, and a priori knowledge of the maximum payoff are not possible. To convert theory into practice, these assumptions have to be dealt with.

4.4.1 Method of Study

These experiments employ practical coevolutionary algorithms with finite population and with variational operators. The biasing information used by these algorithms is updated via rote learning. Specifically, the optimal collaborator for an action i is an action j which *so far* has shown the highest performance when paired with i. As evolution progresses, the optimal collaborator of an action i changes to reflect the better (i, j) pairs that have been evaluated. The main difference between the two representations in Sections 4.4.3 and 4.4.4 is primarily in how an individual chooses its actions.

The stochastic biasing mechanism detailed in Section 4.3.3 is employed to improve the coevolutionary search. However, δ is not fixed as in previous experiments, but it is rather varied throughout a single run: the algorithm starts with $\delta = 1$, decreasing it linearly until it reaches $\delta = 0$

at 75% of the total number of generations, at which point δ remains constant 0 until the end of the run. While the dynamic adjustment of δ is ad-hoc, it is sufficient to demonstrate that the biased coevolutionary algorithms outperform their unbiased counterparts.

I perform several experiments to compare traditional coevolution with biased coevolution in this context. These experiments involve the Climb and the Penalty coordination games introduced in Section 3.2.3. The miscoordination penalty ρ is set to 30 in the Climb domain and to 10 in the penalty domain, to reflect the standard benchmarks used elsewhere. I also test the algorithms on an MTQ-like problem termed Two Peaks, which has the joint reward function defined as

$$f(x,y) = \max \begin{cases} 0 \\ 10 - 32 \times \left(\left(x - \frac{1}{3}\right)^2 + \left(y - \frac{1}{3}\right)^2 \right) \\ 15 - 128 \times \left((x-1)^2 + (y-1)^2 \right) \end{cases}$$

with x and y taking values between 0 and 1. Finally, the experiments involve a cooperative learning domain with increased non-linear interactions: the joint reward function is based on the two-dimensional Rosenbrock function

$$f(x,y) = -\left(100 \times \left(x^2 - y\right)^2 + (1-x)^2\right)$$

where x and y take values between -2.048 to 2.048. For simplicity, each axis is discretized into 16, 32, 64, and 128 segments for experiments with both the Two Peaks and the Rosenbrock functions. Increased discretization results in larger search spaces, but not necessarily more difficult ones — when searching for pure strategies for the Rosenbrock domain (Table 4.4), the rate of finding the global optima for all coevolutionary methods for 32 intervals is lower than that for 64 intervals. The experiments are performed using the ECJ software package (Luke, 2005). In order to establish statistical significance, all claims of "worse" or "better" are verified using nonparametric tests: I use the Welch test (a variation the Student t-test that does not assume equal variance for the samples) repeatedly for pairs of samples. Given that the samples rarely follow a normal distribution, I first rank the set of observations from both samples, then I perform the Welch test on those ranks. The Bonferroni inequality is used to adjust the p-value level for each test such as to obtain 95% confidence over all comparisons; as a consequence, each Welch test is applied at a 99.95% confidence level.

4.4.2 Competing Techniques

I consider both biased and unbiased versions of three CCEAs. The first such algorithm is a "traditional" CCEA. The others are two spatially-embedded CCEAs. They are detailed next.

The traditional CCEA uses a common fitness assessment: an individual is assessed twice to determine its fitness, once with a collaborator chosen at random, and once partnered with the individual in the other population that had received the highest fitness in the previous generation. An individual's fitness is set to the maximum of these two assessments. This algorithm is termed *Traditional* for ease of reference.

In a spatially distributed CCEA, the individuals are positioned at specified locations in a geometric space, such that a notion of a neighborhood exists among individuals (Sarma, 1998; Giacobini *et al.*, 2003; Dorronsoro *et al.*, 2004; Pagie, 1999; Hillis, 1991; Husbands, 1994; Ronge and Nordahl, 1996; Wiegand and Sarma, 2004; Williams and Mitchell, 2004). For consistency across small and moderate population sizes, each population is embedded in a one-dimensional ring. A neighborhood of radius 1 for an individual consists of three individuals in this case: the specific individual, together with the individuals to its immediate left and right (on the ring). The spatial embedding of the populations influences the breeding process as follows: for each location, multiple individuals are selected with replacement from the local neighborhood (the radius of the neighborhood is detailed for each problem domain later), and the better ones are selected for breeding (the best individual is selected for mutation alone, or the better two individuals are selected for crossover, followed by mutation). When creating a child for location *i*, the parent at location *i* always competes for selection to breed.

The spatial embedding also influences the scheme to select collaborators. I experiment with two spatial collaboration schemes. First, each individual is evaluated with the unique collaborator from the other population that has the same location in space. This setting is referred to as *Spatial*. The population size for Spatial is doubled to allow it to have the same total number of evaluations as the other methods. A second spatially-embedded CCEA evaluates each individual with two collaborators: the collaborator at the same location in space (as before), and a random collaborator from a small neighborhood (the radius of the neighborhood is detailed later). This second technique is referred to as *Spatial2* in the remainder of this chapter.

The combination of biasing with each of the three algorithms is termed *Biased Traditional*, *Biased Spatial*, and *Biased Spatial2*, respectively.

4.4.3 Searching for Pure Strategies

A first set of experiments encodes a single action (an integer) in each individual. In other words, each individual deterministically specifies an action. In game-theory parlance, each individual thus represents a "pure strategy". Such an individual breeds children through mutation: the individual's integer is increased or decreased (the direction chosen at random beforehand with probability 0.5) while a biased coin is flipped and lands in the "heads up" position (with probability $\frac{1}{8}$ for Climb and Penalty, and with probability $\frac{1}{4}$ for Two Peaks and Rosenbrock). Evolutionary runs in the Climb

Table 4.1: Percentage of CCEA runs that converged to global optimum, Climb domain with pure strategy representation

	Penalty			
	-30	-300	-3000	-30000
Traditional	56.1%	56.9%	56.8%	56.9%
Biased Traditional	79.9%	77.6%	80.8%	81.0%
Spatial	76.4%	79.7%	77.0%	77.2%
Biased Spatial	85.6%	88.2%	88.4%	87.0%
Spatial2	67.1%	69.8%	71.5%	70.0%
Biased Spatial2	82.4%	80.9%	81.7%	82.7%

Table 4.2: Percentage of CCEA runs that converged to global optimum, Penalty domain with pure strategy representation

	Penalty			
	-10	-100	-1000	-10000
Traditional	88.2%	89.4%	90.3%	88.4%
Biased Traditional	93.2%	93.5%	91.9%	93.4%
Spatial	99.3%	98.9%	99.3%	98.9%
Biased Spatial	99.7%	99.4%	99.3%	99.4%
Spatial2	93.5%	93.1%	94.8%	92.6%
Biased Spatial2	94.6%	95.8%	96.2%	94.2%

and Penalty problem domain use only 3 individuals per population (Spatial uses 6 individuals) and they last for 40 generations. Runs in the Two Peaks and Rosenbrock domains use 20 individuals per population (Spatial uses 40) and they last for 200 generations. Spatial2 selects the second collaborator randomly using a neighborhood of radius 1. Traditional and Biased Traditional use tournament selection of size 2 for breeding, and the most-fit individual is automatically cloned from one generation to the next. Parents are selected from neighborhoods of radius 1 using tournament selection with size 2 for each location in the spatially-embedded models.

Results Summary The use of the proposed biasing mechanism usually results in statistically significant improvements in the rate of finding the global optima. In the few situations where biasing does not help, it does not hurt performance either. As a side-note, the Spatial algorithm consistently outperforms the traditional CCEA.

Results Specifics Tables 4.1–4.4 present the average percentage (out of 1000 runs) that converged to the global optimum. Overall, the spatial methods outperform the traditional method — not surprising given the positive results in the literature, as reported for example

Table 4.3: Percentage of CCEA runs that converged to global optimum, Two Peaks domain with pure strategy representation

	Discretization Level (Number of Actions)			
	16	32	64	128
Traditional	51.6%	50.8%	49.6%	49.0%
Biased Traditional	68.5%	65.8%	59.2%	59.4%
Spatial	86.5%	91.2%	89.3%	85.6%
Biased Spatial	84.1%	88.9%	87.9%	86.6%
Spatial2	72.0%	73.2%	69.3%	66.5%
Biased Spatial2	78.6%	74.7%	72.5%	68.4%

Table 4.4: Percentage of CCEA runs that converged to global optimum, Rosenbrock domain with pure strategy representation

	Discretization Level (Number of Actions)			
	16	32	64	128
Traditional	82.5%	33.6%	37.9%	16.5%
Biased Traditional	87.1%	48.8%	51.9%	21.9%
Spatial	84.3%	40.8%	45.4%	22.1%
Biased Spatial	86.6%	56.6%	82.0%	39.8%
Spatial2	78.2%	33.5%	41.2%	16.4%
Biased Spatial2	74.5%	42.0%	67.3%	22.8%

in (Pagie, 1999) — but the biased version of any method generally outperforms the unbiased version of that method. In the Climb domain, Spatial is significantly better than both Traditional and Spatial2 (Spatial is better than Spatial2 with only 99.914% confidence for Penalty=-3000). For all three methods, biasing significantly improves performance — Biased Spatial in particular converges to the global optima in about 90% of the runs, significantly better than all five other methods.

Spatial is again better than both Traditional and Spatial2 in the Penalty domain. Except for significant improvements of Biased Traditional over Traditional when Penalty=-10 and Penalty=-10000, biasing is not effective at improving results at the 99.95% confidence level (though it does not damage results either). I perform three additional tests using all 4000 runs for each of the methods (1000 for each value of the penalty); the increased number of observations permits to state that biasing is effective at significantly improving the performance of Traditional and (with only 99.89% confidence) Spatial2.

In the Two Peaks domain, Spatial is again better than Spatial2, which is better than Traditional. Enhancing the techniques with the proposed biasing mechanism results in significant improvements for Traditional (with only 99.4% confidence for 128 discretization level), and for

Spatial2 (only for a discretization level of 8). All other differences are statistically insignificant.

In the Rosenbrock domain, Spatial is better than Traditional (with 99.95% confidence for discretization level 128, and only with 99.9% and 99.85% confidence for discretization levels 8 and 16, respectively) and Spatial2 (with 99.95% confidence for discretization levels of 64 and 128, and only with 99% confidence for discretization level 8). Additional nonparametric tests using all 4000 runs indicate that Spatial is significantly better than Traditional and Spatial2 with 99.95% confidence. The methods in combination with biasing usually perform better than alone; no method is ever worse due to biasing.

4.4.4 Searching for Mixed Strategies

Though the pure strategy representation provides a clear connection to theory, its use in these simple problems results in very small search spaces. It would be nice to consider larger problems with similar properties. I accomplish this by encoding a "mixed strategy" (to again use game theory parlance) in each individual. More specifically, individuals consist in this case of a probability distribution over the available actions. When evaluating such individuals with a collaborator (another mixed strategy), 50 independent interactions are performed, each consisting of a joint action chosen at random according to the individuals' mixed strategies. The joint reward for the two individuals is computed as the average reward over the 50 joint rewards. Observe that using mixed strategies creates a potentially more difficult problem domain than using pure strategies for reasons of both search space size and the stochastic nature of the fitness result.

I perform a similar empirical comparison of the six algorithms using the mixed strategy representation. The settings are detailed next. Traditional and Biased select parents via tournament selection of size 2; breeding involves one-point crossover, followed by mutation by adding random Gaussian noise (mean 0 and standard deviation 0.25) with probability $\frac{1}{L}$ for each of the distribution values (where L is the number of actions in the problem domain), followed by the renormalization of the distribution. Runs last for 200 generations in the Climb and Penalty domains, and for 1000 generations in the Two Peaks and Rosenbrock domains. 1000 runs are performed for each treatment to grant statistical significance.

An extensive sensitivity study is used to identify proper values for the parameters of the spatially-embedded coevolutionary algorithms. The results indicate that lower mutation rates work better (following crossover, Gaussian random noise is added to each gene with probability 0.2 for the Climb and Penalty domains, and only with probability $\frac{1}{3L}$ for Two Peaks and Rosenbrock). When using the Traditional and the Spatial2 methods, each population contains 20 individuals for Climb and Penalty, and 100 individuals for the Two Peaks and Rosenbrock domains (as noted previously, Spatial uses twice the population size for an equivalent number of evaluations). Parents

Table 4.5: Percentage of CCEA runs that converged to global optimum, Climb domain with mixed strategy representation

	Penalty			
	-30	-300	-3000	-30000
Traditional	25%	20%	19%	21%
Biased Traditional	100%	100%	100%	100%
Spatial	67%	28%	27%	26%
Biased Spatial	100%	100%	100%	100%
Spatial2	50%	26%	25%	27%
Biased Spatial2	99%	99%	99%	99%

Table 4.6: Percentage of CCEA runs that converged to global optimum, Penalty domain with mixed strategy representation

	Penalty			
	-10	-100	-1000	-10000
Traditional	100%	99%	99%	99%
Biased Traditional	100%	100%	100%	100%
Spatial	100%	99%	99%	98%
Biased Spatial	100%	100%	99%	99%
Spatial2	99%	99%	98%	98%
Biased Spatial2	100%	100%	100%	99%

are selected using tournament selection with size 2 from a neighborhood of radius 1 for Climb and Penalty. Given the larger population sizes for Two Peaks and Rosenbrock, parents are selected from neighborhoods of radius 3; the sensitivity study also suggests a tournament selection size of 5 for the Two Peaks domain, and of 3 for the Rosenbrock domain.

The mixed representation introduces an intriguing problem: what does the optimal collaborator for a mixed strategy look like, and how can it be learned? I opt for the following approach: whenever an action i is selected from a mixed strategy, the optimal collaborator for that mixed strategy will deterministically select action j, where the pair (i, j) has received the highest reward in the past for action i. Only the first joint reward (of the total of 50) is used from each evaluation of a pair of individuals (mixed strategies) to update the history information. To do otherwise would give the estimation procedure an undue advantage[2] over the case study involving the pure-strategy representation.

[2]I also performed experiments using *all* 50 joint rewards to refine the optimal collaborator estimate, and the results improve further — all methods in combination with biasing are able to find the global optimum in most cases.

Table 4.7: Percentage of CCEA runs that converged to global optimum, Two Peaks domain with mixed strategy representation

	Discretization Level (Number of Actions)			
	16	32	64	128
Traditional	0%	0%	0%	0%
Biased Traditional	100%	100%	100%	100%
Spatial	0%	0%	0%	0%
Biased Spatial	100%	100%	100%	100%
Spatial2	0%	0%	0%	0%
Biased Spatial2	100%	100%	100%	100%

Table 4.8: Percentage of CCEA runs that converged to global optimum, Rosenbrock domain with mixed strategy representation

	Discretization Level (Number of Actions)			
	16	32	64	128
Traditional	62%	12%	0%	0%
Biased Traditional	100%	100%	99%	82.9%
Spatial	93%	38%	3%	0%
Biased Spatial	100%	100%	100%	84.3%
Spatial2	85%	25%	1%	0%
Biased Spatial2	100%	100%	100%	79.8%

Results Summary The results suggest that the mixed strategy representation induces a significantly more complex search space than the pure strategy representation: mixed strategies usually have a non-zero probability of exploring different actions that may incur penalties. For this reason, I argue that the slope around the optimal peak has an abrupt gradient that may explain the decrease in performance. Consistent with the previous experiments involving the pure strategy representation, the results indicate that biasing never decreases the performance of a method, but it rather improves the performance in most situations.

Results Specifics Tables 4.5–4.8 present the percentages of runs that converged to the global optimum when using the mixed strategy representation in the Climb, Penalty, and the discretized Two Peaks and Rosenbrock domains. As the evaluation of an individual is averaged over 50 interactions, I consider that a run converged to the global optimum if the fitness of the best individuals (one per population) in the last generation is within 10% of the value of the global optimum — to exceed this threshold, each of the mixed strategies should have probability close to 1 for picking the action corresponding to the global optimum, as the joint reward for any other pair of actions is less than this threshold.

In the Climb domain, both Spatial and Spatial2 significantly outperform Traditional. However, enhancing each of the three methods with biasing results in convergence to the global optimum in almost every run. The Penalty domain is again easier then Climb — most runs find the global optimum, with biasing and without it.

The Two Peaks domain is consistently too difficult for either Traditional, Spatial and Spatial2, but all of them find the global optima in 100% of the runs when in combination with biasing. The Rosenbrock domain is relatively easier for coevolution, especially at low discretization levels. Traditional is again significantly worse than Spatial2, which in turn is significantly worse than Spatial. However, the performance of all methods is significantly superior when in combination with biasing.

Chapter 5: The Informative Cooperative Coevolutionary Algorithm

In Chapter 3, I provided formal proofs that the drift of concurrent learners away from optima is caused by poor estimations for the quality of actions, and I showed that ignoring certain reward information may be helpful to counter this drift. Chapter 4 argued that such an approach might involve significant computational resources in order to provide certain convergence guarantees, and it proposed an approach to reduce these requirements by using information that the learners have observed in the past to bias the search. This biasing technique was proven useful in simple problem domains, but further research is still required to analyze how the biasing information might be approximated in complex problem domains.

Observe that all these approaches aimed to improve the learners' estimations for the quality of agent behaviors. This chapter focuses instead on a class of simpler estimations that can be computed quickly. This research relies on the observation that certain concurrent learning algorithms are guided by the ranking of candidate solutions, and not by the absolute quality values assigned to them. For example, cooperative coevolutionary algorithms employing tournament selection use the fitness value only to obtain a relative ranking of individuals in the population. When comparing individual x_1 with individual x_2, the results are identical if their fitnesses are 100 and 50, respectively, or if their fitnesses are 2 and 1.9 respectively. As a consequence, these concurrent learning algorithms might have good performance even when working with simpler estimations for the quality of individuals, as long as the relative ranking of individuals is preserved. These simpler estimations might however be computed with a significantly lower computational cost: instead of approximating ideal collaborators for each individual, the algorithm might merely require a few collaborators to rank those individuals.

This chapter introduces the informative cooperative coevolutionary algorithm (iCCEA), which puts special emphasis on "informative" individuals — those collaborators that reveal useful ranking relations among the individuals in the other population. The algorithm strives to select the fewest such individuals to allow the search to progress at a faster pace. I describe iCCEA next. I then compare it against a related algorithm, pCCEA (Bucci and Pollack, 2004), and also against three ordinary cooperative coevolutionary algorithms. Overall, the results indicate that iCCEA significantly outperforms its competitors in most two-population test problems that are employed in these experiments.

5.1 Description of iCCEA

iCCEA maintains an archive of good collaboration choices for each of the populations. Because tournament selection is used, the idea behind iCCEA is to intelligently identify a small archive of collaborators which produce the same rank-ordering of fitnesses among individuals in the other population as these individuals would receive were they tested with the full population of collaborators. Collaborators in the archive each helped *some* individual receive its highest reward, and thus provide information about an "interesting" part of the space. iCCEA further reduces the archive size by taking advantage of the ranked nature of tournament selection: if a collaborator improves an individual, but not enough to change its rank (and thus its evolutionary viability), it is not considered for the archive. Further, iCCEA maintains a degree of diversity in the archive by ensuring that it does not contain individuals whose maximal collaborations were too close to one another in solution space. The algorithm promotes this archive to the next generation, and breeds children from parents selected via tournament selection to fill in the rest of the new population.

The size of the archive is therefore an important factor. On one side, larger archives might provide valuable information about the complexity of the search space. On the other side, iCCEA assumes that individuals are evaluated when in combination with each member of the other population's archive. As such, large archives may incur significant computational requirements. Additionally, the archive occupies a share of the population: fewer children are bred into the new population if the archive is large. It follows that large archives might interfere with the search for better solutions. It is also worth noting that iCCEA with an archive size of 1 reduces to a common evaluation approach for CCEAs: use the best individual (from the previous generation) plus some individuals chosen at random from the other population.

iCCEA performs the evaluation of the populations and the breeding of the next generations differently from other cooperative coevolutionary algorithms. The details for each of these procedures follows.

Evaluation At the beginning of a run, the archive ($Archive_p$) of each population p is simply set to the population itself. This means that in the first generation, each individual of a population will be evaluated against all the individuals in the other population (doing full mixing). This is expensive, but it provides good initial information before the archive mechanism kicks in.

In general, evaluation is in two parts. First, individuals are evaluated against the other population's archive members. Second, if more evaluations are desired (if the number of evaluations per individual has not yet reached *MaxEvals*), individuals are repeatedly paired off with individuals in the other population and evaluated together. The pseudocode for the evaluation process is:

Procedure iCCEA-Evaluation
Parameter $MaxEvals$: maximum evaluations per individual
for each population p **do**
  $p' =$ other population than p
  for each individual i in p **do**
    for each individual j in p' **do**
      $F_i^j = -\infty$
    end for
  end for
  for each individual i in p **do**
    for all individuals a in $Archive_{p'}$ **do**
      $F_i^a = Reward_p(i,a)$
      $F_a^i = Reward_{p'}(i,a)$
    end for
  end for
end for
$MaxArchive = \max_p |Archive_p|$
for $\max(0, MaxEvals - MaxArchive)$ times **do**
  for each population p **do**
    shuffle p
  end for
  for $i = 1$ to $PopSize$ **do**
    a_1 = individual with index i in population p_1
    b_1 = individual with index i in population p_2
    $F_{a_1}^{b_1} = Reward_{p_1}(a_1, b_1)$
    $F_{b_1}^{a_1} = Reward_{p_2}(a_1, b_1)$
  end for
end for
for each population p **do**
  $p' =$ other population than p
  for each individual i in p **do**
    $Fitness(i) = \max_{j \in p'} F_i^j$
  end for
end for

Breeding and Archive Selection The breeding and population reassembly phase of iCCEA proceeds similarly to the one of pCCEA: the archive members are selected from the old population and are copied directly into the new population, and the remainder of the new population is filled with children bred using standard EA mechanisms applied to the old population (including the old archive). The entire previous population (including the archive) competes for breeding. The pseudocode is straightforward:

Procedure iCCEA-Breeding
for each population **do**
  select its new archive with iCCEA-Archive-Selection
  copy the archive into the new population
  fill the rest of the new population using standard EA breeding
end for

Archive selection is intended to select those individuals that reveal features of the search space that are useful to the other population. Specifically, iCCEA aims to select a minimal archive of individuals from population p such that when assessing the fitness of individuals in the other population p', testing them against the full set of individuals in p would not change their rank ordering beyond just testing them against the individuals in p's archive. The hope is that this archive would provide an accurate evaluation and ranking of the individuals in p' in the next generation as well.

This archive should be minimal because each individual in p' is being evaluated against every single individual in p's archive. Large archives imply an $O(n^2)$ evaluation cost per generation. Therefore individuals are added to the archive only if they cause individuals in the other population to improve significantly enough so as to effect the ranking. Of the various individuals which change this ranking, iCCEA selects the ones which do so by raising fitnesses to the highest levels.

The archive selection process starts from the empty set and proceeds iteratively. For each individual i not yet in the archive, and for each individual x in the other population, iCCEA first computes $Fit1_x$, the fitness of x if evaluated in combination with all individuals currently in the archive. This is followed by the computation of $Fit2_x^i$, the fitness of x if i were part of the archive. Note that $Fit2_x^i \geq Fit1_x$. iCCEA uses three criteria to determine whether i should be added to the archive. First: does there exist a pair of individuals x and y in the other population whose relative ranks change when i is added to the archive? That is, is it true that $\exists x,y : Fit1_x \leq Fit1_y$ and $Fit2_x^i > Fit2_y^i$? Second, is the individual "eligible"? An individual i is ineligible if there already exists an individual in the archive whose joint solution (with the collaborator whose rank it significantly improved) is sufficiently "close" in genotype space to the joint solution formed by i and the collaborator x whose rank i improves most. Third, and finally, of all individuals i that meet the first two criteria, the one that changed the ranking by raising the fitness of its x to the highest level is added to the archive. Note that the first individual to be selected for the archive is always the one with the highest fitness. The naive algorithmic description below was chosen for clarity but is $O(n^3)$. It is relatively straightforward to design algorithms that take advantage of the relatively small archive size in order to significantly reduce the asymptotic complexity of the iCCEA-Archive-Selection procedure.

Procedure iCCEA-Archive-Selection
Parameter $MinDist$: minimum distance requirement
for each population p **do**
  $p' =$ other population than p
  $Archive_p = \emptyset$
  $Ineligible_p = \emptyset$
  $shouldExit = false$
  repeat
    for each individual i in $p - Archive_p$ **do**
      for each individual x in p' **do**
        $Fit1_x = \max_{j \in Archive_p} F_x^j$
        $Fit2_x^i = \max(Fit1_x, F_x^i)$
      end for
      for each individual x in p' **do**
        for each individual y in p' **do**

$$Fit3_{x,y}^i = \begin{cases} Fit2_x^i & \text{if } Fit1_x \leq Fit1_y \text{ and } Fit2_x^i > Fit2_y^i \\ -\infty & \text{otherwise} \end{cases}$$

        end for
      end for
    end for
    for each individual i in $p - (Archive_p \bigcup Ineligible_p)$ **do**
      $MaxFit_i = \max_{x,y \in p'} Fit3_{x,y}^i$
    end for
    $a = \arg\max_i \ MaxFit_i$
    if $MaxFit_a \neq -\infty$ **then**
      select x such that $MaxFit_a = max_y Fit3_{xy}^a$
      if $min_{i \in Archive_p} distance(\langle a, x \rangle, \langle i, Collaborator_i \rangle) < MinDist$ **then**
        $Ineligible_p = Ineligible_p \bigcup \{a\}$
      else
        $Archive_p = Archive_p \bigcup \{a\}$
        $Collaborator_a = x$
      end if
    else
      $shouldExit = true$
    end if
  until $shouldExit$
end for

5.2 Experiments

This section compares the performance of iCCEA against that of several state-of-the-art cooperative coevolutionary algorithms, which are detailed next.

5.2.1 Competing Techniques

The first algorithm involved in the experiments is iCCEA, introduced in Section 5.1. Parameter settings involve $MaxEvals = 5$, and one of multiple values for $MinDist$: 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.5, 0.75, and 1.0. For simplicity, iCCEA-m refers to the run of the algorithm with $MinDist = m$.

Second, cCCEA evaluates an individual against *every single collaborator* in the other population. The fitness of an individual is set to the maximum of these evaluations. As shown in Section 3, cCCEA will converge to the unique global optimum if the population size is sufficiently large.

The third algorithm employed in the comparison is pCCEA (Bucci and Pollack, 2004). pCCEA employs an elitist archive in each population; this archive maintains individuals in the population which, as collaborators, were effective in assisting *some* individual from the other population. The algorithm uses a Pareto domination relationship among individuals: individual i dominates individual j if, for any collaborator x, i's reward with x is never worse than j's reward with x, and there exists at least one collaborator y with which i's reward is higher than j's reward. At each generation, pCCEA first evaluates every individual with every possible collaborator in the other population, then assembles the set of individuals dominated by no one else. This set forms the archive, and is automatically copied to the next generation. pCCEA then fills the rest of the next generation by iteratively selecting two individuals and comparing them. If neither dominates the other, both are selected for breeding. If one is dominated by the other, only the dominating individual is selected.

It is important to note some possible drawbacks of this algorithm. First, pCCEA does not use the archive to define a small set of collaborators. Instead, it uses the archive only to promote certain collaborators to the next generation where they, along with new population members, will be used to evaluate individuals. Thus it maintains a set of "informative" collaborators but does not try to use that set to minimize the number of necessary evaluations. This is both positive and negative, as it requires many evaluations, but provides diversity in evaluations by adding newly-generated collaborators to the evaluation mix. Second, pCCEA does not use fitness to select individuals. An individual may be selected if it collaborates better than others with even a single partner from the other population, even if that collaboration result is very poor. Third, pCCEA only compares individuals to other individuals and not to the group. Thus an individual may be selected for the archive if it beats every individual in *some* collaboration scenario even if it is never the best choice for *any* collaboration scenario over all the population. Both of these features may tend to slow evaluation as more individuals are promoted to the next generation. Fourth, and importantly, the results of the experiments (described next) indicate that pCCEA's archive tends to converge to the Pareto frontier, which may be unfortunately infinite in even simple cooperative multiagent

domains. In such situations, the archive will rapidly consume the entire population, and search will stagnate.

Fourth, rCCEA assesses an individual's fitness as the maximum reward over evaluations with six individuals: five random individuals, plus the fittest individual in the collaborator population from the previous generation.

Fifth, rCCEA-Perm chooses random collaborators by shuffling each population and then pairing individuals in the populations. In rCCEA-PERM, both the individual and its collaborator count the evaluation towards their tally, and so the number of evaluations per generation is almost half that of rCCEA.

5.2.2 Experimental Setup

The experiments involve three different types of problem domains. The MTQ problem family, Rastrigin, and Griewangk problems are multimodal problems. The OneRidge, Rosenbrock, and Booth problems are unimodal but have diagonal ridges that make the search more difficult. Finally, the SMTQ problem family is both multimodal and has diagonal ridges. Full descriptions for some of these domains can be found in (Popovici and Jong, 2004; Whitley *et al.*, 1996).

Some of the problems are inverted from their original form to be used as maximization problems. All experiments employ two populations of 32 individuals each. Individuals encode real-valued numbers between 0 and 1. Parents are chosen via tournament selection of size 2, and children are created via mutation by adding to the parent's value a number randomly generated from a normal distribution with mean 0 and standard deviation 0.01. The fittest individual is copied automatically to the next population for cCCEA, rCCEA, and rCCEA-Perm. As the algorithms require different numbers of evaluations per generation, each of them is given a budget of 51200 evaluations (plus or minus a few extra to complete the last generation). This results in the evolution lasting 50 generations for pCCEA and cCCEA, 134 generations for rCCEA, 240 generations for rCCEA-Perm, and a variable number of generations for iCCEA.

The ECJ package (Luke, 2005) is used for the experiments. Each experiment is repeated 250 times for statistical significance. Given that results often do not have a normal distribution, the 95% confidence interval for the median of the results is reported (as recommended in (Lehmann, 1975)). Statistical significance is verified via non-parametric pairwise t-tests. Each such test is performed at a 99.999% confidence level (approximated via the Bonferroni inequality) to provide an overall 95% confidence level for all tests.

5.2.3 Experiments in Multimodal Domains

Multimodal domains are challenging for cooperative coevolution because both populations need to coordinate to identify and explore the highest of multiple peaks. I start with an empirical

Table 5.1: 95% confidence interval for the median performance of the methods in the MTQ domain instance with $H_1 = 50$.

Method	Lower Bound	Median	Upper Bound
pCCEA	149.7024	149.79412	149.8466
cCCEA	149.99997	149.99997	149.99998
iCCEA-0.0	150	150	150
iCCEA-0.05	150	150	150
iCCEA-0.1	150	150	150
iCCEA-0.15	150	150	150
iCCEA-0.2	150	150	150
iCCEA-0.25	150	150	150
iCCEA-0.5	150	150	150
iCCEA-0.75	150	150	150
iCCEA-1.0	150	150	150
rCCEA	50	50	149.99998
rCCEA-Perm	50	50	150

comparison in an instance of the MTQ class (see Section 4.3.1 for details). Following, I analyze the performance of the methods in two traditional optimization benchmark problems: Griewangk and Rastrigin. To summarize, the results indicate that iCCEA performs best, especially for relatively small values of *MinDist* (usually for $MinDist \leq 0.5$). At the other extreme, pCCEA appears to have the worst performance. The specifics of these experiments follow.

The MTQ Domain Instance

The MTQ domain instance employed in these experiments uses the following settings: $S_1 = \frac{16}{10}$, $X_1 = \frac{3}{4}$, $Y_1 = \frac{3}{4}$, $H_2 = 150$, $S_2 = \frac{1}{32}$, $X_2 = \frac{1}{4}$, $Y_2 = \frac{1}{4}$. H_1 is varied across experiments, but it is always less than 125.

First, H_1 is set to 50 to instantiate an MTQ problem with a wide difference between the heights of the two peaks. In this case, coevolution might have difficulties finding the global optimum primarily because the optimum's coverage is significantly smaller than that of the suboptimal peak. Table 5.1 presents the performance of the methods in the MTQ domain. The average run of iCCEA lasts about 262 generations for each setting of *MinDist*. Observe that iCCEA converges to the optimal answer in most runs. Non-parametric statistical tests indicate that iCCEA is better than all other methods for any setting of *MinDist*.

Although it appears relatively simple, MTQ using such settings is fairly difficult to optimize: the probability that a random sample exceeds a function value of 149.99 can be computed for $H1$ is set to 50 as $\pi \times (1 - \frac{149.99}{150}) \times \frac{S_2}{16} = 0.0000004090615$. Given 51200 random

Table 5.2: 95% confidence interval for the median performance of the methods in the MTQ domain instance with $H_1 = 125$.

Method	Lower Bound	Median	Upper Bound
pCCEA	148.97783	149.4233	149.65022
cCCEA	149.99985	149.99991	149.99995
iCCEA-0.0	150	150	150
iCCEA-0.05	150	150	150
iCCEA-0.1	150	150	150
iCCEA-0.15	150	150	150
iCCEA-0.2	150	150	150
iCCEA-0.25	150	150	150
iCCEA-0.5	150	150	150
iCCEA-0.75	125	125	149.99998
iCCEA-1.0	125	125	125
rCCEA	125	125	125
rCCEA-Perm	125	125	125

samples (approximately the number of evaluations performed during a typical evolutionary run), the probability that one of them exceeds a function value of 149.99 is $1 - (1 - 0.0000004090615)^{51200} = 0.02072615$. Compare this result to iCCEA, which finds better approximations of the global optimum in more than 50% of the runs. iCCEA thus outperforms random search in this domain, and implicitly distributed learning algorithms relying on random search, such as those proposed in (Brafman and Tennenholtz, 2002).

Similar to the experiments in (Bucci and Pollack, 2004), H_1 is also set to 125 to create a more deceiving domain instance: the individuals on the suboptimal peak have higher fitness and are thus more likely to be selected. This also reduces the size of the area where the optimal peak is superior to the suboptimal one, making the problem harder than when $H_1 = 50$.

The results (summarized in Table 5.2) indicate that iCCEA with *MinDist* smaller or equal to 0.5 is the top tier performer, and it significantly outperforms the other methods. The second tier consists of cCCEA, pCCEA, and iCCEA with *MinDist* greater than 0.5. Last, rCCEA and rCCEA-Perm have significantly worse results than all other methods. The difference in heights leads to a slight increase in archive size for iCCEA: the average run of iCCEA lasts around 253 generations, down from an average of 262 generations when $H_1 = 50$ (this decrease is significant at the 99.999% confidence level).

Table 5.3: 95% confidence interval for the median performance of the methods in the Griewangk domain

Method	Lower Bound	Median	Upper Bound
pCCEA	-0.009099687	-0.008524607	-0.008080815
cCCEA	-2.3027173e-08	-1.6132691e-08	-1.06091615e-08
iCCEA-0.0	-1.2595542e-09	-9.4223985e-10	-7.1052053e-10
iCCEA-0.05	-6.868509e-10	-5.39539185e-10	-4.6161386e-10
iCCEA-0.1	-7.2518436e-10	-5.51555305e-10	-4.5181736e-10
iCCEA-0.15	-7.856029e-10	-6.1868948e-10	-4.777243e-10
iCCEA-0.2	-7.931591e-10	-6.166706e-10	-4.8393356e-10
iCCEA-0.25	-8.377967e-10	-6.9733245e-10	-5.179104e-10
iCCEA-0.5	-7.4639095e-10	-5.629196e-10	-4.5529336e-10
iCCEA-0.75	-0.0073960405	-2.74472505e-09	-1.2845356e-09
iCCEA-1.0	-0.0073960405	-2.56566515e-09	-8.832689e-10
rCCEA	-0.007396041	-0.0073960405	-0.0073960405
rCCEA-Perm	-0.0073960423	-0.0073960414	-0.007396041

The Griewangk Domain

The Griewangk function is defined as

$$Griewangk(x,y) \leftarrow -1 - \frac{\bar{x}^2}{4000} - \frac{\bar{y}^2}{4000} + \cos(\bar{x})\cos(\frac{\bar{y}}{\sqrt{2}})$$

where $\bar{x} = 10.24x - 5.12$, $\bar{y} = 10.24y - 5.12$, and x and y are encoded in individuals as real-valued numbers between 0 and 1. The function has a maximum value equal to 0 for $\bar{x} = \bar{y} = 0$, and several suboptimal peaks surrounding it.

Table 5.3 summarizes the results of the experiments in this problem domain. iCCEA performs significantly better than all other methods when $MinDist \leq 0.5$. Of these settings, $MinDist = 0.0$ appears slightly worse (with confidence around 99%, lower than the desired 99.999%). This is because iCCEA-0.0 has significantly higher archive sizes, yielding a lower numbers of generations per run (161 generations for $MinDist = 0.0$, compared to 233 generations for $MinDist = 0.2$). The second tier of performers consists of cCCEA, rCCEA, rCCEA-Perm, iCCEA-0.75 and iCCEA-1.0. Finally, pCCEA is significantly worse than all other methods.

The Rastrigin Domain

The Rastrigin function is defined as

$$Rastrigin(x,y) \leftarrow -20 - \bar{x}^2 + 10\cos(2\pi\bar{x}) - \bar{y}^2 + 10\cos(2\pi\bar{y})$$

Table 5.4: 95% confidence interval for the median performance of the methods in the Rastrigin domain

Method	Lower Bound	Median	Upper Bound
pCCEA	-0.0001342016	-4.20476935e-05	-2.75237e-05
cCCEA	-3.373491e-05	-1.92212125e-05	-1.20936875e-05
iCCEA-0.0	-0.99495906	-7.32915735e-07	-3.9109466e-07
iCCEA-0.05	-0.99495906	-1.44978885e-06	-4.859953e-07
iCCEA-0.1	-1.5460682e-06	-6.5733175e-07	-3.8390752e-07
iCCEA-0.15	-0.99495906	-1.46234045e-06	-7.4304603e-07
iCCEA-0.2	-0.99495906	-9.1238677e-07	-4.893715e-07
iCCEA-0.25	-0.99495906	-8.07612e-07	-4.2249255e-07
iCCEA-0.5	-9.465858e-07	-5.692597e-07	-3.5596943e-07
iCCEA-0.75	-0.99495906	-8.1733685e-07	-5.13917e-07
iCCEA-1.0	-2.7152691e-06	-6.213839e-07	-4.274801e-07
rCCEA	-0.9949591	-5.13458285e-06	-2.3090972e-06
rCCEA-Perm	-1.700194e-06	-9.64721415e-07	-5.1975854e-07

where $\bar{x} = 10.24x - 5.12$, $\bar{y} = 10.24y - 5.12$, and x and y encoded in individuals as real-valued numbers between 0 and 1. The function has a maximum value equal to 0 for $\bar{x} = \bar{y} = 0$, and many suboptimal peaks surrounding it.

Table 5.4 summarizes the results in this domain. The first tier of performers consists of iCCEA (for any value of *MinDist*) and rCCEA-Perm, which are not significantly worse than any other method. rCCEA is worse then iCCEA for four settings of *MinDist*. Last, cCCEA and pCCEA are dominated by ten and eleven other methods, respectively. Surprisingly, iCCEA has about 263 generations per run, indicating that the archive size is very low. This effect is due to the fact that the highly multimodal Rastrigin function describes essentially a quadratic curve ($1 - \bar{x}^2 - \bar{y}^2$), accompanied by a plethora of smaller peaks. As a consequence, iCCEA appears to exploit the underlying quadratic function to achieve a small archive at each generation.

5.2.4 Experiments in Domains with Diagonal Ridges

Diagonal ridges create additional difficulties for concurrent learning: progress can be achieved only when both learners simultaneously explore actions such that the joint actions are along the ridge. The Rosenbrock and Booth benchmark optimization functions create non-trivial domains with diagonal ridges. The OneRidge domain, however, takes these difficulties to the limit: each point along the ridge is in fact a Nash equilibrium. As a consequence, there are an infinite number of suboptimal Nash equilibria where the concurrent learners can get stuck: suboptimal Nash equilibria can distort the fitness estimations by completely eliminating all information about the global optimum (Panait and Luke, 2005b).

Table 5.5: 95% confidence interval for the median performance of the methods in the OneRidge domain

Method	Lower Bound	Median	Upper Bound
pCCEA	1.47526	1.4772663	1.4805671
cCCEA	1.9103878	1.91416505	1.9197007
iCCEA-0.0	1.5467398	1.5517363	1.5576606
iCCEA-0.05	2	2	2
iCCEA-0.1	2	2	2
iCCEA-0.15	2	2	2
iCCEA-0.2	2	2	2
iCCEA-0.25	2	2	2
iCCEA-0.5	2	2	2
iCCEA-0.75	2	2	2
iCCEA-1.0	2	2	2
rCCEA	2	2	2
rCCEA-Perm	2	2	2

The results of experiments (detailed next) indicate that restrictions on the diversity of iCCEA's archive are crucial for good performance in these three domains. Otherwise, the archive usually becomes too large and interferes with the exploration of the space.

The OneRidge Domain

The OneRidge problem maximizes the function

$$OneRidge(x,y) \leftarrow 1 + 2\min(x,y) - \max(x,y)$$

where x and y range between 0 and 1. OneRidge is particularly difficult for concurrent learners because it contains a very large number of Nash equilibria: for any value v between 0 and 1, (v,v) is a Nash equilibrium. This implies that for any Nash equilibrium (except for the global optimum $(1,1)$) there are an infinite number of better Nash equilibria that are infinitesimally close. Unfortunately, both populations must simultaneously change to a new equilibrium in order for solutions to improve. To emphasize the algorithms' capacity to follow this ridge, the populations are randomly initialized such that all individuals are smaller than 0.5.

Table 5.5 summarizes the results. rCCEA and rCCEA-Perm find the global optimum in every single run. iCCEA achieves identical optimal performance when *MinDist* is greater than 0. Among the remaining methods, cCCEA is significantly better, followed by iCCEA-0.0, then pCCEA. Increases in *MinDist* put additional constraints on archive inclusion, resulting in longer runs

Table 5.6: 95% confidence interval for the median performance of the methods in the Rosenbrock domain

Method	Lower Bound	Median	Upper Bound
pCCEA	-0.1084212	-0.086038075	-0.07215236
cCCEA	-5.4309703e-06	-3.97336365e-06	-3.095801e-06
iCCEA-0.0	-7.0464353e-06	-5.6810313e-06	-4.386692e-06
iCCEA-0.05	-2.9603286e-06	-2.0364071e-06	-1.484208e-06
iCCEA-0.1	-1.5654674e-06	-1.05892685e-06	-6.922747e-07
iCCEA-0.15	-9.347807e-07	-7.36914e-07	-5.559006e-07
iCCEA-0.2	-7.9747747e-07	-5.1362627e-07	-4.3587843e-07
iCCEA-0.25	-1.1045466e-06	-8.6313633e-07	-5.565543e-07
iCCEA-0.5	-1.5428313e-06	-1.1737227e-06	-7.397234e-07
iCCEA-0.75	-9.623004e-07	-6.5651085e-07	-4.610377e-07
iCCEA-1.0	-9.135398e-07	-6.48494135e-07	-4.043417e-07
rCCEA	-1.45301265e-05	-1.1222902e-05	-8.5360825e-06
rCCEA-Perm	-5.950367e-07	-4.9335307e-07	-4.3179807e-07

(from 134 generations for iCCEA-0.0, to 247 generations for iCCEA-0.05, to 264 generations for iCCEA-1.0).

Note that iCCEA-0.0 is still significantly better than pCCEA, despite the fact that both algorithms lack any constraints on the archive size. This difference stems from two factors. First, iCCEA has more restrictions when constructing the archive than pCCEA does, usually resulting in smaller archive sizes (in fact, pCCEA's archive covers the whole population within a few generations, while iCCEA's archive has about half that size). Second, iCCEA tends to perform fewer evaluations per generation, allowing the search to last longer for the same computational budget.

The Rosenbrock Domain

The performance of the methods in the Rosenbrock domain are examined next. This problem domain is defined by

$$Rosenbrock(x,y) \leftarrow -(100(\bar{x}^2 - \bar{y})^2 + (1-\bar{x})^2)$$

where $\bar{x} = 10.24x - 5.12$, $\bar{y} = 10.24y - 5.12$, and x and y are encoded in individuals as real-valued numbers between 0 and 1. Similarly to OneRidge, Rosenbrock is particularly difficult for concurrent learners because optimization requires both populations to simultaneously follow a narrow ridge up-the-hill; in addition, the ridge in the Rosenbrock domain has a non-linear shape.

The results of the methods are presented in Table 5.6. The nonparametric statistical tests indicate that rCCEA-Perm and iCCEA with $MinDist \geq 0.1$ are the top-tier performers. The second tier consists of cCCEA, iCCEA-0.0, and iCCEA-0.05. rCCEA and pCCEA have the worst performance;

Table 5.7: 95% confidence interval for the median performance of the methods in the Booth domain

Method	Lower Bound	Median	Upper Bound
pCCEA	-2.2757607e-05	-1.78912995e-05	-1.3247128e-05
cCCEA	-4.1714014e-07	-3.2144021e-07	-2.7121857e-07
iCCEA-0.0	-0.00010195026	-8.06368125e-05	-5.630301e-05
iCCEA-0.05	-5.884756e-07	-4.18359055e-07	-3.5493673e-07
iCCEA-0.1	-3.606497e-07	-2.82521425e-07	-2.2286326e-07
iCCEA-0.15	-4.869323e-07	-3.895216e-07	-3.1035532e-07
iCCEA-0.2	-3.2798286e-07	-2.77684885e-07	-2.2034128e-07
iCCEA-0.25	-4.5106958e-07	-3.7739077e-07	-2.7794223e-07
iCCEA-0.5	-3.0225107e-07	-2.44586035e-07	-2.1388162e-07
iCCEA-0.75	-3.2224608e-07	-2.5254693e-07	-1.9835213e-07
iCCEA-1.0	-4.3931948e-07	-3.6072215e-07	-2.66255e-07
rCCEA	-1.5708204e-07	-1.31100655e-07	-1.02842776e-07
rCCEA-Perm	-4.172578e-07	-2.9863297e-07	-2.3673356e-07

in particular, pCCEA is significantly worse than all other methods. The difference among the iCCEA variations again highlights the importance of putting some restrictions on minimum distance between members of the archive: iCCEA-0.0 evaluates populations with a large archive, resulting in around 74 generations per run, while iCCEA-0.2 uses a moderately-sized archive that allows it 254 generations per run.

The Booth Domain

Similarly to the OneRidge domain, the Booth problem also creates a ridge that can be pursued primarily by simultaneous exploration by both populations. The Booth function is defined as:

$$Booth(x,y) \leftarrow -(\bar{x}+2\bar{y}-7)^2-(2\bar{x}+\bar{y}-5)^2$$

where $\bar{x} = 10.24x - 5.12$, $\bar{y} = 10.24y - 5.12$, and x and y encoded in individuals as real-valued numbers between 0 and 1. The Booth function essentially creates a squashed peak along the diagonal axis.

The results, summarized in Table 5.7, indicate that rCCEA significantly outperforms all other methods. iCCEA falls in the second tier for $MinDist > 0$, together with cCCEA and rCCEA-Perm. Last, pCCEA is worse than every other method but iCCEA-0.0 (which is significantly worse than everything else). As in the OneRidge and Rosenbrock domain, iCCEA-0.0 uses larger archives and exhausts its computational budget in fewer generations (around 160), while even small values of *MinDist* reduces the archive size significantly (runs lasts for about 261 generations when $MinDist > 0$). I am still exploring the causes for rCCEA's excellent performance in this problem

Table 5.8: 95% confidence interval for the median performance of the methods in the SMTQ domain instance with $H_1 = 50$.

Method	Lower Bound	Median	Upper Bound
pCCEA	130.5234	135.36403	139.55858
cCCEA	149.99992	149.99994	149.99995
iCCEA-0.0	149.99994	149.99994	149.99995
iCCEA-0.05	149.99998	149.99998	149.99998
iCCEA-0.1	149.99998	149.99998	149.99998
iCCEA-0.15	149.99998	149.99998	150
iCCEA-0.2	149.99998	149.99998	149.99998
iCCEA-0.25	149.99998	149.99998	150
iCCEA-0.5	149.99998	149.99998	150
iCCEA-0.75	149.99998	149.99998	149.99998
iCCEA-1.0	149.99998	149.99998	149.99998
rCCEA	50	149.99981	149.99994
rCCEA-Perm	149.99995	149.999975	149.99998

domain; I currently attribute its success to a good match of rCCEA's settings to this problem domain, compared to iCCEA's efforts to adjust the archive size to proper settings.

5.2.5 Experiments in Multimodal Domains with Diagonal Ridges

Finally, I create a new class of problem domains that combines the difficulties associated with both multimodal search spaces and domains with diagonal ridges. To this end, I start with the MTQ class of problems, and change each peak to have an ellipsoid shape aligned diagonally. This leads to diagonal ridges towards the two optima, similar to the unique ridge in the Booth domain. The new class of problems, termed SMTQ, is defined as

$$\text{SMTQ}(x,y) \leftarrow \max \begin{cases} H_1 \times (1 - \frac{32\left(x_1^r - X_1\right)^2}{S_1} - \frac{8\left(y_1^r - Y_1\right)^2}{S_1}) \\ H_2 \times (1 - \frac{32\left(x_2^r - X_2\right)^2}{S_2} - \frac{8\left(y_2^r - Y_2\right)^2}{S_2}) \end{cases}$$

Table 5.9: 95% confidence interval for the median performance of the methods in the SMTQ domain instance with $H_1 = 125$.

Method	Lower Bound	Median	Upper Bound
pCCEA	133.09122	137.50821	140.33017
cCCEA	149.99974	149.99985	149.9999
iCCEA-0.0	149.9999	149.99991	149.99995
iCCEA-0.05	149.99997	149.99998	149.99998
iCCEA-0.1	149.99997	149.99998	149.99998
iCCEA-0.15	149.99995	149.99997	149.99998
iCCEA-0.2	149.99997	149.99998	149.99998
iCCEA-0.25	149.99997	149.99998	149.99998
iCCEA-0.5	149.99997	149.99998	149.99998
iCCEA-0.75	149.99994	149.99997	149.99997
iCCEA-1.0	125	149.99995	149.99997
rCCEA	125	125	125
rCCEA-Perm	125	125	125

where x_1^r, y_1^r, x_2^r, and y_2^r are defined as

$$x_1^r = (x - X_1)\cos\frac{\pi}{4} + (y - Y_1)\sin\frac{\pi}{4} + X_1$$

$$y_1^r = (x - X_1)\cos\frac{\pi}{4} - (y - Y_1)\sin\frac{\pi}{4} + Y_1$$

$$x_2^r = (x - X_2)\cos\frac{\pi}{4} + (y - Y_2)\sin\frac{\pi}{4} + X_2$$

$$y_2^r = (x - X_2)\cos\frac{\pi}{4} - (y - Y_2)\sin\frac{\pi}{4} + Y_2$$

The same values for H_2, X_1, Y_1, X_2, Y_2, S_1, and S_2 are used as for the MTQ class.

Tables 5.8–5.9 present the results of the methods in the SMTQ domains for $H_1 = 50$ and $H_1 = 125$. iCCEA with $MinDist \in \{0.05, ..., 0.5\}$ performs significantly better than pCCEA, cCCEA, rCCEA, and rCCEA-Perm in both domains. This is expected given the results in the previous domains: no distance restrictions ($MinDist = 0.0$) leads to large archives that interfere with following ridges towards optima in domains such as in Section 5.2.4, while large values of *MinDist* tamper with iCCEA's attempts to search multiple peaks concurrently in multimodal domains (Section 5.2.3).

Chapter 6: The Lenient Multiagent Q-learning Algorithm

The research in Chapters 3–5 assumed that the joint reward information is perfectly accurate. There are, however, a large number of problem domains where this assumption does not hold. For example, learning agents in multirobot systems might only have access to noisy reward information due to their limited, inaccurate sensors.

This chapter proposes a new concurrent learning algorithm for domains involving stochastic rewards. I address in particular the more difficult (and also more realistic) case where agents receive the reward, but they do not know what actions their teammates have chosen. As a consequence, the reward received by an agent when selecting its actions is affected by two types of noise. The first type of noise occurs because an agent does not know its teammates' actions, although their choices have a direct impact on the reward observed by that agent. The second type of noise is due to the stochastic nature of the reward associated with each joint action. The learning agents have the difficult task of discerning between these two types of noise: the first one needs to be eliminated, while the second one is critical for convergence to the joint action with maximum expected reward.

Reinforcement learning techniques are particularly suited for stochastic domains. Section 1.2.2 presented Q-learning, a population reinforcement learning method, and Section 1.2.4 described a straightforward extension of Q-learning to multiagent domains. This extension (termed multiagent Q-learning) decomposes the utility tables associated with joint actions into simpler tables associated with the actions of each agent. These utilities are updated based on every reward that the agent observes. As shown in Section 3.3 and in general throughout Chapter 3, concurrent learners that use all rewards are very likely to converge to suboptimal solutions, while those that ignore lower rewards are more likely to converge to the global optimum in deterministic domains. In this chapter, I further that discussion with an extension for multiagent problems characterized by stochastic rewards. For simplicity, I assume that the environment has a single state, and I only focus on computing the utility of choosing different actions; this is similar to the analysis of multiagent Q-learning in (Claus and Boutilier, 1998; Lauer and Riedmiller, 2000; Kapetanakis and Kudenko, 2002a; Kapetanakis and Kudenko, 2002b).

Next, I describe the lenient multiagent Q-learning algorithm (or LMQ). The name emphasizes the strong similarity between the underlying workings of the algorithm and the notion of lenience as commonly occurring in collaborative learning among people. Consider the following example for an intuitive explanation of how LMQ works. Look back at the scenario involving John and David learning to play soccer, as described in Section 3.1 on page 38. At early stages of learning,

John pays attention to the fact that David can score occasionally as a result of his pass, and he shows lenience to other mistakes David might make. This is also a consequence of John hoping that David will improve as he learns more about playing soccer. As learning progresses, John should tend to prefer actions with higher expected reward. For example, he should prefer a pass that results in David scoring with 75% probability, over another pass from which David has only a 2% probability of scoring. This can be accomplished if John becomes more critical of his teammate's actions by ignoring fewer lower rewards. This is precisely the approach taken by the LMQ algorithm.

Section 6.2 compares the performance of LMQ against that of several other concurrent learning algorithms. The results indicate that LMQ has a superior rate of convergence to the global optimum, especially when the level of noise is high. In addition, LMQ requires less memory than the better of its competitors.

6.1 Description of LMQ

LMQ allows agents to exhibit a time-dependent level of lenience towards their teammates. It is based on the following idea: if an agent receives rewards $r_1, r_2, ..., r_k$ when choosing action a_1 at various times during the early stages of learning, the agent ignores most of these rewards and only updates the utility of a_1 based on the maximum of $r_1, r_2, ..., r_k$. The reason for this is that those rewards were obtained while the other learning agent selected (or, better said, explored) some actions $b_1, ..., b_k$, most of which it will ignore in the future due to their lower utilities. As both agents become more selective at choosing their actions, it is expected that they will each tend to primarily select a single action (the best one). At this point, each agent should update the utility of that action based on every reward it observes. This will lower the optimistic estimation of the utility for that action until it equals the mean reward obtained by the agents for that joint reward. If this mean reward becomes lower than the estimated utility of other actions, the agent will start to explore these other actions instead.

The algorithm is implemented as follows. LMQ always updates the utility of the action if the current reward exceeds the utility associated with that action. Otherwise, it uses a probabilistic approach: if the agent has not explored that action sufficiently, it should show lenience to its teammates and not update its policy; but if the agent has explored that action many times in the past, it should tend to be more critical and use the reward to lower the utility of that action. To this end, LMQ associates a temperature with each action: this temperature starts with a high value, and it is decreased every time the agent selects that action. As a consequence, the agents' time-dependent degree of lenience relies on these temperatures: if the temperature associated with an action is high, the agent is more lenient and ignores low rewards it receives for choosing that action. This initially results in an overoptimistic evaluation of the utility for an action, leading

to the possibility of agents temporarily preferring suboptimal actions. However, the utilities of such actions will decrease with time due to agents' fading lenience to one another, and the agents will become more likely to choose the optimal actions. There is also a small (0.01) probability of ignoring small rewards at all times: I found this to work well because the agents have non-zero probabilities of selecting an action at each time step. It might be helpful to correlate this probability with the agents' exploration policies, and I plan future research on this issue.

Aside from these enhancements, the algorithm follows a traditional Q-learning approach: the action selection uses the Boltzman distribution, and the utility is updated based in part on the reward currently received for an action. On top of the memory required to store the utility table, the proposed algorithm needs storage to encode a temperature for every action. Note that this is about the same as the memory requirement of OMQ, and half that of FMQ, two multiagent Q-learning competitors detailed in Section 6.2.1. The pseudocode for the LMQ algorithm (that each concurrent learner employs) follows.

Algorithm LMQ
Parameters
τ_{max}: maximum temperature
α: temperature multiplication coefficient
β: exponent coefficient
ν: temperature decay coefficient
λ: learning rate
N: number of actions
Initialization
for each action a **do**
$U_i =$ random value between 0 and 0.001
$\tau_i = \tau_{max}$
end for
Learning Algorithm
loop {at each time step}
{action selection}
$\tau_{min} = 10^{-6} + \min_{i=1}^{N} \tau_i$
$W_i = \frac{e^{\frac{U_i}{\tau_{min}}}}{\sum_{j=1}^{N} e^{\frac{U_j}{\tau_{min}}}}$
use probability distribution W_1, ..., W_N to select action i
$\tau_i = \tau_i \times \nu$
{utility update}
perform action i and observe reward r
$RandVal =$ random value between 0 and 1
if $(U_i \leq r)$ or $\left(RandVal < 10^{-2} + \beta^{-\alpha \times \tau_i}\right)$ **then**
$U_i = \lambda \times U_i + (1 - \lambda) \times r$
end if
end loop

6.2 Experiments & Results

This section compares the performance of the LMQ algorithm against that of several state-of-the-art multiagent Q-learning algorithms, which are detailed next.

6.2.1 Competing Techniques

Aside from LMQ, the experiments include three other existing algorithms. First, I use the performance of a straightforward multiagent Q-learning algorithm (Section 1.2.4) as a standard benchmark. This algorithm is termed MQ in the remainder of this chapter. The pseudocode for MQ follows.

Algorithm MQ
loop {at each time step x}
{action selection}
$\tau = e^{-sx} \times \tau_{max} + 1$
$W_i = \frac{e^{\frac{U_i}{\tau}}}{\sum_{j=1}^{N} e^{\frac{U_j}{\tau}}}$
use probability distribution W_1, ..., W_N to select action i
{utility update}
perform action i and observe reward r
$U_i = \lambda \times U_i + (1 - \lambda) \times r$
end loop

where τ_{max} is the maximum temperature of the system, s is a temperature decay coefficient, λ is the learning rate, and N is the total number of actions. The values of U_i are initialized randomly between 0 and 0.001.

Second, the experiments employ the multiagent reinforcement learning algorithm described in (Lauer and Riedmiller, 2000). This algorithm, termed OMQ throughout this chapter, updates the utilities of actions based in part on the maximum reward previously received when performing those actions. OMQ has good performance in deterministic domains, but it might have problems in domains characterized by stochastic rewards, as shown in (Kapetanakis and Kudenko, 2002b). The pseudocode for OMQ is:

Algorithm OMQ
loop {at each time step x}
  {action selection}
  $\tau = e^{-sx} \times \tau_{max} + 1$
  $W_i = \frac{e^{\frac{U_i}{\tau}}}{\sum_{j=1}^{N} e^{\frac{U_j}{\tau}}}$
  use probability distribution W_1, ..., W_N to select action i
  {utility update}
  perform action i and observe reward r
  if $MaxReward_i < r$ **then**
    $MaxReward_i = r$
  end if
  $U_i = \lambda \times U_i + (1-\lambda) \times MaxReward_i$
end loop

where parameters are similar to the ones of MQ. $MaxReward_i$ stores the maximum reward previously observed for action i, and it is initialized to $-\infty$.

Third, LMQ is tested against the FMQ algorithm (Kapetanakis and Kudenko, 2002a). FMQ uses the maximum reward received per action to bias the probability of choosing that action. Previous experiments showed that FMQ outperforms MQ and OMQ in domains with limited amounts of noise, but its performance deteriorates if the level of noise is high (Kapetanakis and Kudenko, 2002b). The pseudocode for FMQ follows.

Algorithm FMQ
loop {at each time step x}
  {action selection}
  $\tau = e^{-sx} \times \tau_{max} + 1$
  $W_i = \frac{e^{\frac{U_i + c \times freq(MaxReward_i) \times MaxReward_i}{\tau}}}{\sum_{j=1}^{N} e^{\frac{U_j + c \times freq(MaxReward_j) \times MaxReward_j}{\tau}}}$
  use probability distribution W_1, ..., W_N to select action i
  {utility update}
  perform action i and observe reward r
  if $MaxReward_i < r$ **then**
    $MaxReward_i = r$
  end if
  $U_i = \lambda \times U_i + (1-\lambda) \times r$
end loop

where the parameters are similar to those of OMQ. Additionally, $freq$ returns the frequency of observing the maximum reward for an action (of all the times that action was selected), and c is a weight assigned to the influence of the FMQ heuristic. The initial value of $freq$ is 0 (assume $0 \times (-\infty) = 0$ for simplicity).

6.2.2 Experimental Setup

I test these algorithms in four repeated coordination games. The first two, Climb and Penalty, are described in Section 3.2.3 (here, I use $\sigma = 30$ for the Climb domain and $\sigma = 10$ for the Penalty domain to reflect the standard benchmarks used elsewhere). The other two, Partially-Stochastic Climb and Fully-Stochastic Climb, are stochastic variations of the Climb domain that have been previously introduced in (Kapetanakis and Kudenko, 2002b). In the Partially-Stochastic Climb domain, only the reward associated with the suboptimal joint action $(2,2)$ is affected by noise: both agents observe a reward of either 14 or 0 (with probability 50% each). All rewards in the Fully-Stochastic Climb domain are affected by noise: both agents observe either of the two possible rewards with probability 50%. Given that agents do not observe each other's actions, the noisy reward information makes these two stochastic variations significantly more difficult than the original Climb domain. The joint payoff matrix for the Partially-Stochastic and Fully-Stochastic Climb domains are defined as:

Partially-Stochastic Climb

$$\begin{bmatrix} 11 & -30 & 0 \\ -30 & 14/0 & 6 \\ 0 & 0 & 5 \end{bmatrix}$$

Fully-Stochastic Climb

$$\begin{bmatrix} 12/10 & 5/-65 & 8/-8 \\ 5/-65 & 14/0 & 12/0 \\ 5/-5 & 5/-5 & 10/0 \end{bmatrix}$$

Learning lasted 7500 time steps, each involving the agents choosing an action, observing a reward, and updating the utility estimates. A brief sensitivity study indicated the following settings to work better for LMQ: $\tau_{max} = 500$, $\alpha = 2$, $\beta = 2.5$, $\nu = 0.995$, and $\lambda = 0.95$. Similarly, a sensitivity study for FMQ involving all combinations of $c \in \{2.5, 5, 7.5, 10, 12.5, 15, 17.5\}$ and $s \in \{0,0008, 0.008, 0.08, 0.8, 8\}$, indicated $c = 15$ and $s = 0.008$ to work best. MQ and OMQ also used $s = 0.008$.

6.2.3 Results

Tables 6.1–6.4 report the number of runs that converged to each of the nine joint actions (based on each agent's action with maximal utility at the end of the run). These numbers are averaged over 10 trials of 1000 runs each.

LMQ consistently converged to the global optimum in the Climb, Penalty, and Partially-Stochastic Climb domains. This is equivalent to the performance of FMQ in these problem domains, and it is also significantly better than the performance of both MQ and OMQ[1]. However, the lenience

[1]The results of OMQ in the Penalty domain are worse than those reported in (Lauer and Riedmiller, 2000). This is caused by OMQ missing a special coordination feature for domains with multiple optima. I opted not to code this feature because it detracted from the primary emphasis of this chapter on stochastic problem domains, where this feature did not provide any help.

MQ

	a	b	c
a	414.6	0	0
b	0	236.3	0
c	0	308	41.1

OMQ

	a	b	c
a	1000	0	0
b	0	0	0
c	0	0	0

FMQ

	a	b	c
a	997.6	0	0
b	0	2.4	0
c	0	0	0

LMQ

	a	b	c
a	991.9	0	0
b	0	8.1	0
c	0	0	0

Table 6.1: Average number of runs (out of 1000) that converged to each of the joint actions in the Climb domain.

MQ

	a	b	c
a	499.9	0	0
b	0	0	0
c	0	0	500.1

OMQ

	a	b	c
a	414.8	0	87.1
b	0	0	0
c	84.3	0	413.8

FMQ

	a	b	c
a	500.9	0	0
b	0	0	0
c	0	0	499.1

LMQ

	a	b	c
a	493.6	0	0
b	0	0	0
c	0	0	506.4

Table 6.2: Average number of runs (out of 1000) that converged to each of the joint actions in the Penalty domain.

also helped the agents learn the global joint action in the Fully-Stochastic Climb domain in almost 95% of the runs. This contrasts FMQ's poor performance in this domain: FMQ converged to the global optimum solution in only around 71.5% of runs.

MQ

	a	b	c
a	388	0	0
b	0	128	0
c	0	443.4	40.6

OMQ

	a	b	c
a	0	0	0
b	0	1000	0
c	0	0	0

FMQ

	a	b	c
a	993.4	0	0
b	0	6.1	0
c	0	0.4	0

LMQ

	a	b	c
a	998.2	0	0
b	0	0	0
c	0	1.4	0.4

Table 6.3: Average number of runs (out of 1000) that converged to each of the joint actions in the Partially-Stochastic Climb domain.

MQ

	a	b	c
a	382.1	0	0
b	0	141.1	328
c	0	0	148.8

OMQ

	a	b	c
a	0	0	0
b	0	1000	0
c	0	0	0

FMQ

	a	b	c
a	714.9	0	0.4
b	0	176.3	54.7
c	0	0	53.7

LMQ

	a	b	c
a	949.3	0	0
b	0	13.6	17.5
c	0	0	19.6

Table 6.4: Average number of runs (out of 1000) that converged to each of the joint actions in the Fully-Stochastic Climb domain.

For additional clarity, Figure 6.1 illustrates the utilities associated with each action for two LMQ agents learning in the Fully-Stochastic Climb domain. Both agents start with low utilities for all actions. After around 2500 time steps, each agent assigns a utility value of around 14 to its action *b*, higher than the utility of every other action. As both agents primarily choose action *b*, they decrease the temperature associated with this action, and they start to incorporate lower rewards into *b*'s utility estimate(remember that the joint action (b,b) has an average utility of 7). As a consequence, the utility of action *b* decreases, and the agents start to explore other actions. This generates extra miscoordination penalties which lower the estimates for the utilities of other actions as well. Given the higher temperature for action *a*, its utility is soon affected by high rewards, and as a result, both agents start to prefer this action over others. This leads to a decrease

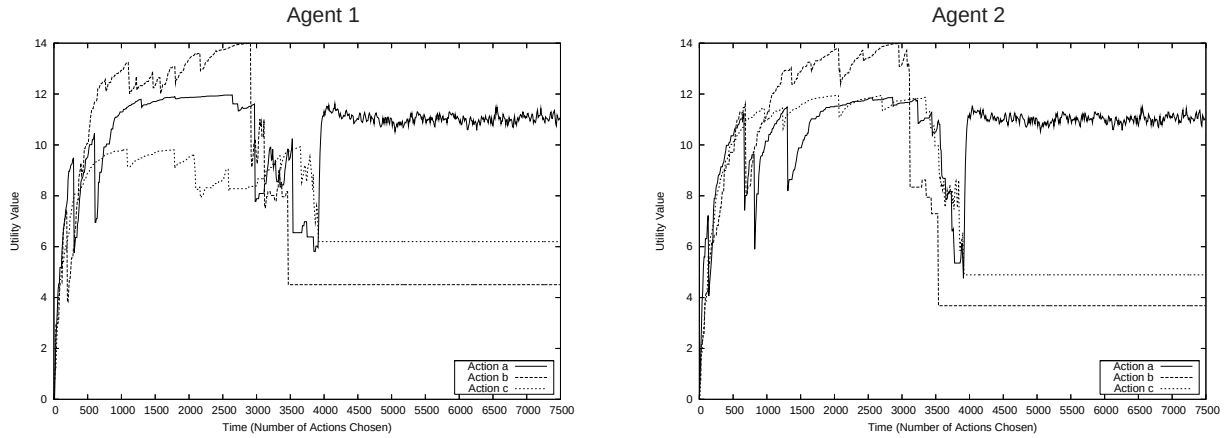

Figure 6.1: Utility of each agent's three actions.

in miscoordination penalties, and both agents end up with precise utilities for their action *a* (around 11), higher than the utilities associated with any other action.

Chapter 7: Conclusions

Multiagent systems have proven to be a powerful approach to complex real-world problems. However, designing multiagent solutions is often a difficult task: one can usually describe what goals the team of agents should accomplish, but cannot easily specify how each agent should behave for those goals to be achieved. Multiagent learning promises to automate this design process, and it might therefore hold the key to significant advancements in emerging research areas that have already multiagent systems.

Recent years have witnessed an increased interest in the research community to identify and characterize what makes concurrent learning different from the more traditional applications of machine learning in single-agent settings. This question has been the primary focus of this thesis. My research identified a key difficulty in concurrent learning: each learner might incorrectly have a poor estimation for the quality of its behaviors due to the impact of other learners' behaviors onto the rewards it receives. An unfortunate consequence of this problem is that the search for better team performance can drift to suboptimal solutions.

One solution to make concurrent learners perform better is to ensure that each of them has an accurate estimate for the quality of its behaviors. In this thesis, I employed formal models to prove the theoretical advantages of learners with accurate estimates. I also applied this theoretical intuition to design three novel concurrent learning algorithms and demonstrated their superior performance over other state-of-the-art techniques.

This thesis concludes with a summary of its main contributions to the area of multiagent learning and some directions for future work.

7.1 Contributions

The primary research contributions of this thesis to the area of cooperative multiagent learning are summarized below.

- **It establishes theoretical guarantees of convergence to the global optimum for certain multiagent learning algorithms.** I claim that multiagent learning might drift to suboptimal solutions simply because each learner has a poor estimate for the quality of its possible behaviors, and I extend certain theoretical models to account for increasingly accurate estimates. Using these models, I prove that multiagent learning algorithms are guaranteed to converge to the global optimum, if properly set and if given enough resources. This result

extends prior theoretical research, showing that claims on the existence of uncontrollable pathologies that plague multiagent learning can be simply attributed to improper settings. Additionally, the theoretical analysis indicates that the key to improving multiagent learning algorithms is to make sure each learning process has an accurate estimate for the quality of its behaviors.

- **It demonstrates the advantages of biasing cooperative coevolutionary algorithms.** I propose and analyze several approaches to combine past and current information in order to improve a learner's estimation for the quality of its behaviors. I show that some biasing schemes might be very sensitive to different settings, while others are not. I then show that several multiagent learning algorithms can be enhanced with a biasing mechanism to achieve significantly better performance.

- **It details a novel multiagent learning algorithm that benefits form simpler, functionally-equivalent estimates that can be obtained with reduced computational costs.** In addition, learners take an active approach to improve each other's estimates for the quality of their behaviors, and as a consequence, they are able to shift more computational resources to searching for better behaviors. I provide empirical results that demonstrate the advantages of this approach in domains where each agent has an infinite set of actions to choose from.

- **It introduces a novel multiagent learning algorithm for stochastic domains.** I address the more difficult task of achieving good quality estimates in domains where the reward information is noisy. In particular, I show that learners can benefit from showing lenience to one another, especially at early stages of learning. I propose a lenient multiagent learning algorithm that significantly outperforms other state-of-the-art algorithms, particularly as the level of noise increases.

The thesis also includes two other important contributions:

- **It provides a broad survey of the cooperative multiagent learning literature.** Previous surveys of this area have largely focused on issues common to specific subareas (for example, reinforcement learning or robotics). These communities have usually worked on similar issues, with little or no cross-fertilization of ideas. The thesis provides an extensive survey of multiagent learning work in a spectrum of areas, including reinforcement learning, evolutionary computation, game theory, complex systems, agent modeling, and robotics. I identify two broad categories of multiagent learning approaches, each with its own special issues: applying a single learner to discover joint solutions to multiagent problems, or using multiple concurrent learners. I also expose critical open issues that the research community needs to address before these methods achieve widespread applicability to real-world problems.

- **It provides graphical illustrations of why concurrent learners might perform suboptimally.** I present intuitive graphical illustrations to support the claim that certain algorithmic decisions might lead to learners having poor quality estimates for their behaviors. This argument is supported further by the visualization of the basins of attraction to optimal and suboptimal solutions for learners with increasingly accurate estimates.

7.2 Future Work

This thesis explores a variety of issues in the area of cooperative multiagent learning, and as such, it opens a diverse set of directions for future research. Here are a few of them.

- The research in this thesis puts forward a formal model for analyzing the impact of improving the estimation for the quality of the agents' behaviors on the overall performance of cooperative coevolutionary algorithms. I also take a first step at providing such models for other concurrent learning paradigms by extending a replicator dynamics model for multiagent Q-learning. My future work will concern with proving convergence guarantees and limitations for the multiagent Q-learning model, as well as with developing novel theoretical models for other concurrent learning paradigms.

- The thesis shows that concurrent learners can have good performance when working with only rough estimates for the quality of their actions. While these estimations only preserve the relative rank ordering of possible behaviors for the agent, they might require significantly less computation. Can mathematical models help us make informed decisions about this tradeoff?

- The formal models I employ make certain assumptions that are infeasible for practical algorithms. While the thesis analyzes both infinite and finite numbers of collaborators, I plan future theoretical work on models involving finite populations as well.

- I perform most of the research in this thesis in very simple domains. However, real-world problems involve many significant challenges: large, possibly infinite sets of states; partial observability of the state of the team, inability to observe other agent's states or actions, to name but a few. Scaling concurrent learning to address such issues is a must, if these algorithms are to be ever used in practice.

- Last, and certainly not least, most agent-based models involve more than two agents (usually tens, hundreds, or thousands of agents). Applying concurrent learning techniques in such situations is certainly a significant challenge for future work. To this end, I am currently experimenting with the decomposition of the learning task along the lines of the roles the

agents need to fill in the team, as opposed to the usual approach involving each agent learning its own behavior. The proposed partitioning drastically simplifies the learning of behaviors for large teams: instead of specifying a behavior for each agent, learning needs to only search for a behavior for each role. I believe that this shift of focus from agents to roles permits learning to scale easily to large teams: the number of roles depends only on the complexity of the problem domain, and it is independent of the size of the team.

Bibliography

[Ackley and Littman, 1994] David H. Ackley and Michael Littman. Altruism in the evolution of communication. In *Artificial Life IV: Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems, third edition*. MIT Press, 1994.

[Andre and Teller, 1999] David Andre and Astro Teller. Evolving team Darwin United. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, 1999.

[Andre *et al.*, 1996] David Andre, Forest H. Bennett III, and John Koza. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In *Genetic Programming 1996: Proceedings of the First Annual Conference*. MIT Press, 1996.

[Bäck, 1996] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolutionary Strategies, Evolutionary Programming, and Genetic Algorithms*. Oxford Press, 1996.

[Balan and Luke, 2006] Gabriel Balan and Sean Luke. History-based traffic control. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006 (to appear)*, pages 616–621. ACM, 2006.

[Balch, 1997] Tucker Balch. Learning roles: Behavioral diversity in robot teams. Technical Report GIT-CC-97-12, Georgia Institute of Technology, 1997.

[Balch, 1998] Tucker Balch. *Behavioral Diversity in Learning Robot Teams*. PhD thesis, College of Computing, Georgia Institute of Technology, 1998.

[Balch, 1999] Tucker Balch. Reward and diversity in multirobot foraging. In *IJCAI-99 Workshop on Agents Learning About, From and With other Agents*, 1999.

[Banerjee *et al.*, 2000] Bikramjit Banerjee, Rajatish Mukherjee, and Sandip Sen. Learning mutual trust. In *Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies*, pages 9–14, 2000.

[Barto *et al.*, 1990] Andrew Barto, Richard Sutton, and Christopher Watkins. Learning and sequential decision making. In Michael Gabriel and John Moore, editors, *Learning and computational neuroscience : foundations of adaptive networks*. M.I.T. Press, Cambridge, Mass, 1990.

[Bassett, 2002] Jeffrey K. Bassett. A study of generalization techniques in evolutionary rule learning. Master's thesis, George Mason University, Fairfax VA, USA, 2002.

[Berenji and Vengerov, 2000a] Hamid R. Berenji and David Vengerov. Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. In *Proceedings of 9th IEEE International Conference on Fuzzy Systems*, 2000.

[Berenji and Vengerov, 2000b] Hamid R. Berenji and David Vengerov. Learning, cooperation, and coordination in multi-agent systems. Technical Report IIS-00-10, Intelligent Inference Systems Corp., 333 W. Maude Avennue, Suite 107, Sunnyvale, CA 94085-4367, 2000.

[Bernstein *et al.*, 2000] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of MDPs. In *Proceedings of UAI-2000: The Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[Beyer, 2004] H.G. *et al* Beyer, editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2005*, New York, New York, 2004. ACM Press.

[Blumenthal and Parker, 2004] H. Joseph Blumenthal and Gary Parker. Co-evolving team capture strategies for dissimilar robots. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[Bonabeau *et al.*, 1999] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. SFI Studies in the Sciences of Complexity. Oxford University Press, 1999.

[Bongard, 2000] Josh C. Bongard. The legion system: A novel approach to evolving heterogeneity for collective problem solving. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin, and Terence C. Fogarty, editors, *Genetic Programming: Proceedings of EuroGP-2000*, volume 1802, pages 16–28, Edinburgh, 15-16 2000. Springer-Verlag.

[Boutilier, 1996] Craig Boutilier. Learning conventions in multiagent stochastic domains using likelihood estimates. In *Uncertainty in Artificial Intelligence*, pages 106–114, 1996.

[Bowling and Veloso, 2001] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1021–1026, 2001.

[Bowling and Veloso, 2002] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[Bowling, 2000] Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 89–94. Morgan Kaufmann, San Francisco, CA, 2000.

[Boyan and Littman, 1994] Justin A. Boyan and Michael Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 671–678. Morgan Kaufmann Publishers, Inc., 1994.

[Brafman and Tennenholtz, 2002] Ronen Brafman and Moshe Tennenholtz. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems (NIPS-2002)*, 2002.

[Bucci and Pollack, 2004] Anthony Bucci and Jordan Pollack. On identifying global optima in cooperative coevolution. In Beyer (2004), pages 539–544.

[Bui *et al.*, 1998] Hung H Bui, Svetha Venkatesh, and Dorota Kieronska. A framework for coordination and learning among team of agents. In W. Wobcke, M. Pagnucco, and C. Zhang, editors, *Agents and Multi-Agent Systems: Formalisms, Methodologies and Applications, Lecture Notes in Artificial Intelligence, Volume 1441*, pages 164–178. Springer-Verlag, 1998.

[Bui *et al.*, 1999] Hung H Bui, Svetha Venkatesh, and Dorota Kieronska. Learning other agents' preferences in multi-agent negotiation using the Bayesian classifier. *International Journal of Cooperative Information Systems*, 8(4):275–294, 1999.

[Bull and Fogarty, 1994] Lawrence Bull and Terence C. Fogarty. Evolving cooperative communicating classifier systems. In A. V. Sebald and Lawrence J. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming (EP94)*, pages 308–315, 1994.

[Bull, 1997] Lawrence Bull. Evolutionary computing in multi-agent environments: Partners. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA)*, pages 370–377. Morgan Kaufmann, 1997.

[Bull, 1998] Lawrence Bull. Evolutionary computing in multi-agent environments: Operators. In D Wagen V W Porto, N Saravanan and A E Eiben, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 43–52. Springer Verlag, 1998.

[Cangelosi, 2001] Angelo Cangelosi. Evolution of communication and language using signals, symbols, and words. *IEEE Transactions on Evolutionary Computation*, 5(2):93–101, 2001.

[Chalkiadakis and Boutilier, 2003] Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *Proceedings of The Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*. ACM, 2003.

[Chalupsky *et al.*, 2002] Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David Pynadath, Thomas Russ, and Milind Tambe. Electric elves: agent technology for supporting human organizations. In *AI Magazine - Summer 2002*. AAAI Press, 2002.

[Chang *et al.*, 2003] Yu-Han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Proceedings of Neural Information Processing Systems (NIPS-03)*, 2003.

[Claus and Boutilier, 1998] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, 1998.

[Collins and Jefferson, 1991] Robert Collins and David Jefferson. An artificial neural network representation for artificial organisms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature: 1st Workshop (PPSN I)*, pages 259–263, Berlin, 1991. Springer-Verlag.

[Collins and Jefferson, 1992] Robert Collins and David Jefferson. AntFarm : Towards simulated evolution. In Christopher Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Artificial Life II*, pages 579–601. Addison-Wesley, Redwood City, CA, 1992.

[Crawford and Veloso, 2004] Elisabeth Crawford and Manuela Veloso. Opportunities for learning in multi-agent meeting scheduling. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[Das *et al.*, 1994] Rajarshi Das, Melanie Mitchell, and James P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In *Parallel Problem Solving from Nature III, LNCS 866*, pages 344–353. Springer-Verlag, 1994.

[de Boer, 1997] Bart de Boer. Generating vowel systems in a population of agents. In *Proceedings of the Fourth European Conference Artificial Life*. MIT Press, 1997.

[De Jong, 2006] Kenneth De Jong. *Evolutionary Computation: A unified approach*. MIT Press, 2006.

[Devroye, 1986] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[Dorronsoro *et al.*, 2004] Bernabe Dorronsoro, Enrique Alba, Mario Giacobini, and Marco Tomassini. The influence of grid shape and asynchronicity on cellular evolutionary algorithms. In G. *et al* Greenwood, editor, *Proceedings of CEC 2004*, pages 2152–2158. IEEE Press, 2004.

[Dresner and Stone, 2004] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[Dudek *et al.*, 1993] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for swarm robots. In *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems*, 1993.

[Durfee *et al.*, 1987] Edmund Durfee, Victor Lesser, and Daniel Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, 1987.

[Fogel, 1999] Lawrence Fogel. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. Wiley Series on Intelligent Systems, 1999.

[Garland and Alterman, 2004] Andrew Garland and Richard Alterman. Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems*, 8:267–301, 2004.

[Ghavamzadeh and Mahadevan, 2004] Mohammad Ghavamzadeh and Sridhar Mahadevan. Learning to communicate and act using hierarchical reinforcement learning. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[Giacobini *et al.*, 2003] Mario Giacobini, Enrique Alba, and Marco Tomassini. Selection intensity in asynchronous cellular evolutionary algorithms. In E. *et al* Cantú-Paz, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2003*, pages 955–966, Berlin, Germany, 2003. Springer.

[Goldberg and Deb, 1990] David Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms (FOGA)*, pages 69–93. Morgan Kaufmann, 1990.

[Goldberg, 1989] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[Goldman and Rosenschein, 1996] Claudia Goldman and Jeffrey Rosenschein. Mutually supervised learning in multiagent systems. In Gerhard Weiß and Sandip Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, pages 85–96. Springer-Verlag: Heidelberg, Germany, Berlin, 1996.

[Good, 2000] Benjamin McGee Good. Evolving multi-agent systems: Comparing existing approaches and suggesting new directions. Master's thesis, University of Sussex, 2000.

[Gordin *et al.*, 1997] Maria Gordin, Sandip Sen, and Narendra Puppala. Evolving cooperative groups: Preliminary results. In *Working Papers of the AAAI-97 Workshop on Multiagent Learning*, pages 31–35, 1997.

[Greenwald and Hall, 2003] Amy Greenwald and Keith Hall. Correlated Q-learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[Grefenstette *et al.*, 1990] John Grefenstette, Connie Loggia Ramsey, and Alan Schultz. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5:355–381, 1990.

[Grefenstette, 1991] John Grefenstette. Lamarckian learning in multi-agent environments. In Rick Belew and Lashon Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 303–310, San Mateo, CA, 1991. Morgan Kaufman.

[Guestrin *et al.*, 2002] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *Proceedings of the 2002 AAAI Symposium Series: Collaborative Learning Agents*, 2002.

[Gustafson and Hsu, 2001] Steven M. Gustafson and William H. Hsu. Layered learning in genetic programming for a co-operative robot soccer problem. In Julian F. Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G. B. Tettamanzi, and William B. Langdon, editors,

Genetic Programming: Proceedings of EuroGP-2001, volume 2038, pages 291–301, Lake Como, Italy, 18-20 2001. Springer-Verlag.

[Gustafson, 2000] Steven M. Gustafson. Layered learning in genetic programming for a co-operative robot soccer problem. Master's thesis, Kansas State University, Manhattan, KS, USA, 2000.

[Hara and Nagao, 1999] Akira Hara and Tomoharu Nagao. Emergence of cooperative behavior using ADG; Automatically Defined Groups. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 1038–1046, 1999.

[Haynes and Sen, 1995] Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Gerhard Weiß and Sandip Sen, editors, *Adaptation and Learning in Multiagent Systems*, Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, Germany, 1995.

[Haynes and Sen, 1996a] Thomas Haynes and Sandip Sen. Adaptation using cases in cooperative groups. In Ibrahim Imam, editor, *Working Notes of the AAAI-96 Workshop on Intelligent Adaptive Agents*, Portland, OR, 1996.

[Haynes and Sen, 1996b] Thomas Haynes and Sandip Sen. Cooperation of the fittest. Technical Report UTULSA-MCS-96-09, The University of Tulsa, April12, 1996.

[Haynes and Sen, 1996c] Thomas Haynes and Sandip Sen. Learning cases to resolve conflicts and improve group behavior. In Milind Tambe and Piotr Gmytrasiewicz, editors, *Working Notes of the AAAI-96 Workshop on Agent Modeling*, pages 46–52, Portland, OR, 1996.

[Haynes and Sen, 1997a] Thomas Haynes and Sandip Sen. Crossover operators for evolving a team. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 162–167, Stanford University, CA, USA, 13-16July 1997. Morgan Kaufmann.

[Haynes and Sen, 1997b] Thomas D. Haynes and Sandip Sen. Co-adaptation in a team. *International Journal of Computational Intelligence and Organizations (IJCIO)*, 1997.

[Haynes *et al.*, 1995a] Thomas Haynes, Sandip Sen, Dale Schoenefeld, and Roger Wainwright. Evolving a team. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 23–30, MIT, Cambridge, MA, USA, 10–12November 1995. AAAI.

[Haynes *et al.*, 1995b] Thomas Haynes, Sandip Sen, Dale Schoenefeld, and Roger Wainwright. Evolving multiagent coordination strategies with genetic programming. Technical Report UTULSA-MCS-95-04, The University of Tulsa, May31, 1995.

[Haynes *et al.*, 1995c] Thomas Haynes, Roger Wainwright, Sandip Sen, and Dale Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In L. Eshelman, editor,

Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), pages 271–278, Pittsburgh, PA, USA, 15-19July 1995. Morgan Kaufmann.

[Haynes *et al.*, 1996] Thomas Haynes, Kit Lau, and Sandip Sen. Learning cases to compliment rules for conflict resolution in multiagent systems. In S. Sen, editor, *AAAI Spring Symposium on Adaptation, Coevolution, and Learning in Multiagent Systems*, pages 51–56, 1996.

[Hillis, 1991] Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Artificial Life II, SFI Studies in the Sciences of Complexity*, 10:313–324, 1991.

[Hofbauer and Sigmund, 1998] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[Holland, 1975] John Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, 1975.

[Holland, 1985] John Holland. Properties of the bucket brigade. In *Proceedings of an International Conference on Genetic Algorithms*, 1985.

[Hölldobler and Wilson, 1990] Bert Hölldobler and Edward O. Wilson. *The Ants*. Harvard University Press, 1990.

[Hsu and Gustafson, 2002] William H. Hsu and Steven M. Gustafson. Genetic programming and multi-agent layered learning by reinforcements. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 764–771, New York, 9-13July 2002. Morgan Kaufmann Publishers.

[Hu and Wellman, 1996] Junling Hu and Michael Wellman. Self-fulfilling bias in multiagent learning. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.

[Hu and Wellman, 1998a] Junling Hu and Michael Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[Hu and Wellman, 1998b] Junling Hu and Michael Wellman. Online learning about other agents in a dynamic multiagent system. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents'98)*, pages 239–246, New York, 1998. ACM Press.

[Hu and Wellman, 2003] Junling Hu and Michael Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[Huhns and Singh, 1998] Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: themes, approaches and challenges. In M.N. Huhns and M.P. Singh, editors, *Readings in Agents*, pages 1–23. Morgan Kaufmann, 1998.

[Husbands and Mill, 1991] Phil Husbands and Frank Mill. Simulated coevolution as the mechanism for emergent planning and scheduling. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, 1991.

[Husbands, 1994] Phil Husbands. Distributed coevolutionary genetic algorithms for multi–criteria and multi-constraint optimisation. In *Evolutionary Computing, AISB Workshop for Selected Papers*, pages 150–165. Springer–Verlag, 1994.

[Iba, 1996] Hitoshi Iba. Emergent cooperation for multiple agents using genetic programming. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature IV: Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *LNCS*, pages 32–41, Berlin, Germany, 1996. Springer Verlag.

[Iba, 1998] Hitoshi Iba. Evolutionary learning of communicating agents. *Information Sciences*, 108, 1998.

[Iba, 1999] Hitoshi Iba. Evolving multiple agents by genetic programming. In Lee Spector, William Langdon, Una-May O'Reilly, and Peter Angeline, editors, *Advances in Genetic Programming 3*, pages 447–466. The MIT Press, Cambridge, MA, 1999.

[Ito, 1997] Akira Ito. How do selfish agents learn to cooperate? In *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 185–192. MIT Press, 1997.

[Jansen and Wiegand, 2003] Thomas Jansen and R. Paul Wiegand. Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA. In E. Cantu-Paz *et al*, editor, *Prooceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Springer-Verlag, 2003.

[Jennings *et al.*, 1998] N.R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agents research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[Jim and Giles, 2000] Kam-Chueun Jim and C. Lee Giles. Talking helps: Evolving communicating agents for the predator-prey pursuit problem. *Artificial Life*, 6(3):237–254, 2000.

[Juille and Pollack, 1998] H. Juille and J. Pollack. Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In *Proceedings of the Third Annual Genetic Programming Conference (GP-98)*, 1998.

[Kaelbling *et al.*, 1996] Leslie Kaelbling, Michael Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kapetanakis and Kudenko, 2002a] Spiros Kapetanakis and Daniel Kudenko. Improving on the reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Second Symposium on Adaptive Agents and Multi-agent Systems (AISB02)*, 2002.

[Kapetanakis and Kudenko, 2002b] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI02)*, 2002.

[Koza, 1992] John Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Lauer and Riedmiller, 2000] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, 2000.

[Leerink *et al.*, 1995] Laurens R. Leerink, Simon R. Schultz, and Marwan A. Jabri. A reinforcement learning exploration strategy based on ant foraging mechanisms. In *Proceedings of the Sixth Australian Conference on Neural Networks*, Sydney, Australia, 1995.

[Lehmann, 1975] Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. McGraw-Hill, 1975.

[Lesser *et al.*, 1987] Victor Lesser, Daniel Corkill, and Edmund Durfee. An update on the distributed vehicle monitoring testbed. Technical Report UM-CS-1987-111, University of Massachessets Amherst, 1987.

[Lichbach, 1996] Mark Irving Lichbach. *The cooperator's dilemma*. University of Michigan Press, 1996.

[Littman, 1994] Michael Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[Littman, 2001] Michael Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328. Morgan Kaufmann Publishers Inc., 2001.

[Luke and Spector, 1996] Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 1996. MIT Press.

[Luke *et al.*, 1997] Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *Proceedings of the First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[Luke *et al.*, 2005] Sean Luke, Keith Sullivan, Liviu Panait, and Gabriel Balan. Tunably decentralized algorithms for cooperative target observation. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2005*, pages 911–917, 2005.

[Luke, 1998] Sean Luke. Genetic programming produced competitive soccer softbot teams for RoboCup97. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 214–222, University of Wisconsin, Madison, Wisconsin, USA, July 1998. Morgan Kaufmann.

[Luke, 2005] Sean Luke. ECJ 13: A Java EC research system. Available at http://cs.gmu.edu/∼eclab/projects/ecj/, 2005.

[Makar *et al.*, 2001] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 246–253, Montreal, Canada, 2001. ACM Press.

[Mataric, 1994a] Maja Mataric. Learning to behave socially. In *Third International Conference on Simulation of Adaptive Behavior*, 1994.

[Mataric, 1994b] Maja Mataric. Reward functions for accelerated learning. In *International Conference on Machine Learning*, pages 181–189, 1994.

[Mataric, 1997] Maja Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[Mataric, 1998] Maja Mataric. Using communication to reduce locality in distributed multi-agent learning. *Joint Special Issue on Learning in Autonomous Robots, Machine Learning, 31(1-3), 141-167, and Autonomous Robots, 5(3-4), Jul/Aug 1998, 335-354*, 1998.

[Maynard Smith, 1982] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[Michalewicz, 1996] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 3rd edition, 1996.

[Miconi, 2001] Thomas Miconi. A collective genetic algorithm. In E. Cantu-Paz *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 876–883, 2001.

[Miconi, 2003] Thomas Miconi. When evolving populations is better than coevolving individuals: The blind mice problem. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[Mitchell *et al.*, 1996] Melanie Mitchell, James P. Crutchfield, and Rajarshi Das. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA'96),*, 1996.

[Monekosso and Remagnino, 2001] Ndedi Monekosso and Paolo Remagnino. Phe-Q: A pheromone based Q-learning. In *Australian Joint Conference on Artificial Intelligence*, pages 345–355, 2001.

[Monekosso and Remagnino, 2002] Ndedi Monekosso and Paolo Remagnino. An analysis of the pheromone Q-learning algorithm. In *Proceedings of the VIII Iberoamerican Conference on Artificial Intelligence IBERAMIA-02*, pages 224–232, 2002.

[Monekosso *et al.*, 2002] Ndedi Monekosso, Paolo Remagnino, and Adam Szarowicz. An improved Q-learning algorithm using synthetic pheromones. In E. Nawarecki B. Dunin-Keplicz, editor, *From Theory to Practice in Multi-Agent Systems, Second International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS 2001 Cracow, Poland, September 26-29, 2001. Revised Papers*, Lecture Notes in Artificial Intelligence LNAI-2296. Springer-Verlag, 2002.

[Moody *et al.*, 2004] John Moody, Yufeng Liu, Matthew Saffell, and Kyoungju Youn. Stochastic direct reinforcement: Application to simple games with recurrence. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[Mühlenbein and Schlierkamp-Voosen, 1993] Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, 1(1):25–49, 1993.

[Mukherjee and Sen, 2001] Rajatish Mukherjee and Sandip Sen. Towards a pareto-optimal solution in general-sum games. In *Agents-2001 Workshop on Learning Agents*, 2001.

[Mundhe and Sen, 2000] Manisha Mundhe and Sandip Sen. Evaluating concurrent reinforcement learners. In *Proceedings of the International Conference on Multiagent System*, 2000.

[Nagayuki *et al.*, 2000] Yasuo Nagayuki, Shin Ishii, and Kenji Doya. Multi-agent reinforcement learning: An approach based on the other agent's internal model. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-00)*, 2000.

[Nair *et al.*, 2003] Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[Nowak and Sigmund, 1998] Martin Nowak and Karl Sigmund. Evolution of indirect reciprocity by image scoring / the dynamics of indirect reciprocity. *Nature*, 393:573–577, 1998.

[Nowe *et al.*, 2001] Ann Nowe, Katja Verbeeck, and Tom Lenaerts. Learning agents in a homo egualis society. Technical report, Computational Modeling Lab - VUB, March 2001.

[Nunes and Oliveira, 2004] Luis Nunes and Eugenio Oliveira. Learning from multiple sources. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[Ohko *et al.*, 1997] Takuya Ohko, Kazuo Hiraki, and Yuichiro Arzai. Addressee learning and message interception for communication load reduction in multiple robots environments. In

Gerhard Weiß, editor, *Distributed Artificial Intelligence Meets Machine Learning : Learning in Multi-Agent Environments, Lecture Notes in Artificial Intelligence 1221*. Springer-Verlag, 1997.

[Pagie, 1999] Ludo Pagie. *Information Integration in Evolutionary Processes*. PhD thesis, Universiteit Utrecht, Netherlands, 1999.

[Panait and Luke, 2004a] Liviu Panait and Sean Luke. Ant foraging revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 2004.

[Panait and Luke, 2004b] Liviu Panait and Sean Luke. Learning ant foraging behaviors. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 2004.

[Panait and Luke, 2004c] Liviu Panait and Sean Luke. A pheromone-based utility model for collaborative foraging. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[Panait and Luke, 2005a] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[Panait and Luke, 2005b] Liviu Panait and Sean Luke. Time-dependent collaboration schemes for cooperative coevolutionary algorithms. In *Proceedings of the 2005 AAAI Fall Symposium on Coevolutionary and Coadaptive Systems*, 2005.

[Panait *et al.*, 2003] Liviu Panait, R. Paul Wiegand, and Sean Luke. Improving coevolutionary search for optimal multiagent behaviors. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[Panait *et al.*, 2004] Liviu Panait, R. Paul Wiegand, and Sean Luke. A visual demonstration of convergence properties of cooperative coevolution. In Schwefel (2004), pages 892–901.

[Panait *et al.*, 2006] Liviu Panait, Sean Luke, and R. Paul Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. IEEE Transactions on Evolutionary Computation (to appear), 2006.

[Papadimitriou and Tsitsiklis, 1987] Christos Papadimitriou and John Tsitsiklis. Complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[Parker, 2000] Lynne Parker. Multi-robot learning in a cooperative observation task. In *Proceedings of Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000)*, 2000.

[Parunak, 1996] H. Van Dyke Parunak. Applications of distributed artificial intelligence in industry. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed AI*. John Wiley & Sons, 1996.

[Peeters *et al.*, 2004] Maarten Peeters, Katja Verbeeck, and Ann Nowe. Multi-agent learning in conflicting multi-level games with incomplete information. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[Peshkin *et al.*, 2000] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Kaelbling. Learning to cooperate via policy search. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–314. Morgan Kaufmann, 2000.

[Popovici and Jong, 2004] Elena Popovici and Kenneth De Jong. Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In Beyer (2004), pages 507–514.

[Potter and De Jong, 1994] Mitchell Potter and Kenneth De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P. Schwefel, editors, *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pages 249–257. Springer, 1994.

[Potter *et al.*, 2001] Mitchell Potter, Lisa Meeden, and Alan Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of The Seventeenth International Conference on Artificial Intelligence (IJCAI-2001)*, 2001.

[Potter, 1997] Mitchell Potter. *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997.

[Puppala *et al.*, 1998] Narendra Puppala, Sandip Sen, and Maria Gordin. Shared memory based cooperative coevolution. In *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pages 570–574, Anchorage, Alaska, USA, 1998. IEEE Press.

[Quinn *et al.*, 2002] Matt Quinn, Lincoln Smith, Giles Mayley, and Phil Husbands. Evolving formation movement for a homogeneous multi-robot system: Teamwork and role-allocation with real robots. Cognitive Science Research Paper 515. School of Cognitive and Computing Sciences, University of Sussex, Brighton, BN1 9QG. ISSN 1350-3162, 2002.

[Quinn, 2001a] Matt Quinn. A comparison of approaches to the evolution of homogeneous multi-robot teams. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, pages 128–135. IEEE Press, 2001.

[Quinn, 2001b] Matt Quinn. Evolving communication without dedicated communication channels. In *Advances in Artificial Life: Sixth European Conference on Artificial Life (ECAL01)*, 2001.

[Ronge and Nordahl, 1996] Andreas Ronge and Mats G. Nordahl. Genetic programs and co-evolution developing robust general purpose controllers using local mating in two dimensional populations. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *LNCS*, pages 81–90, Berlin, Germany, 1996. Springer Verlag.

[Rosin and Belew, 1997] Christopher Rosin and Richard Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[Salustowicz *et al.*, 1997] Rafal Salustowicz, Marco Wiering, and Jürgen Schmidhuber. Learning team strategies with multiple policy-sharing agents: A soccer case study. Technical report, ISDIA, Corso Elvezia 36, 6900 Lugano, Switzerland, 1997.

[Salustowicz *et al.*, 1998] Rafal Salustowicz, Marco Wiering, and Jürgen Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 33(2/3):263–282, 1998.

[Sarma, 1998] Jayshree Sarma. *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 1998.

[Saunders and Pollack, 1996] Gregory M. Saunders and Jordan Pollack. The evolution of communication schemes over continuous channels. In *From Animals to Animats 4 - Proceedings of the Fourth International Conference on Adaptive Behaviour,*, 1996.

[Sauter *et al.*, 2001] John Sauter, H. Van Dyke Parunak, Sven Brueckner, and Robert Matthews. Tuning synthetic pheromones with evolutionary computing. In Robert E. Smith, Claudio Bonacina, Cefn Hoile, and Paul Marrow, editors, *Evolutionary Computation and Multi-Agent Systems (ECOMAS)*, pages 321–324, San Francisco, California, USA, 7 2001.

[Sauter *et al.*, 2002] John Sauter, Robert Matthews, H. Van Dyke Parunak, and Sven Brueckner. Evolving adaptive pheromone path planning mechanisms. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 434–440, 2002.

[Schmidhuber and Zhao, 1996] Jürgen Schmidhuber and Jieyu Zhao. Multi-agent learning with the success-story algorithm. In *ECAI Workshop LDAIS / ICMAS Workshop LIOME*, pages 82–93, 1996.

[Schmidhuber, 1996] Jürgen Schmidhuber. Realistic multi-agent reinforcement learning. In *Learning in Distributed Artificial Intelligence Systems, Working Notes of the 1996 ECAI Workshop*, 1996.

[Schneider *et al.*, 1999] Jeff Schneider, Weng-Keen Wong, Andrew Moore, and Martin Riedmiller. Distributed value functions. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 371–378, 1999.

[Schwefel, 2004] Hans Paul Schwefel *et al* Schwefel, editor. *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Berlin, Germany, 2004. Springer.

[Sekaran and Sen, 1995] Mahendra Sekaran and Sandip Sen. To help or not to help. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages 736–741, Pittsburgh, PA, 1995.

[Sen and Sekaran, 1995] Sandip Sen and Mahendra Sekaran. Using reciprocity to adapt to others. In G. Weiß and S. Sen, editors, *International Joint Conference on Artificial Intelligence Workshop on Adaptation and Learning in Multiagent Sytems, Lecture Notes in Artificial Intelligence*, pages 206–217. Springer-Verlag, 1995.

[Sen and Sekaran, 1996] Sandip Sen and Mahendra Sekaran. Multiagent coordination with learning classifier systems. In Gerhard Weiß and Sandip Sen, editors, *Proceedings of the IJCAI Workshop on Adaption and Learning in Multi-Agent Systems*, volume 1042, pages 218–233. Springer Verlag, 1996.

[Sen and Sekaran, 1998] Sandip Sen and Mahendra Sekaran. Individual learning of coordination knowledge. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):333–356, 1998.

[Sen *et al.*, 1994] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, 1994.

[Shoham *et al.*, 2004] Yoav Shoham, Rob Powers, and Trond Grenager. On the agenda(s) of research on multi-agent learning. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[Singh *et al.*, 2000] Satinder P. Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

[Steeb *et al.*, 1988] Randall Steeb, Stephanie J. Cammarata, Frederick Hayes-Roth, Perry W. Thorndyke, and Robert Wesson. Distributed intelligence for air fleet control. In A.H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 90–101. Morgan Kaufmann Publishers, 1988.

[Steels and Kaplan, 1999] Luc Steels and Frederic Kaplan. Collective learning and semiotic dynamics. In *Proceedings of the European Conference on Artificial Life*, pages 679–688, 1999.

[Steels, 1995] Luc Steels. A self-organizing spatial vocabulary. *Artificial Life*, 2(3):319–332, 1995.

[Steels, 1996a] Luc Steels. Emergent adaptive lexicons. In P. Maes, editor, *Proceedings of the Simulation of Adaptive Behavior Conference*. MIT Press, 1996.

[Steels, 1996b] Luc Steels. Self-organising vocabularies. In *Proceedings of Artificial Life V*, 1996.

[Steels, 1996c] Luc Steels. The spontaneous self-organization of an adaptive language. In S. Muggleton, editor, *Machine Intelligence 15*. Oxford University Press, Oxford, UK, 1996.

[Steels, 1997] Luc Steels. Synthesising the origins of language and meaning using co-evolution, self-organisation and level formation. In J. Hurford, C. Knight, and M. Studdert-Kennedy, editors, *Approaches to the Evolution of Language: Social and Cognitive bases*. Edinburgh University Press, 1997.

[Steels, 2000] Luc Steels. The puzzle of language evolution. *Kognitionswissenschaft*, 8(4):143–150, 2000.

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

[Stone, 1997] Peter Stone. Layered learning in multiagent systems. In *Proceedings of National Conference on Artificial IntelligenceAAAI/IAAI*, 1997.

[Stone, 1998] Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Carnegie Mellon University, 1998.

[Subramanian *et al.*, 1997] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 832–839, 1997.

[Suematsu and Hayashi, 2002] Nobuo Suematsu and Akira Hayashi. A multiagent reinforcement learning algorithm using extended optimal response. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 370–377, 2002.

[Suryadi and Gmytrasiewicz, 1999] Dicky Suryadi and Piotr J. Gmytrasiewicz. Learning models of other agents using influence diagrams. In *Preceedings of the 1999 International Conference on User Modeling*, pages 223–232, 1999.

[Sutton and Barto, 1998] Richard. Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Sutton, 1988] Richard Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[’t Hoen and Tuyls, 2004] Pieter J. ’t Hoen and Karl Tuyls. Analyzing multi-agent reinforcement learning using evolutionary dynamics. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, 2004.

[Tambe, 1995] Milind Tambe. Recursive agent and agent-group tracking in a real-time dynamic environment. In Victor Lesser and Les Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems (ICMAS-95)*. AAAI Press, 1995.

[Tan, 1993] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 487–494. Morgan Kaufmann, San Francisco, CA, USA, 1993.

[Tumer *et al.*, 2002] Kagan Tumer, Adrian K. Agogino, and David H. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 378–385, 2002.

[Tuyls *et al.*, 2003] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for Q-learning in multi-agent systems. In *AAMAS-2003 — Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[Tuyls *et al.*, 2006] Karl Tuyls, Pieter J. 't Hoen, and Bram Vanschoenwinkel. An evolutionary dynamical analysis of multi-agent learning in iterated games. *Autonomous Agents and Multi-Agent Systems*, 12(1):115–153, 2006.

[Varga *et al.*, 1994] László Z. Varga, Nick R. Jennings, and David Cockburn. Integrating intelligent systems into a cooperating community for electricity distribution management. *International Journal of Expert Systems with Applications*, 7(4):563–579, 1994.

[Verbeeck *et al.*, 2003] Katja Verbeeck, Ann Nowe, and Karl Tuyls. Coordinated exploration in stochastic common interest games. In *Proceedings of Third Symposium on Adaptive Agents and Multi-agent Systems*, pages 97 – 102, 2003.

[Verbeeck *et al.*, 2005] Katja Verbeeck, Ann Nowe, Maarten Peeters, and Karl Tuyls. Multi-agent reinforcement learning in stochastic single and multi-stage games. In Daniel Kudenko, Dimitar Kazakov, and Eduardo Alonso, editors, *Adaptive Agents and Multi-agent Systems III: Adaption and Multi-Agent Learning*, Lecture Notes in Computer Science, pages 275 – 294. Springer-Verlag, 2005.

[Vidal and Durfee, 1997] Jose Vidal and Edmund Durfee. Agents learning about agents: A framework and analysis. In *Working Notes of AAAI-97 Workshop on Multiagent Learning*, 1997.

[Vidal and Durfee, 1998] Jose Vidal and Edmund Durfee. The moving target function problem in multiagent learning. In *Proceedings of the Third Annual Conference on Multi-Agent Systems*, 1998.

[Vidal and Durfee, 2003] Jose Vidal and Edmund Durfee. Predicting the expected behavior of agents that learn about agents: the CLRI framework. *Autonomous Agents and Multi-Agent Systems*, January 2003.

[Vose, 1999] Michael Vose. *The Simple Genetic Algorithm*. MIT Press, 1999.

[Wagner, 2000] Kyle Wagner. Cooperative strategies and the evolution of communication. *Artificial Life*, 6(2):149–179, Spring 2000.

[Wang and Sandholm, 2002] Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems (NIPS-2002)*, 2002.

[Watkins and Dayan, 1992] Christopher Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

[Watkins, 1989] Christopher J. Watkins. *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge University, Cambridge, United Kingdom, 1989.

[Weihmayer and Velthuijsen, 1994] Robert Weihmayer and Hugo Velthuijsen. Application of distributed AI and cooperative problem solving to telecommunications. In J. Liebowitz and D. Prereau, editors, *AI Approaches to Telecommunications and Network Management*. IOS Press, 1994.

[Weiß, 1997] Gerhard Weiß, editor. *Distributed Artificial Intelligence Meets Machine Learning : Learning in Multi-Agent Environments*. Number 1221 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1997.

[Weiß, 1999] Gerhard Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[Wellman and Hu, 1998] Michael Wellman and Junling Hu. Conjectural equilibrium in multiagent learning. *Machine Learning*, 33(2-3):179–200, 1998.

[Werger and Mataric, 1999] Barry B. Werger and Maja Mataric. Exploiting embodiment in multi-robot teams. Technical Report IRIS-99-378, University of Southern California, Institute for Robotics and Intelligent Systems, 1999.

[White *et al.*, 1998] Tony White, Bernard Pagurek, and Franz Oppacher. ASGA: Improving the ant system by integration with genetic algorithms. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 610–617, University of Wisconsin, Madison, Wisconsin, USA, 22-25 1998. Morgan Kaufmann.

[Whiteson and Stone, 2003] Shimon Whiteson and Peter Stone. Concurrent layered learning. In *AAMAS-2003 — Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[Whitley *et al.*, 1996] Darrell Whitley, Soraya B. Rana, John Dzubera, and Keith E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.

[Whitley, 1989] Darrell Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, San Mateo, CA, 1989. Morgan Kaufman.

[Wiegand and Sarma, 2004] R. Paul Wiegand and Jayshree Sarma. Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In Schwefel (2004), pages 912–922.

[Wiegand *et al.*, 2001] R. Paul Wiegand, William Liles, and Kenneth De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In L. *et al* Spector, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001*, pages 1235–1242. Morgan Kaufmann, 2001.

[Wiegand *et al.*, 2002a] R. Paul Wiegand, William Liles, and Kenneth De Jong. Analyzing cooperative coevolution with evolutionary game theory. In D. Fogel, editor, *Proceedings of CEC 2002*, pages 1600–1605. IEEE Press, 2002.

[Wiegand *et al.*, 2002b] R. Paul Wiegand, William Liles, and Kenneth De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In R. *et al* Poli, editor, *Foundations of Genetic Algorithms (FOGA) VII*, pages 231–248. Morgan Kaufmann, 2002.

[Wiegand, 2004] R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2004.

[Wiering *et al.*, 1999]
Marco Wiering, Rafal Salustowicz, and Jürgen Schmidhuber. Reinforcement learning soccer teams with incomplete world models. *Journal of Autonomous Robots*, 1999.

[Williams and Mitchell, 2004] Nathan Williams and Melanie Mitchell. Investigating the success of spatial coevolution. In Beyer (2004), pages 523–530.

[Williams, 2004] Andrew Williams. Learning to share meaning in a multi-agent system. *Autonomous Agents and Multi-Agent Systems*, 8:165–193, 2004.

[Wilson, 1975] Edward O. Wilson. *Sociobiology: The New Synthesis*. Belknap Press, 1975.

[Wolpert and Tumer, 2001] David H. Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[Wolpert *et al.*, 1998] David H. Wolpert, Kagan Tumer, and Jeremy Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems-11*, pages 952–958, Denver, 1998.

[Yanco and Stein, 1993] Holly Yanco and Lynn Andrea Stein. An adaptive communication protocol for cooperating mobile robots. In *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, pages 478–485, 1993.

[Zhang and Cho, 1998] Byoung-Tak Zhang and Dong-Yeon Cho. Coevolutionary fitness switching: Learning complex collective behaviors using genetic programming. In *Advances in Genetic Programming III*, pages 425–445. MIT Press, 1998.

[Zhao and Schmidhuber, 1996] Jieyu Zhao and Jürgen Schmidhuber. Incremental self-improvement for life-time multi-agent reinforcement learning. In Pattie Maes, Maja Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart W. Wilson, editors, *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4*, pages 516–525, Cape Code, USA, 9-13 1996. MIT Press.

Curriculum Vitae

Liviu Panait was born in Bucharest, Romania on the 24th of February, 1977. He spent his first 22 years in Romania, specializing in exact sciences, in particular in computer science, mathematics, and physics. He participated in national and international competitions in these fields, where he distinguished himself through his accomplishments. In 1998-1999, he was a member of the University of Bucharest's team that was awarded the fourth place at the prestigious ACM International Programming Contest World Finals, following the first place at the South-Eastern Europe Regional Phase. In 1999, he received a Certificate of Excellence from the Romanian National Association of Software Companies.

Dr. Panait came to the United States of American on the 29th of July, 1999. He was granted a Master of Science degree from George Mason University in 2002, and he successfully defended his PhD thesis in August 2006. His graduate research focused on machine learning, multiagent systems, and artificial life. He authored over 35 articles, which appeared in prestigious journals, conferences, symposia, and workshops. His research was nominated five times for Best Paper Awards at the Genetic and Evolutionary Computation Conferences between 2001 and 2006. Dr. Panait was the recipient of the 2006 Outstanding Graduate Student Award and of numerous fellowships from the George Mason University's Volgenau School of Information Technology and Engineering.

Dr. Panait co-chaired the AAMAS 2006 workshop on Adaptation and Learning in Autonomous Agents and Multiagent Systems, and the AAMAS 2007 workshop on Adaptive and Learning Agents; he also co-organized the AAAI 2005 Fall Symposium on Coevolutionary and Coadaptive Systems, and he served as a program committee member or as an invited reviewer for several international conferences and journals. Dr. Panait is a member of the IEEE Task Force on Coevolution, and he is a co-author of the ECJ evolutionary computation library and the MASON multi-agent simulation toolkit.