Journal of Machine Learning Research 9 (2008) 423-457 Submitted 6/07; Revised 12/07; Published 3/08

Theoretical Advantages of Lenient Learners: An Evolutionary Game Theoretic Perspective

Liviu Panait LIVIU@GOOGLE.COM
Google Inc.
Santa Monica, CA 90401, USA

Karl Tuyls K.TUYLS@MICC.UNIMAAS.NL
Maastricht University
MiCC-IKAT, The Netherlands

Sean Luke SEAN@CS.GMU.EDU
Department of Computer Science
George Mason University
Fairfax, VA 22030, USA

Editor: Leslie Pack Kaelbling

Abstract

This paper presents the dynamics of multiple learning agents from an evolutionary game theoretic perspective. We provide replicator dynamics models for cooperative coevolutionary algorithms and for traditional multiagent Q-learning, and we extend these differential equations to account for lenient learners: agents that forgive possible mismatched teammate actions that resulted in low rewards. We use these extended formal models to study the convergence guarantees for these algorithms, and also to visualize the basins of attraction to optimal and suboptimal solutions in two benchmark coordination problems. The paper demonstrates that lenience provides learners with more accurate information about the benefits of performing their actions, resulting in higher likelihood of convergence to the globally optimal solution. In addition, the analysis indicates that the choice of learning algorithm has an insignificant impact on the overall performance of multiagent learning algorithms; rather, the performance of these algorithms depends primarily on the level of lenience that the agents exhibit to one another. Finally, the research herein supports the strength and generality of evolutionary game theory as a backbone for multiagent learning.

Keywords: multiagent learning, reinforcement learning, cooperative coevolution, evolutionary game theory, formal models, visualization, basins of attraction

1. Introduction

Multiagent learning is a relatively new research area that has witnessed a significant research effort in the past decade. Work in this area is motivated by the wide-applicability of multiagent systems to a large set of complex real-world problem domains, as well as by the difficulty of hard-coding solutions for such distributed approaches (Panait and Luke, 2005a). Multiagent learning techniques have been successfully applied to domains such as multi-rover coordination (Tumer and Agogino, 2007; Knudson and Tumer, 2007), congestive games (Agogino and Tumer, 2006), air traffic control (Agogino and Tumer, 2007), and spacecraft power management (Airiau et al., 2006), to name but a few. Moreover, the formal analysis in Jansen and Wiegand (2003, 2004) demonstrated that these

techniques' strengths rely on the decomposition of the problem into simpler subproblems that can be explored separately, resulting at times in significant speedups over traditional learning algorithms. It is no surprise, then, that such techniques have also been successfully applied to problems outside of the multiagent systems realm, such as the discovery of cascade neural networks for classification tasks (Potter and Jong, 2000), the optimization of inventory control (Eriksson and Olsson, 1997), and the coevolution of neural networks for pole balancing (Gomez et al., 2006).

Multiagent learning is, however, a very challenging problem. Agents typically have limited knowledge of even which actions their teammates are currently performing, and must act not only in a constantly changing environment but one which may be actively *co-adapting* to the agents' actions, often in ways they cannot fully perceive (Panait and Luke, 2005a). Even when agents are notionally *cooperating* with one another, their limited perception capabilities can give rise to phenomena that hinder the agents' ability to coordinate effectively. Such dynamics are still not well understood formally.

Recent research has highlighted a tendency for multiagent learning algorithms to converge to suboptimal solutions, which is in marked contrast to the well-established convergence guarantees of their single-agent counterparts. For example, straightforward extensions of Q-learning to multi-agent systems fail to reach the optimal policy in fairly simple domains (Claus and Boutilier, 1998; Kapetanakis and Kudenko, 2002), despite Q-learning's convergence guarantees in single-agent domains (Sutton and Barto, 1998; Watkins and Dayan, 1992). Similarly, cooperative coevolutionary algorithms Potter and Jong (1994); Potter (1997) can not only converge but also be *attracted to* joint suboptima in the search space (Wiegand, 2004), despite traditional evolutionary algorithms being guaranteed to converge to the global optimum when given enough time and sufficient random exploration (Vose, 1999). Different reasons account for this: other agents are learning as well; the reinforcement an agent receives depends on the actions taken by the other agents; and not all information is observable. Such features make it very difficult to engineer learning algorithms capable of finding optimal solutions under these conditions.

This paper presents a theoretical foundation for multiagent learning in coordination games with no internal state; we only consider symmetric games where all agents receive equal reward. This foundation may be directly applied to two popular algorithms: cooperative coevolutionary algorithms (CCEAs) and multiagent Q-learning. These two techniques are representative of the two families (evolutionary algorithms and multiagent reinforcement learning) which form the bulk of the cooperative multiagent learning field (Panait and Luke, 2005a). In both of these algorithms it is common for an agent to base its assessment of a particular action on the expected reward received for that action. For example, a reinforcement learning agent usually updates its estimate for an action's utility every time[1] that action is performed in the environment, despite the dependence of the reward on the actions performed by the other agents. The formal analysis in this paper demonstrates that the reinforcement provided by the expected reward might be the primary cause for many observed problems with multiagent learning.

The paper applies a new theoretical framework to argue for a different approach to these problems, in which each agent tends to *ignore* lower rewards garnered by a particular action. We call this approach *lenience*: each agent shows lenience to its teammates by ignoring low rewards due to actions chosen by teammates that are poor matches to the agent's current action. For example, consider a simple scenario where agents A_1 and A_2 learn to play soccer together. Let's assume that they

1. This mechanism is essentially equivalent (over time) to using the expected reward.

do not have any *a priori* information about each other's skills. At some point, A_1 attempts a pass that should allow A_2 to receive the ball in a very favorable position (for example, alone with the goalie). However, A_2 is not able to receive the ball properly because he was not expecting the pass, or simply because A_2 has not learned the ball reception skill. Given the failure of their collaboration in this particular instance, should A_1 assume that the pass was bad in general and that it should be avoided in the future (due to the low reward the agents just received)? This is the approach usually taken by straightforward extensions of single-agent learning algorithms to multiagent domains. Alternatively, we argue that A_1 might benefit from showing lenience towards his teammate by completely ignoring the low reward observed following A_2's mistake, and only hope that A_2 will perform better once it learns to play better. In other words, A_1 might assume that the pass was good, and that the low reward is primarily due to A_2 not having learned the skill yet. With time, A_1 will be able to distinguish between good and bad passes based on A_2's future performance.

Lenience is particularly important early in the game, when the agents have not yet identified high-performing actions and so the average reward of a given action can be misleadingly low due to partnering with poor-performing joint actions. In previous work, we have demonstrated the efficacy of algorithms based on this notion: specifically, the Lenient Multiagent Q-learning algorithm (Panait et al., 2006) and the iCCEA algorithm (Panait and Luke, 2006). Here, we will establish a more formal justification for the effectiveness of lenience in multiagent learning, accompanied by an intuitive visualization of the impact that lenience has onto the tendency of these algorithms to converge to suboptimal solutions.

The remainder of this paper is structured as follows. Section 2 introduces basic formal models and background material, and describe the pathologies. In Section 3 we argue that lenience will benefit agents in overcoming such problems. Sections 4 and 5 present enhanced theoretical models of the multiagent learning paradigms. Finally, Section 6 concludes the paper and suggests directions for future research.

2. Background

This paper presents a formal analysis of two multiagent learning algorithms: cooperative coevolutionary algorithms and multiagent Q-learning. Sections 2.1.1 and 2.1.2 briefly describe each of them. Following, Sections 2.2.1 and 2.2.2 introduce the basics of the replicator equations both in discrete and continuous time. Sections 2.3.1 and 2.3.2 present two evolutionary game theory models that capture the dynamics of the two learning algorithms, that is, CCEA and multiagent Q-learning. These replicator dynamics (RD) models represent the foundation for our work and will be extended in Section 3 to account for lenience. More details on both models can be found in Wiegand (2004), Tuyls et al. (2006) and Tuyls et al. (2003).

2.1 Multiagent Learning Methods

This section provides a brief overview of two multiagent learning algorithms.

2.1.1 COOPERATIVE COEVOLUTIONARY ALGORITHMS

Evolutionary algorithms are beam search methods that employ and refine sets of candidate solutions to a given problem. The terminology in the field is borrowed from biology to reflect the main source of inspiration (Darwin's models for evolution) for these techniques; 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 algorithm 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. For additional information, interested readers are referred to Jong (2006).

Coevolution is an intuitive 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 Jong, 1994). Here, each agent is given its own population[2] 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. The populations evolve concurrently (this is closer to the asynchronous nature of multiagent systems). Potter (1997), Potter and Jong (2000) and Wiegand (2004) provide comprehensive overviews of cooperative coevolutionary algorithms.

2.1.2 MULTIAGENT Q-LEARNING

Q-learning (Watkins, 1989; Sutton and Barto, 1998) is particularly useful in domains where reinforcement information (expressed as penalties or rewards) is stochastic and is observed after a sequence of actions has been performed. The algorithm associates a utility (or *Q*uality) 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

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

where the agent has just been in state s_t, has performed action a_t, has observed reward r_t, and has transitioned to the new (current) state s_{t+1}; α is the learning rate, and γ is a discount factor (bounded between 0 and 1) for incorporating future rewards into the utility estimate. Note that $Q(s_t,a_t)$ does not only depend on the immediate reward r_{t+1} observed by the agent, but also on future rewards (via the $\max_a Q(s_{t+1},a)$ term). The discount factor γ controls the impact of future versus immediate

2. Given this assumption that each agent has its own population, we sometimes use the term *population* instead of *agent* when the context deems it more appropriate.

rewards onto the *Q*uality terms: lower values of γ reduce the impact of future rewards onto the quality estimate for a state-action pair.

Action selection in Q-learning is usually based on a stochastic process; popular choices include ε-*greedy exploration* (select the best action with probability $1-\varepsilon$, or a random action otherwise) and *Boltzmann exploration* (the selection process further depends on a "temperature" parameter). As previously mentioned, 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 (Singh et al., 2000; Watkins and Dayan, 1992).

Multiagent Q-learning is 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 (s,a) pairs. All agents choose their actions independently and concurrently, perform them in parallel, and observe the same reward associated with the joint action. In contrast to single-agent Q-learning, the information an agent observes depends on the actions chosen by other agents. The reward information is then used by each agent to update its Q-values. Agents usually cannot perceive which actions their teammates have chosen.

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

$$P(a_k) = \frac{e^{\frac{Q(s,a_k)}{\tau}}}{\sum_i e^{\frac{Q(s,a_i)}{\tau}}} \tag{1}$$

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).

2.2 The Replicator Dynamics

John Maynard Smith introduced Evolutionary Game Theory (EGT) by applying traditional game theory to Biology (Smith, 1982; Maynard Smith and Price, 1973). EGT is particularly well-suited as a model of cooperative learning agents. In EGT, each agent holds a vector of proportions over possible actions, indicating the proportion of "individuals" or "replicators" in an infinite "population" (the agent) which have adopted a given action. EGT also commonly presumes one or more matrixes that represent the performance of actions in the context of actions of the other agents. Each timestep, the proportions of actions for each agent are changed based on the rewards in the matrix as well as on the current probabilities of other agents' choosing their actions.

Replicator dynamics (RD) is a key concept in EGT to express the adaptation of each population over time. RD can be represented either in discrete or continuous time. We elaborate on both forms of RD, starting with the discrete time version. In essence the replicator equations describe how a population of different actions evolves through time. An abstraction of an evolutionary process usually combines two basic elements: selection and mutation. Selection favors some population actions over others, while mutation provides variety in the population. The most basic form of RD only highlights the role of selection, that is, how the most fit actions in a population are selected.

2.2.1 DISCRETE TIME REPLICATOR DYNAMICS

Suppose there is a single population of actions (or replicators) and consider a discrete time process $t = 1,2,\ldots$. Let $A = (a_{ij})_{i,j=1}^{n}$ be the reward matrix ($a_{ij} \geq 0$ is the reward for the joint action (i,j),

and n is the total number of possible actions). Let us assume that the individuals in the population (considered infinite) represent different actions that the agent can perform. The state of the population can be described by a vector $x(t) = (x_1(t), ..., x_n(t))$, where $x_i(t)$ denotes the proportion of individual i in the population (it is also directly related to the probability that the agent will select action i).

At each time step t, the state $x(t)$ of the population changes according to the fitness values of the different individuals. More precisely, the expected number of offsprings for a single individual representing action i equals the ratio between the expected payoff for such an individual, $\sum_{j=1}^{n} a_{ij}x_j(t)$, and the average payoff $\sum_{k=1}^{n} x_k(t)\left(\sum_{j=1}^{n} a_{kj}x_j(t)\right)$ for all individuals in the population. Therefore, the ratio $x_i(t+1)$ of individuals representing action i at the next time step equals:

$$x_i(t+1) = x_i(t)\frac{\sum_{j=1}^{n} a_{ij}x_j(t)}{\sum_{k=1}^{n} x_k(t)\left(\sum_{j=1}^{n} a_{kj}x_j(t)\right)}.$$

Dividing both sides by $x_i(t)$ leads to:

$$\frac{x_i(t+1)}{x_i(t)} = \frac{\sum_{j=1}^{n} a_{ij}x_j(t)}{\sum_{k=1}^{n} x_k(t)\left(\sum_{j=1}^{n} a_{kj}x_j(t)\right)}.$$

A minor manipulation of the fractions leads to the general discrete time replicator dynamics:

$$\frac{\Delta x_i(t)}{x_i(t)} = \frac{\sum_{j=1}^{n} a_{ij}x_i(t) - x(t)Ax(t)}{x(t)Ax(t)}.$$

This system of equations expresses Darwinian selection as follows: the rate of change Δx_i of a particular action i is the difference between its average payoff, $\sum_{j=1}^{n} a_{ij}x_i(t)$, and the average payoffs for all actions, $x(t)Ax(t)$.

2.2.2 CONTINUOUS TIME REPLICATOR DYNAMICS

From a mathematical viewpoint, it is usually more convenient to work in continuous time. Therefore we now derive the continuous version of the above replicator equations. To do this we make time infinitesimal. More precisely, suppose that the amount of time that passes between two periods is given by δ with $\delta \in [0,1]$. We also assume that during a time period δ, only a fraction δ of the individuals die and generate offspring. Then the above equation can be rewritten as follows:

$$\frac{\Delta x_i(t)}{x_i(t)} = \frac{x_i(t+\delta) - x_i(t)}{x_i(t)} = \frac{\sum_{j=1}^{n} \delta a_{ij}x_i(t) - x(t)\delta Ax(t)}{x(t)\delta Ax(t) + (1-\delta)}.$$

Now, at the limit when δ approaches 0, the continuous replicator equations are:

$$\frac{dx_i}{dt} = \left[\sum_{j=1}^{n} a_{ij}x_i - x \cdot Ax\right] x_i$$

which can be rewritten as:

$$\frac{dx_i}{dt} = [(Ax)_i - x \cdot Ax]\, x_i. \tag{2}$$

In Equation 2, x_i represents the density of action i in the population, and A is the payoff matrix that describes the different payoff values that each individual replicator receives when interacting with other replicators in the population. The state x of the population can be described as a probability vector $x = (x_1, x_2, ..., x_n)$ which expresses the different densities of all the different types of replicators in the population. Hence $(Ax)_i$ is the payoff that replicator i receives in a population with state x and $x \cdot Ax$ describes the average payoff in the population. The growth rate $\frac{\frac{dx_i}{dt}}{x_i}$ of the proportion of action i in the population equals the difference between the action's current payoff and the average payoff in the population. Gintis (2001), Hofbauer and Sigmund (1998) and Weibull (1996) detail further information on this issues.

2.3 EGT Models for Multiagent Learning Algorithms

This paper is concerned with formal models of multiple agents that learn concurrently. For simplicity, the discussion focuses on only two such learning agents in a symmetric game (both agents receive the same payoff). As a result, we need two systems of differential equations, one for each agent. This setup corresponds to an RD for asymmetric games, where the available actions or strategies of the agents to belong to two different population. For example, consider the case of extending the continuous RD model to a multiagent learning scenario involving a two-player symmetric game with two agents, and consider that A^T is the payoff matrix that describes the reinforcements received by the second agent. Then, another equation similar to Equation 2 would emerge again to characterize the dynamics of the second learner.

This translates into the following replicator equations for the two populations:

$$\frac{dp_i}{dt} = [(Aq)_i - p \cdot Aq]p_i, \tag{3}$$

$$\frac{dq_i}{dt} = [(A^T p)_i - q \cdot A^T p]q_i. \tag{4}$$

Equations 3 and 4 indicate that the growth rate of the types in each population is additionally determined by the composition of the other population, in contrast to the single population (learner) case described by Equation 2.

Next, we present two EGT models for multiagent learning. First, Section 2.3.1 presents a model for cooperative coevolutionary algorithms, which is based on the discrete time RD model in Section 2.2.1. Then, Section 2.3.2 presents a model for multiagent Q-learning, which is based on the continuous time RD model in Section 2.2.2.

2.3.1 EGT MODEL FOR COOPERATIVE COEVOLUTIONARY ALGORITHMS

Wiegand (2004) describes an evolutionary game theoretic model for cooperative coevolutionary algorithms. The model applies to stateless domains involving only two agents, and where each agent has a finite set of actions to choose from. The two populations (one per agent) contain individuals, each of them representing an action that the agent might perform. The model further assumes that the two populations are infinite, and it computes the proportions of individuals in the populations at each generation. If the first agent has a finite number n of distinct actions 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 (as in Section 2.3). Equations 5–8 describe the EGT model for CCEAs, as expressed in Wiegand (2004):

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} y_j^{(t)}, \tag{5}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} x_i^{(t)}, \tag{6}$$

$$x_i^{(t+1)} = \left(\frac{u_i^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}} \right) x_i^{(t)}, \tag{7}$$

$$y_j^{(t+1)} = \left(\frac{w_j^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}} \right) y_j^{(t)} \tag{8}$$

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 action is computed for the two populations (Equations 5 and 6). The fitness of action i is estimated as the mean payoff over pairwise collaborations with every action in the other population. Note that this computation uses the proportions of each action in the (infinite) populations. Second, the distributions of the two populations at the next generation are computed (Equations 7 and 8 model the effects of fitness proportional selection).

2.3.2 EGT MODEL FOR MULTIAGENT Q-LEARNING

This section introduces the RD model of Multiagent Q-learning. For a complete mathematical derivation of it, please refer to Tuyls et al. (2006, 2003). Basically, the derivation boils down to constructing a continuous time limit of the Q-learning model (in the same manner as when going from the discrete replicator equations to the continuous version), where Q-values are interpreted as Boltzmann probabilities for the action selection. For simplicity, we only consider games between two agents, and assume that the game is stateless: the reward that the agents receive depends solely on the actions they have currently performed in the environment. We admit up front that such scenarios are unrealistic for practical purposes, but they are complex enough to emphasize specific challenges for multiagent reinforcement learning algorithms. For this reason, several previous empirical investigations of these techniques have employed such domains (for example, by Claus and Boutilier 1998, Kapetanakis and Kudenko 2002 and Lauer and Riedmiller 2000).

Each agent has a probability vector over its action set, more precisely $x_1, \ldots, x_n$ over action set $a_1, \ldots, a_n$ for the first agent and $y_1, \ldots, y_m$ over $b_1, \ldots, b_m$ for the second agent. Equation 1 can also be rewritten as follows:

$$x_i(k) = \frac{e^{\frac{Qa_i(k)}{\tau}}}{\sum_{j=1}^{n} e^{\frac{Qa_j(k)}{\tau}}}$$

where $x_i(k)$ is the probability of playing action i at time step k and τ is the temperature. Now we can find an expression for, $x_i(k+1)$:

$$\begin{aligned}\frac{x_i(k+1)}{x_i(k)} &= \frac{e^{\frac{Qa_i(k+1)}{\tau}}\sum_j e^{\frac{Qa_j(k)}{\tau}}}{e^{\frac{Qa_i(k)}{\tau}}\sum_j e^{\frac{Qa_j(k+1)}{\tau}}} \\ &= \frac{e^{\frac{\Delta Qa_i(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}\end{aligned}$$

from which follows that

$$x_i(k+1) = x_i(k)\frac{e^{\frac{\Delta Qa_i(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}.$$

Considering the difference equation for x_i, we have that

$$\begin{aligned}x_i(k+1) - x_i(k) &= \frac{x_i(k) e^{\frac{\Delta Qa_i(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}} - x_i(k) \\ &= x_i(k)\left(\frac{e^{\frac{\Delta Qa_i(k)}{\tau}} - \sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}\right).\end{aligned}$$

For the continuous time version, suppose that the amount of time that passes between two repetitions of the game is given by δ with $0 < \delta \leq 1$. The variable $x_i(k\delta)$ describes the x-values at time $k\delta = t$. Under these assumptions, it follows that

$$\begin{aligned}\frac{x_i(k\delta+\delta) - x_i(k\delta)}{\delta} &= \frac{x_i(k\delta)}{\delta\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}} \times \\ &\quad \left(e^{\frac{\Delta Qa_i(k\delta)}{\tau}} - \sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}\right).\end{aligned}$$

We are interested in the limit with $\delta \to 0$. At a given time $t \geq 0$, the state of the limit process is computed by taking the limit of $x_i(k\delta)$ with $\delta \to 0$ and $k\delta \to t$.

$$\begin{aligned}\lim_{\delta\to 0}\frac{\Delta x_i(k\delta)}{\delta} &= \lim_{\delta\to 0}\frac{x_i(k\delta)}{\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}} \times \\ &\quad \lim_{\delta\to 0}\left(\frac{e^{\frac{\Delta Qa_i(k\delta)}{\tau}}}{\delta} - \frac{\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}}{\delta}\right). \qquad (9)\end{aligned}$$

The first limit equals $x_i(t)$ (or x_i for a shorter notation):

$$\lim_{\delta\to 0}\frac{x_i(k\delta)}{\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}} = x_i(t) \qquad (10)$$

because the exponent term becomes 1 ($\Delta Q_{a_j}(k\delta)$ becomes 0) and $\sum_j x_j(k\delta)$ equals 1(sum of all probabilities equals 1). Therefore,

$$\lim_{\delta\to 0}\frac{\Delta x_i(k\delta)}{\delta} = x_i \times \lim_{\delta\to 0}\left(\frac{e^{\frac{\Delta Q_{a_i}(k\delta)}{\tau}}}{\delta} - \frac{\sum_j x_j(k\delta)e^{\frac{\Delta Q_{a_j}(k\delta)}{\tau}}}{\delta}\right).$$

The second limit is undefined (this is a $\frac{0}{0}$ situation), which allows us to use L'Hôpital's rule. The second term now equals

$$\lim_{\delta\to 0}\left(\frac{e^{\frac{\Delta Q_{a_i}(k\delta)}{\tau}}}{\delta} - \frac{\sum_j x_j(k\delta)e^{\frac{\Delta Q_{a_j}(k\delta)}{\tau}}}{\delta}\right) = \frac{dQ_{a_i}(t)}{\tau dt} - \sum_j x_j(t)\frac{dQ_{a_j}(t)}{\tau dt}. \quad (11)$$

The total limit now becomes (by combining Equations 9 – 11),

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{1}{\tau}\left(\frac{dQ_{a_i}}{dt} - \sum_j \frac{dQ_{a_j}}{dt}x_j\right). \quad (12)$$

This completes the derivation of the continuous time model for the Q-learning process. As can be seen in Equation 12, we need an expression for $\frac{dQ_{a_i}(t)}{dt}$. This can be derived the differential equation for the Q-function by performing the following steps.

The Q-learning update rule for the first agent can be written as follows:

$$Q_{a_i}(k+1) = Q_{a_i}(k) + \alpha(r_{a_i}(k+1) + \gamma\max_{a_i} Q - Q_{a_i}(k))$$

which implies,

$$\Delta Q_{a_i}(k) = \alpha(r_{a_i}(k+1) + \gamma\max_{a_i} Q - Q_{a_i}(k)).$$

This expression is the difference equation for the Q-function. To make this equation infinitesimal, going from discrete steps to a continuous version, we suppose that the amount of time that passes between two repetitions of updates of the Q-values is given by δ with $0 < \delta \leq 1$. The variable $Q_{a_i}(k\delta)$ describes the Q-values at time $k\delta$. It follows that

$$\Delta Q_{a_i}(k\delta) = \alpha(r_{a_i}((k+1)\delta) + \gamma\max_{a_i} Q - Q_{a_i}(k\delta))((k+1)\delta - k\delta)$$

which is the same as writing

$$\Delta Q_{a_i}(k\delta) = \alpha(r_{a_i}((k+1)\delta) + \gamma\max_{a_i} Q - Q_{a_i}(k\delta))\delta.$$

We are interested in the limit $\delta \to 0$. The state of the limit process at some time $t \geq 0$ (which we keep fixed) can be computed by taking the limit of $Q_{a_i}(k\delta)$ with $\delta \to 0$ and $k\delta \to t$. Bringing δ to the left side, and taking the limit for $\delta \to 0$, it follows that

$$\frac{dQ_{a_i}}{dt} = \alpha(r_{a_i} + \gamma\max_{a_i} Q - Q_{a_i}). \quad (13)$$

Now, substituting Equation 13 in Equation 12 yields

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{1}{\tau}\alpha\left(r_{a_i} - \sum_j x_j r_{a_j} - Q_{a_i} + \sum_j Q_{a_j} x_j\right)$$

because $\sum_j x_j = 1$, this expression becomes

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(r_{a_i} - \sum_j x_j r_{a_j} + \sum_j x_j (Q_{a_j} - Q_{a_i})\right).$$

Given that $\frac{x_j}{x_i}$ equals $\frac{e^{\frac{Q_{a_j}}{\tau}}}{e^{\frac{Q_{a_i}}{\tau}}}$, it follows that

$$\alpha \sum_j x_j ln \frac{x_j}{x_i} = \frac{\alpha}{\tau} \sum_j x_j (Q_{a_j} - Q_{a_i})$$

which gives us

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(r_{a_i} - \sum_j x_j r_{a_j}\right) + \alpha \sum_j x_j ln\left(\frac{x_j}{x_i}\right).$$

Let us put this equation in the context of the two concurrent learning agents. For clarity, we use u_i (instead of r_{a_i}) to indicate the expected reward that the first agent expects for performing action i, and w_j to indicate the expected reward that the second agent expects for performing action j. Remember that the payoff matrix for the second agent is simply the transposed payoff matrix used by the first agent.

Thus $u_i = \sum_k a_{ik} y_k$ 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 $w_j = \sum_k a_{kj} x_k$). The RD model above therefore has the simpler form,

$$u_i = \sum_k a_{ik} y_k, \tag{14}$$

$$w_j = \sum_k a_{kj} x_k, \tag{15}$$

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(u_i - \sum_k x_k u_k\right) + \alpha \sum_k x_k \ln \frac{x_k}{x_i}, \tag{16}$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau}\left(w_j - \sum_k y_k w_k\right) + \alpha \sum_k y_k \ln \frac{y_k}{y_j}. \tag{17}$$

2.4 The Relative Overgeneralization Pathology

Wiegand et al. (2002) used the EGT model in Equations 5–8 to provide a formal analysis for the properties of cooperative coevolutionary algorithms. Note that the model described is deterministic: 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 (the stochastic nature of evolutionary algorithms is compensated for by the use of infinite populations). Using this model, Wiegand et al.

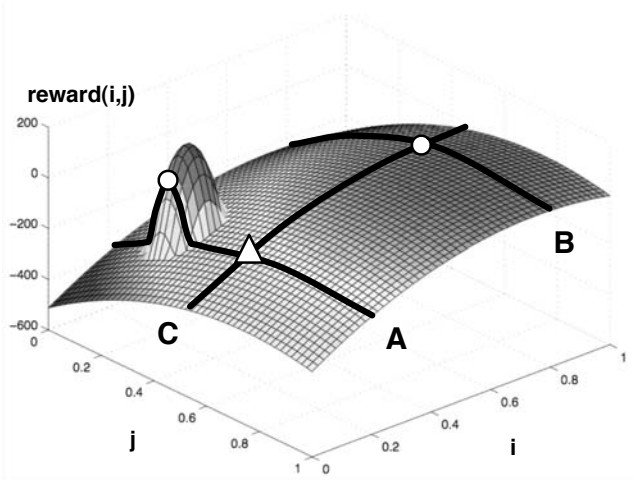

Figure 1: The relative generalization pathology in multiagent cooperative learning. The axes i and j are the various actions afforded to two different agents, and the axis $reward(i, j)$ is the reward received from a given joint action (higher is better).

(2002) showed that pure Nash equilibria are stable, attracting fixed points. This implies that starting from certain initial configurations and iterating the model until convergence may result in convergence to suboptimal Nash equilibria. A suboptimal Nash equilibrium is one that is Pareto-dominated by at least one other Nash equilibrium. This pathology, termed *relative overgeneralization* in Wiegand (2004), indicates that CCEAs have a tendency to move not to points of optimum reward, but rather to points which are "jacks of all trades but masters of none." This argument is taken further in Wiegand and Potter (2006) to suggest that cooperative coevolutionary algorithms might not necessarily search for solutions of high quality, but rather for robust solutions.

To illustrate how this can occur, consider a trivial scenario involving two agents, each with a set of actions. Both agents receive the same reward based on the joint action selected, and the objective of the two agents is to arrive on the joint action that optimizes their joint reward. Let i and j be the spaces of actions for the two agents. Figure 1 shows one possible joint space of actions $\langle i, j \rangle$ and their resulting reward. The optimum is at the top of the small peak; but it has been shown that a dynamical system such as in Equations 5 – 8 will tend to converge to the top of the broad suboptimal peak (Wiegand, 2004). The reason is as follows. Line A represents the possible rewards for the first population following action i_A when paired with various actions j from the other population. Some of these rewards are near the optimum; but most of them are below the suboptimal rewards that action i_B (line B) receives. If the utility of an action is based on *average reward*, the utility for i_B is higher that that of i_A, even though i_A has the potential to form better solutions (the circle on the higher peak in Figure 1). As a result, the system moves towards solutions more in line with i_B. This is the relative overgeneralization pathology.

Similar issues were reported in the multiagent Q-learning literature as well. For example, Fulda and Ventura (2007) attribute the convergence to suboptimal solutions to what they term *action shadowing*. Action i shadows action j if both of the following conditions apply. First, the rewards ob-

served by the agent upon performing action i over a period of time are higher than the ones observed for action j during the same period of time. This results in an expected higher utility for performing action i, as opposed to performing action j. Second, the reward obtained for j when the teammate selects a specific action k is higher than the reward obtain for i with any action selected by the teammate. For example, the domain illustrated in Figure 1 presents a scenario where action i_B shadows action i_A. This occurs because of low rewards received for i_A with poorly-matched actions (such as j_C) selected by the teammate. The circles represent how good actions i_A and i_B could be (when matched by specific actions of the teammate), while the triangle illustrates the low rewards due to action j_C that contribute to action shadowing (action shadowing is the result of many such actions).

Note informally that both relative overgeneralization and action shadowing might be solved by using the same approach: basing an action's utility on the *maximum* reward over some large number N of interactions with actions chosen by the other player. As N grows, the utility of an action approaches the joint reward it would produce were it paired with its optimum collaborating action; and thus each population could select from its actions knowing the true optimum potential of each.

This general notion of using the maximum is at the center of what we call *lenient learning*. The remainder of the paper demonstrates the advantages of being lenient in a multiagent learning system.

3. Lenient Learners

Lenient learning attempts to address the primary pathologies described in Section 2.4 by allowing agents to ignore the lower rewards received for a given action, notionally due to poorly-matched actions from the teammate, and instead concentrate on the higher rewards seen so far for that action.

Consider the Climb and Penalty domains introduced in Claus and Boutilier (1998) and used in previous investigations of multiagent Q-learning (Kapetanakis and Kudenko, 2002; Lauer and Riedmiller, 2000) and cooperative coevolution (Panait et al., 2003). The joint payoff matrices for these coordination games are defined as:

$$Climb: \begin{bmatrix} 11 & -30 & 0 \\ -30 & 7 & 6 \\ 0 & 0 & 5 \end{bmatrix}, \qquad Penalty: \begin{bmatrix} 10 & 0 & -10 \\ 0 & 2 & 0 \\ -10 & 0 & 10 \end{bmatrix}.$$

Observe that Climb has two Nash equilibria, the optimum $(1,1)$ (both agents play their first strategy: the indices of the matrix) and the suboptimum $(2,2)$ (both agents play their second strategy), and that it is strongly deceptive. Suboptimal meaning that $(2,2)$ is Pareto-dominated by $(1,1)$. Penalty has three Nash equilibria: $(1,1)$, $(2,2)$, and $(3,3)$. Of them, $(2,2)$ is suboptimal (Pareto-dominated), but is more forgiving if one or the other population deviates from the Nash equilibrium. This means that $(2,2)$ risk dominates $(1,1)$ and $(3,3)$, because playing action 2 will provide a higher expected payoff when an agent is uncertain about the other agent's action,.

Now consider two agents learning the Climb coordination game. Given that the agents have just started to learn and have no knowledge about the structure of the payoff matrix, we may assume that the agents choose their actions randomly. The expected reward for each action an agent might perform is:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 11*\tfrac{1}{3} - 30*\tfrac{1}{3} + 0*\tfrac{1}{3} &= -6.33\\
ExpectedReward(a_2) = &\quad -30*\tfrac{1}{3} + 7*\tfrac{1}{3} + 6*\tfrac{1}{3} &= -5.67\\
ExpectedReward(a_3) = &\quad 0*\tfrac{1}{3} + 0*\tfrac{1}{3} + 5*\tfrac{1}{3} &= 1.67
\end{aligned}$$

Observe that action a_1 has the lowest expected reward. The agent might thus become less likely to choose action a_1 due to the lower rewards it receives at early stages of learning. This is problematic, however, as a_1 corresponds to the optimal Nash equilibrium. The same applies to the Penalty coordination problem:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 10*\tfrac{1}{3} + 0*\tfrac{1}{3} - 10*\tfrac{1}{3} &= 0\\
ExpectedReward(a_2) = &\quad 0*\tfrac{1}{3} + 2*\tfrac{1}{3} + 0*\tfrac{1}{3} &= 0.67\\
ExpectedReward(a_3) = &\quad -10*\tfrac{1}{3} + 0*\tfrac{1}{3} + 10*\tfrac{1}{3} &= 0
\end{aligned}$$

One solution to remedy this problem is to allow agents to show lenience towards their teammates: the agents can *ignore* lower rewards observed upon performing their actions, and only update the utilities of actions based on the higher rewards. This can be achieved if the learners compare the observed reward with the estimated utility for an action, and update the utility only if it is lower than the reward. In some sense, such an approach changes the learning focus from performing as well as possible in the context of the current (likely mediocre) teammate, to performing as well as possible in the context of improved behaviors for its teammate (as the teammate learns, it likely will not choose those actions that resulted in lower rewards!).

Panait et al. (2006) presents evidence on the advantages of such an approach. The paper introduced a time-dependent degree of lenience: the agents start by ignoring most of the low rewards that they observe, but as learning progresses, agents tend to explore certain "better" actions and also ignore fewer low rewards. The paper also presented empirical evidence that several multiagent learning paradigms can significantly benefit from agents being lenient to one another. Similarly, cooperative coevolutionary algorithms often use agents that ignore lower rewards (for example, in Wiegand et al., 2001), and we previously demonstrated the advantages of a time-dependent degree of lenience for cooperative coevolution (Panait and Luke, 2005b). Although such algorithms have shown their strengths in empirical analysis, formal models of multiagent learning have always been one step behind.

This paper focuses on the mathematical foundations of a simpler approach to lenient learners: each agent collects the rewards it receives for performing actions. Upon observing N rewards for an action, only the maximum of these N rewards is used to update the utility associated with that action. The set of observed rewards for that action is cleared, and the learning process continues. In other words, the agents employ a fixed degree of lenience (which is determined by N) to their teammates. The more low rewards the agents ignore (the higher N is), the more lenient the agents can be said to be. Although this approach does not model any existing multiagent learning algorithm, its formal analysis has two major contributions. First, it provides a valuable addition to our set of tools to study such multiagent learning algorithms, and also strengthens evolutionary game theory as a primary framework for such analysis. Second, it adds to our understanding of the causes for multiagent learning algorithms drifting away from globally optimal solutions.

Next, we extend the EGT models from Section 2.3 such that each learner uses only the maximum of N rewards (it ignores the lower $N-1$ rewards) to improve its utility estimate. The following theorem establishes a key result for the formal model of lenient learners.

Theorem 1 *Let $(a_{ij})_{i,j=1}^{n}$ be the payoff matrix (a_{ij} is the payoff received by the agents when one performs action i and the other performs action j), and let $(p_j)_{j\in 1..n}$ be the probability that the teammate selects action j. The expected maximum payoff for i over N pairwise combinations with actions $j_1 \ldots j_N$ chosen with replacement according to $(p_j)_{j\in 1..n}$ is*

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

Proof

The expected maximum reward of action i over N pairwise combinations can be expressed as a weighted sum of all possible rewards a_{ij} that action i can receive with different actions j for the teammate. The weight of each term a_{ij} equals the probability that a_{ij} is the maximum reward observed over N trials. This probability can be computed based on the difference between the probability of receiving N rewards that are no higher than a_{ij}, which equals $\left(\sum_{k:a_{ik}\leq a_{ij}} p_k\right)^N$, minus the probability of receiving N rewards that are strictly lower than a_{ij}, which equals $\left(\sum_{k:a_{ik}<a_{ij}} p_k\right)^N$. Observe that an action i might receive the same reward in combination with different actions of the teammate, and as a result, there is an extra weight $\frac{p_j}{\sum_{k:a_{ik}=a_{ij}} p_k}$ that computes the probability of observing the reward a_{ij} in combination with action j of the teammate, out of all the other actions the teammate might have selected and would have resulted in the same reward. ■

Using Theorem 1, let us compute the expected reward for a lenient learner that ignores the lower of two observed rewards for each of its actions. We assume again that the teammate selects actions at random. The expected rewards in the Climb coordination problem are:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 11*\tfrac{5}{9} - 30*\tfrac{1}{9} + 0*\tfrac{1}{3} &&= 2.78\\
ExpectedReward(a_2) = &\quad -30*\tfrac{1}{9} + 7*\tfrac{5}{9} + 6*\tfrac{1}{3} &&= 2.56\\
ExpectedReward(a_3) = &\quad 0*\tfrac{2}{9} + 0*\tfrac{2}{9} + 5*\tfrac{5}{9} &&= 2.78
\end{aligned}$$

Observe that a_1 has the highest reward (tied with that of a_3). Were the agent to be more lenient (for example, if it ignored the lower two of three observed rewards), the expected reward for a_1 would be significantly higher than those for a_2 and a_3. Similarly, the expected rewards in the Penalty domain (with learners ignoring the lower of every two rewards) are:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 10*\tfrac{5}{9} + 0*\tfrac{1}{3} - 10*\tfrac{1}{9} &&= 4.44\\
ExpectedReward(a_2) = &\quad 0*\tfrac{2}{9} + 2*\tfrac{5}{9} + 0*\tfrac{2}{9} &&= 1.11\\
ExpectedReward(a_3) = &\quad -10*\tfrac{1}{9} + 0*\tfrac{1}{3} + 10*\tfrac{5}{9} &&= 4.44
\end{aligned}$$

Note that the Penalty game has two global optima, and that the lenient agents might have difficulties choosing both the same optimum. As we will show in this paper, lenience can help multiagent learning algorithms converge to the global optimum in domains with a unique such global optimum, as well as in some domains with multiple optima.

The results above support the hypothesis that lenient learners might benefit from estimating the expected reward for an action based on how good that action might be once the teammate has learned a better behavior. However, the analysis so far relied on the assumption that the teammate only chooses random actions. The EGT frameworks for multiagent learning, as described in Section 2.3, clearly eliminate this assumption by explicitly keeping track of the agents' probabilities of selecting each action over time, and by using that information to describe the learning process of each agent. The paper continues with extensions of these models to account for lenience: Section 4 demonstrates the benefits of lenience for cooperative coevolutionary algorithms, while Section 5 presents evidence for the advantages of lenience in multiagent Q-learning algorithms.

4. Enhanced Evolutionary Game Theory Model for Cooperative Coevolutionary Algorithms

As described in Section 2.3.1, the EGT model for CCEAs updates the utility of an action based on the expected reward for that action in combination with all possible actions that the teammate might select (Equations 5 – 6). Next, we use Theorem 1 to enhance this model to account for lenient learners.

Assessing fitness as the maximum of multiple rewards has been used before in practical cooperative coevolutionary algorithms (for example, in Wiegand et al., 2001). The research here contributes a formal analysis of this approach, which has so far been lacking.

Definition 1 *The EGT model in Equations 18 – 21 represents a theoretical model for a cooperative coevolutionary algorithm where the fitness of an individual is computed as the maximum reward with N collaborators.*

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} \frac{y_j^{(t)}}{\sum_{k:a_{ik}=a_{ij}} y_k^{(t)}} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} y_k^{(t)} \right)^N \right), \tag{18}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} \frac{x_i^{(t)}}{\sum_{k:a_{kj}=a_{ij}} x_k^{(t)}} \left(\left(\sum_{k:a_{kj}\leq a_{ij}} x_k^{(t)} \right)^N - \left(\sum_{k:a_{kj}<a_{ij}} x_k^{(t)} \right)^N \right), \tag{19}$$

$$x_i^{(t+1)} = \left(\frac{u_i^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}} \right) x_i^{(t)}, \tag{20}$$

$$y_j^{(t+1)} = \left(\frac{w_j^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}} \right) y_j^{(t)}. \tag{21}$$

Note that the enhanced model is equivalent to Wiegand's standard EGT model in Equations 5 – 8 when a single collaborator is used ($N = 1$). As demonstrated next, CCEAs are expected to achieve better performance as N increases. The proof that is the case relies on the following lemmas which

establish bounds on the probabilities that the initial populations have extremely large or small proportions of certain genotypes.

Lemma 1 *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*

$$P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_\varepsilon\right) < \varepsilon, \tag{22}$$

$$P\left(\max_{i=1..n} x_i^{(0)} \geq 1-\theta_\varepsilon\right) < \varepsilon, \tag{23}$$

$$P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_\varepsilon\right) < \varepsilon, \tag{24}$$

$$P\left(\max_{j=1..m} y_j^{(0)} \geq 1-\theta_\varepsilon\right) < \varepsilon. \tag{25}$$

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 that between 1 and the largest number) as the coordinates for the point.

It follows that $\min_{i=1..n} x_i^{(0)}$ is the smallest distance between two such numbers (and possibly the boundaries 0 and 1). Similarly, $\max_{i=1..n} x_i^{(0)}$ is the largest distance between two numbers.

Suppose $\gamma > 0$ is a small number. We 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 22 and 24 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 23, 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 25 is similar. ■

Lemma 1 basically proved that the probability that an agent starts with an extremely low or an extremely high probability of selecting an action is very low, given a random initialization of the multiagent learning process. Following, Lemma 2 derives a complementary result, namely that each agent should have a fair likelihood of selecting each of its actions when the learning process starts.

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* $\eta_\varepsilon > 0$ *such that*

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

Proof We apply Lemma 1 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 1. 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, we prove that the revised EGT model described by Equations 18−21 converges to the global optimum with an arbitrarily-high probability.

Theorem 2 *Given a joint reward matrix A with a unique global optimum $a_{i^\star j^\star}$, for any $\varepsilon > 0$ there exists $N_\varepsilon \geq 1$ such that the theoretical CCEA model in Equations 18−21 converges to the global optimum with probability greater than $(1-\varepsilon)$ for any number of collaborators N such that $N \geq N_\varepsilon$.*

Proof

ε only serves as a guarantee for the worst case scenario for the proportions of individuals in the initial populations. From Lemma 2, it follows that there exists $\eta_\varepsilon > 0$ such that with probability at least $1-\varepsilon$, it is the case 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.

We 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, proving only the first (by refining Equation 18) is sufficient:

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

Next, we identify 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 $\sum_{k=1}^{m} y_k^{(t)} = 1$, the previous inequality can be further refined:

$$
\begin{aligned}
u_{i^\star}^{(t)} &\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(1 - y_{j^\star}^{(t)} \right)^N \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \left(1 - y_{j^\star}^{(t)} \right)^N \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \\
&= a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t)} \right)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \right). \qquad (26)
\end{aligned}
$$

The inequalities $\eta_\varepsilon < y_{j^\star}^{(0)} < 1-\eta_\varepsilon$ hold for the initial populations, as inferred earlier from Lemma 2. It follows from Equation 26 that

$$
u_{i^\star}^{(0)} \ge a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \right). \qquad (27)
$$

However,

$$\lim_{N\to\infty} a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}<0} a_{i^\star j} \right) \quad = \quad a_{i^\star j^\star} \tag{28}$$

Given that $a_{i^\star j^\star} > \alpha$, Equation 28 implies that there exists $N_1 \geq 1$ such that

$$a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}<0} a_{i^\star j} \right) > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha \tag{29}$$

for all $N \geq N_1$. From Equations 26 and 29, it follows that

$$u_{i^\star}^{(0)} > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha \tag{30}$$

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

$$w_{j^\star}^{(t)} \geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right), \tag{31}$$

$$w_{j^\star}^{(0)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right) \tag{32}$$

and that there exists $N_2 \geq 1$ such that

$$a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right) > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha$$

which implies that

$$w_{j^\star}^{(0)} > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha$$

for all $N \geq N_2$.

Let $N_\varepsilon = max(N_1, N_2)$, and let $N \geq N_\varepsilon$. Next, we show by induction by t (the number of iterations of the model, that is, the number of generations) that the following four inequalities hold (note that the last two imply a monotonic increase in the actions that comprise the global optimum):

$$u_{i^\star}^{(t)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}<0} a_{i^\star j} \right), \tag{33}$$

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

$$x_{i^\star}^{(t+1)} \geq x_{i^\star}^{(t)}, \tag{34}$$

$$y_{j^\star}^{(t+1)} \geq y_{j^\star}^{(t)}. \tag{35}$$

At the first generation ($t = 0$), the first two inequalities hold (Equations 27 and 32). 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, Equation 20 implies that

$$\begin{aligned} x_{i^\star}^{(1)} &= \left(\frac{u_{i^\star}^{(0)}}{\sum_{k=1}^{n} x_k^{(0)} u_k^{(0)}} \right) x_{i^\star}^{(0)} \\ &> \left(\frac{u_{i^\star}^{(0)}}{\sum_{k=1}^{n} x_k^{(0)} u_{i^\star}^{(0)}} \right) x_{i^\star}^{(0)} \\ &= x_{i^\star}^{(0)}. \end{aligned}$$

and Equation 21 similarly implies that

$$\begin{aligned} y_{j^\star}^{(1)} &= \left(\frac{w_{j^\star}^{(0)}}{\sum_{k=1}^{m} y_k^{(0)} w_k^{(0)}} \right) y_{j^\star}^{(0)} \\ &> \left(\frac{w_{j^\star}^{(0)}}{\sum_{k=1}^{m} y_k^{(0)} w_{j^\star}^{(0)}} \right) y_{j^\star}^{(0)} \\ &= y_{j^\star}^{(0)}. \end{aligned}$$

To prove the inductive step, it follows from Equation 26 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,

$$\begin{aligned} x_{i^\star}^{(t+1)} &= \left(\frac{u_{i^\star}^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}}\right) x_{i^\star}^{(t)} \\ &> \left(\frac{u_{i^\star}^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_{i^\star}^{(t)}}\right) x_{i^\star}^{(t)} \\ &= x_{i^\star}^{(t)}. \end{aligned}$$

Similarly, Equation 31 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,

$$\begin{aligned} y_{j^\star}^{(t+1)} &= \left(\frac{w_{j^\star}^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}}\right) y_{j^\star}^{(t)} \\ &> \left(\frac{w_{j^\star}^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_{j^\star}^{(t)}}\right) y_{j^\star}^{(t)} \\ &= y_{j^\star}^{(t)} \end{aligned}$$

Having shown that both $\left(x_{i^\star}^{(t)}\right)_t$ and $\left(y_{j^\star}^{(t)}\right)_t$ are monotonically increasing (Equations 34–35), and given that they are bounded between 0 and 1, it follows that they converge to some value. Let $\bar{x} = \lim_{t\to\infty} x_{i^\star}^{(t)}$. We used Lemma 2 to define $\eta_\varepsilon > 0$ such that $\eta_\varepsilon < x_i^{(0)}$ (among other inequalities). This clearly implies $\bar{x} > 0$, as $\left(x_{i^\star}^{(t)}\right)_t$ is monotonically increasing. We also have that

$$\begin{aligned} \frac{x_{i^\star}^{(t+1)}}{x_{i^\star}^{(t)}} &= \frac{u_{i^\star}^{(t)}}{\sum_{k=1}^{n} u_k^{(t)} x_k^{(t)}} \\ &\geq \frac{u_{i^\star}^{(t)}}{u_{i^\star}^{(t)} x_{i^\star}^{(t)} + \alpha\left(1 - x_{i^\star}^{(t)}\right)} \\ &= 1 + \frac{\left(u_{i^\star}^{(t)} - \alpha\right)\left(1 - x_{i^\star}^{(t)}\right)}{u_{i^\star}^{(t)} x_{i^\star}^{(t)} + \alpha\left(1 - x_{i^\star}^{(t)}\right)}. \end{aligned}$$

But $\lim_{t\to\infty} \frac{x_{i^\star}^{(t+1)}}{x_{i^\star}^{(t)}} = 1$, and therefore

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

which implies

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

Finally, Equations 30 and 33 imply that $u_{i^\star}^{(t)} - \alpha > \frac{a_{i^\star j^\star}}{2} - \frac{\alpha}{2} > 0$. As a consequence $\lim_{t\to\infty} x_{i^\star}^{(t)} = 1$. The proof that $y_{j^\star}^{(t)}$ also converges to 1 is similar and is omitted for brevity. It follows that $x_i^{(t)}$ and $y_j^{(t)}$ converge to 0 for all $i \neq i^\star$ and for all $j \neq j^\star$. ■

Theorem 2 shows that CCEAs will converge to the global optimum in domains with a unique such optimum, if set appropriately and if given enough resources. This proves that the earlier claims of uncontrolled drift to suboptimal solutions, as reported in Wiegand (2004), were entirely due to the use of a formal model that employed a simplified fitness assessment mechanism. The term "enough" is used here to indicate two types of resources that CCEAs need: large populations to explore the space (the populations are in fact assumed to be infinite in our formal model, for simplicity), and large numbers of collaborators to provide accurate fitness estimates for all populations. This paper does not go into great depths with respect to the requirement for large populations. As for the second requirement, Section 4.1 analyzes the impact of the number of collaborators onto the performance of the CCEA.

Observe also that Theorem 2 does not cover domains with multiple optima, such as the Penalty domain discussed in Section 3. Next, we use a visualization technique to study the impact of lenience onto cooperative coevolutionary algorithms as applied to domains with one optimum, as well as to domains with multiple global optima.

4.1 Basins of Attraction due to Estimations

This section describes an intuitive way to visualize the impact of improved estimates on the performance of the system. Given specific initial populations, the EGT model is completely deterministic. As shown by Wiegand (2004), the populations are expected to converge to Nash equilibria in the payoff matrix. As a result, it is straightforward to provide two-dimensional visualizations of the basins of attraction for different Nash equilibria when each population contains only two genotypes: the two-dimensional location of a pixel uniquely encodes the initial ratios of one of the genotypes for each population, and the color of the pixel specifies the equilibrium to which the system converges after repeated iteration (see Panait et al., 2004, for details).

However, such simple problem domains present a limited range of challenges to cooperative coevolutionary algorithms. This section will instead illustrate the basins of attraction of Nash equilibria in two 3×3 coordination games: Climb and Penalty. This domains can stress the challenges posed by poor estimates of fitness, as discussed in Section 3.

Given that fitness proportional selection works properly only when all individuals have positive fitness, we add 30 to all values in the Climb payoff matrix, and we add 10 to all values in the Penalty

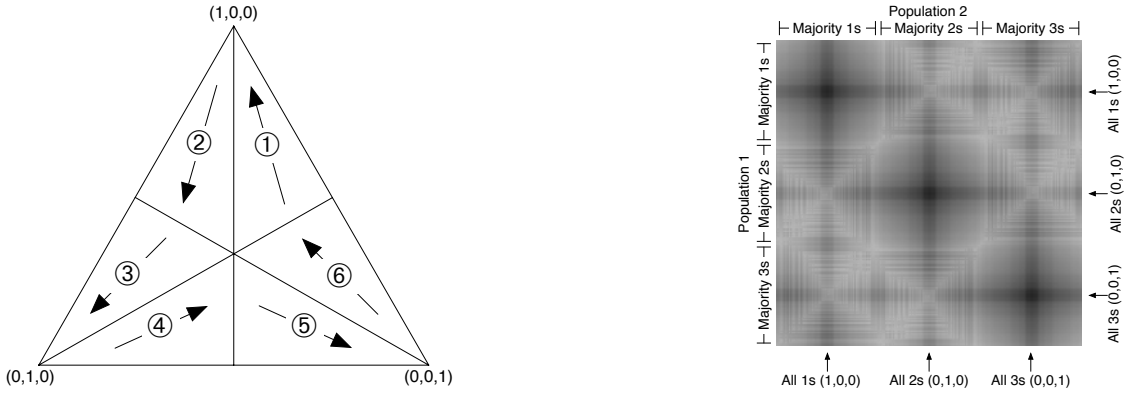

Figure 2: The projection of $\Delta^3 \times \Delta^3$ to $[0,1]^2$. (a): The projection divides the simplex Δ^3 into six equal-area triangles; arrows show the direction for sorting points in each area. (b): Visualization of the cartesian product of two simplexes.

payoff matrix. In contrast, in the next Section (Section 5), all experiments will employ the original payoff matrices.

We apply the EGT model in Equations $18-21$ to each of these two problem domains. The model is iterated for 100000 generations or until the proportion of one action in each population exceeds a threshold of $1-10^{-10}$. A specific color identifies each of the possible end results (Nash equilibria). For consistency, black dots always indicate convergence to a suboptimal solution, while white and grey dots indicate convergence 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 objective of this process is to group together regions of the space of initial populations that are expected to converge to the same Nash equilibrium. The projection of Δ^3 to one dimension starts by dividing it into six equal-area triangles, as shown in Figure 2a. 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.

We sample 216 initial populations in the simplex: the six areas in Figure 2a are each divided 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. We add random noise distributed uniformly between -0.00005 and 0.00005 to reduce certain artifacts due to identical distributions of the two populations (see the discussion at the end of this section about the thin diagonal line in Figure 4). 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 negligibly small.

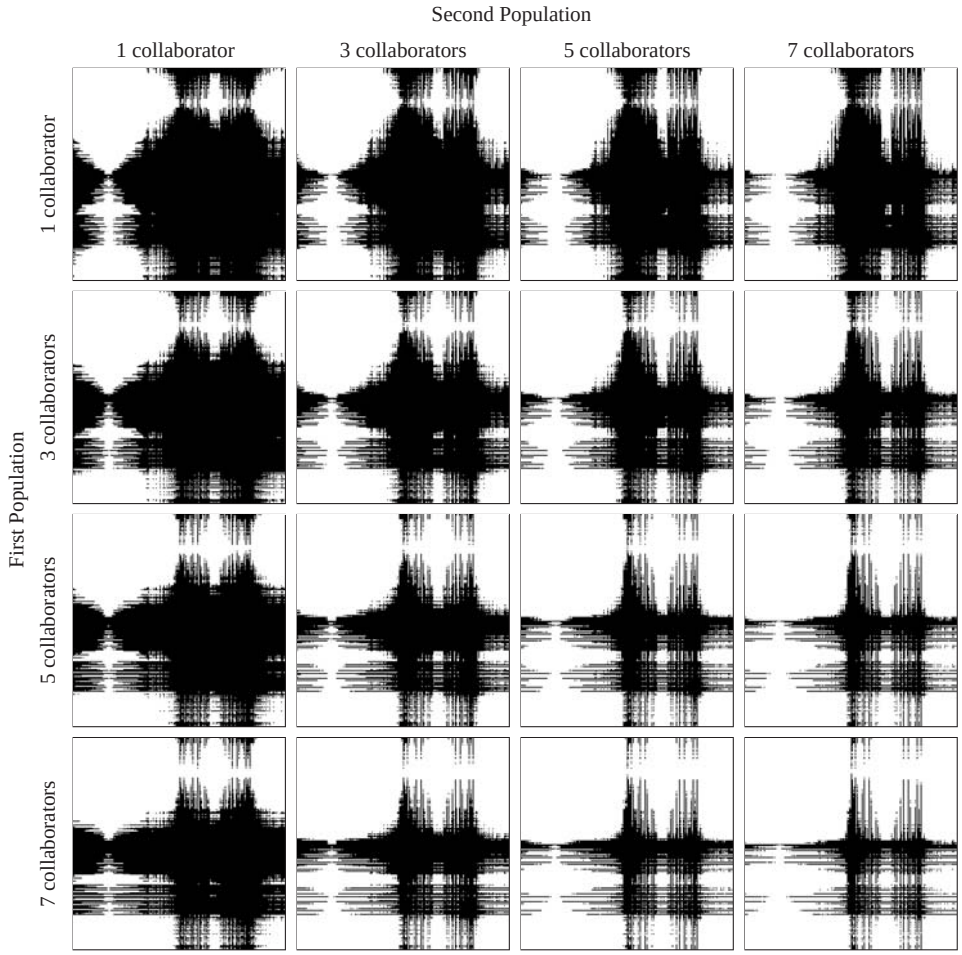

Figure 3: Basins of attraction for CCEA in the Climb problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria, respectively.

The right image in Figure 2 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 this figure show locations that have high ratios of 1s, 2s, or 3s in one or the other population.

First, Figure 3 visualizes the impact of improved estimates due to increased numbers of collaborators for each population (to parallel Theorem 2). The images shows the basins of attraction in the Climb coordination game. Observe 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 optimum 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

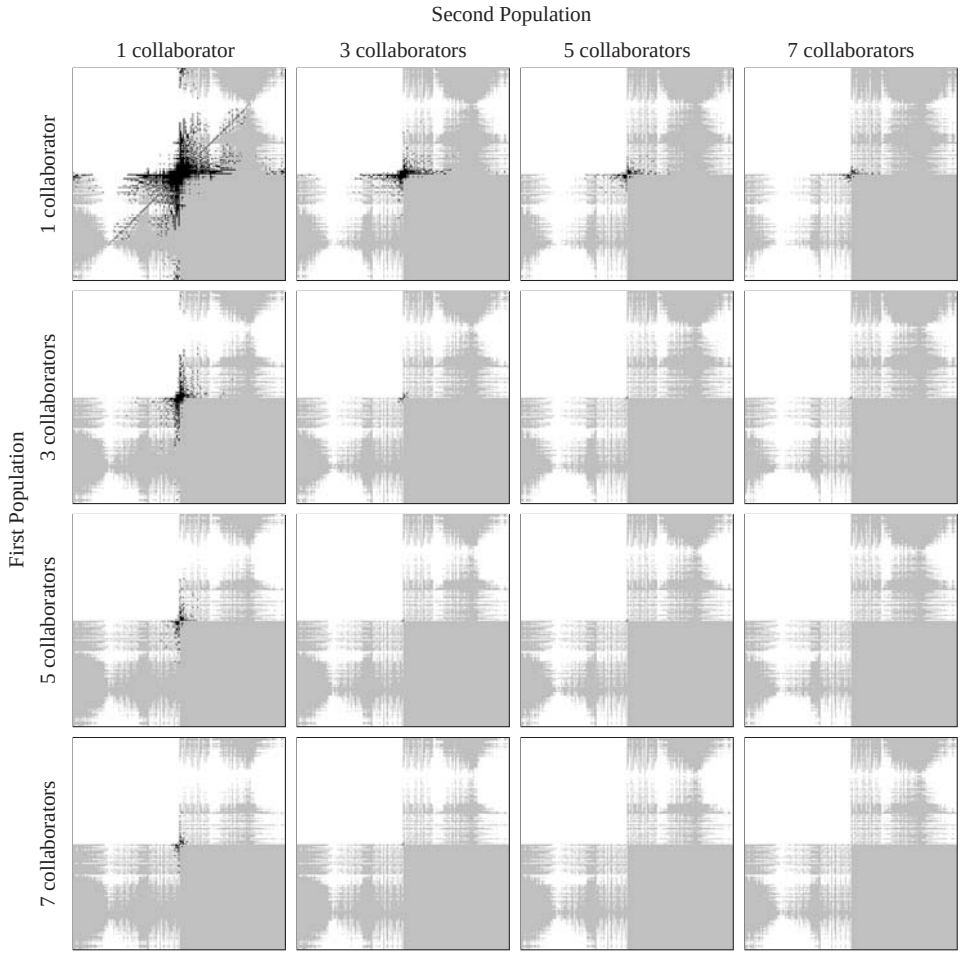

Figure 4: Basins of attraction for CCEA in the Penalty problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

better estimate of fitness (via an increased number of collaborators), the basin of attraction for the suboptimal equilibrium 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 of 2s or 3s required to still converge to the sub-optimum.

Figure 4 presents the basins of attraction in the Penalty game. Note that the Penalty game has two global optima and thus does not fit the hypothesis of the formal proof in Theorem 2. The basins of attraction illustrated in Figure 4 indicate that lenient learners exhibit good performance in some games with multiple global optima as well. What precise characteristics of the coordination game pose difficulties to lenient learners is the focus of ongoing analysis.

Observe that the two global optima cover most of the space in Figure 4 even when a single collaborator is used; the suboptimal equilibria covers mainly areas where at least one of the pop-

ulation 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 solely 2s. The visualization of the basins of attraction suggests that Penalty is a much easier coordination game than Climb. Note also the thin diagonal line in the top-left graph of Figure 4. 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.

The visualization of the basins of attraction to Nash equilibria provided an intuitive approach to grasping the impact that lenience has onto cooperative coevolutionary algorithms. Figures 3 – 4 showed that the number of trajectories for CCEAs that converge to suboptimal solutions decreases significantly across both problem domains, as the level of lenience increases. Next, we apply this visualization technique to show that lenience is not specific to only cooperative coevolution, but it can be used to improve the performance of other multiagent learning algorithms as well.

5. Evolutionary Game Theory Models for Lenient Multiagent Q-Learning

The RD model in Equations 14–17 assumes that the agent will update the utility of an action based on the average reward it expects to receive for that action. This approach is therefore similar to the one used by the EGT model of cooperative coevolutionary algorithms in Equations 5–8. As argued in Section 3 and demonstrated in Section 4 (for cooperative coevolutionary algorithms only), using the average reward usually results in poor estimates for the quality of actions, causing potential attraction to towards suboptimal solutions.

To remedy this situation, we extend the RD model such that each learner ignores lower rewards to improve its estimate. The key to this extension is in Theorem 1 in Section 3. The following RD model is a straightforward combination of these previous results:

$$u_i = \sum_{j=1}^{m} \frac{a_{ij} y_j \left(\left(\sum_{k:a_{ik} \leq a_{ij}} y_k \right)^N - \left(\sum_{k:a_{ik} < a_{ij}} y_k \right)^N \right)}{\sum_{k:a_{ik}=a_{ij}} y_k}, \tag{36}$$

$$w_j = \sum_{i=1}^{n} \frac{a_{ij} x_i \left(\left(\sum_{k:a_{kj} \leq a_{ij}} x_k \right)^N - \left(\sum_{k:a_{kj} < a_{ij}} x_k \right)^N \right)}{\sum_{k:a_{kj}=a_{ij}} x_k}, \tag{37}$$

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau} \left(u_i - \sum_k x_k u_k \right) + \alpha \sum_k x_k \ln \frac{x_k}{x_i}, \tag{38}$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau} \left(w_j - \sum_k y_k w_k \right) + \alpha \sum_k y_k \ln \frac{y_k}{y_j}. \tag{39}$$

Note that the two equations describing the update rule for the two learners have not changed. What has changed, however, is the expected reward that is used to update the utilities of actions Equations 36 and 37). As expected, observe that the two RD models described by Equations 14–17 and 36–39 are equivalent when $N = 1$ (that is, when the agents do not ignore any low rewards). For

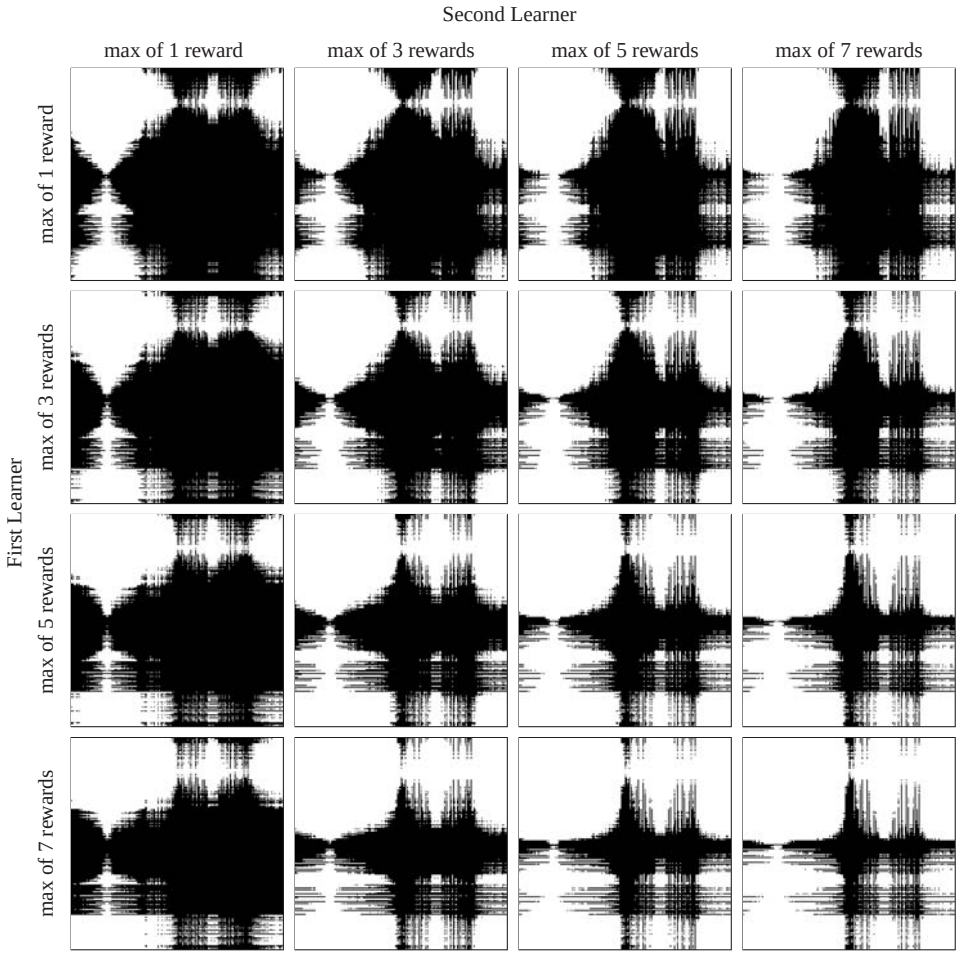

Figure 5: Basins of attraction in the Climb problem domain for multiagent Q-learning that updates the utility of an action based on the maximum of 1, 3, 5 and 7 of the rewards received when selecting that action. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

this reason, the setting $N = 1$ constitutes a benchmark representing the formal model for traditional Q-learners.

5.1 Basins of Attraction to Nash Equilibria for Multiagent Q-Learning

Given that the RD model provides differential equations, the next state of the concurrent learning system can be approximated by assuming the variations in the derivative are small over short periods of time θ:

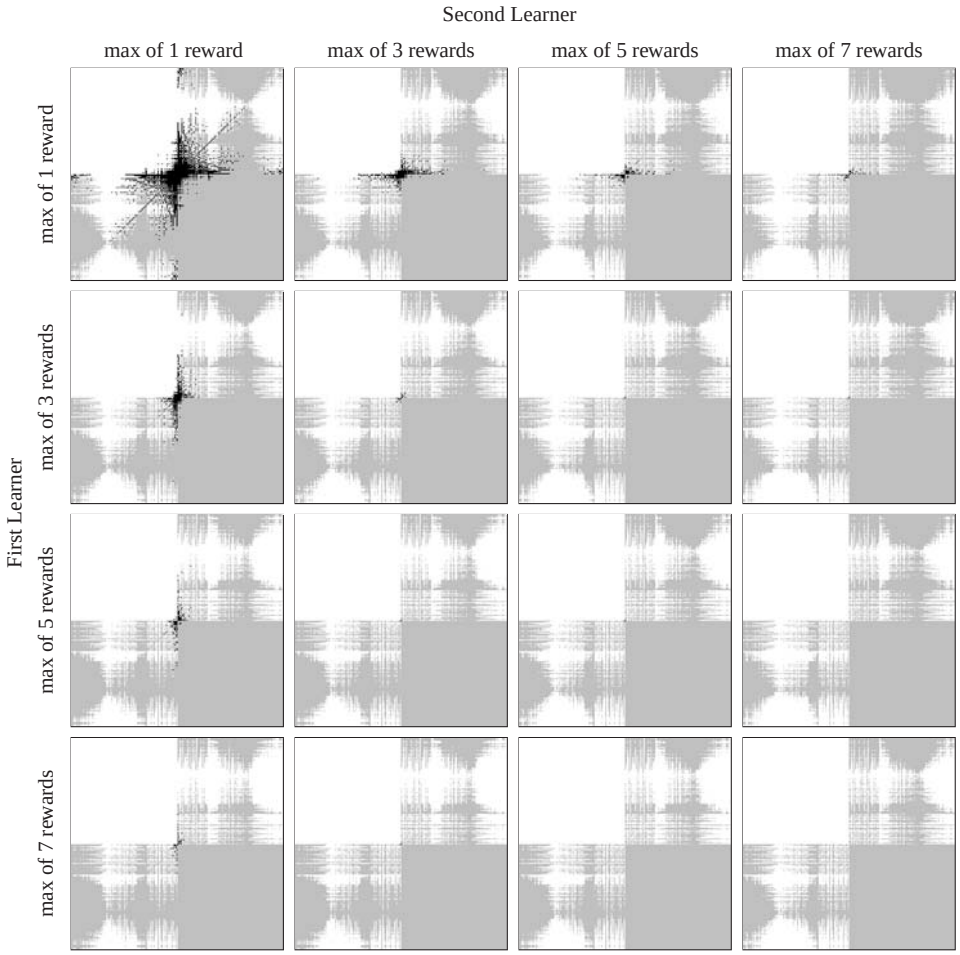

Figure 6: Basins of attraction in the Penalty problem domain for multiagent Q-learning that updates the utility of an action based on the maximum of 1, 3, 5 and 7 of the rewards received when selecting that action. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

$$\begin{aligned} x_i' &= x_i + \theta \frac{dx_i}{dt}, \\ y_j' &= y_j + \theta \frac{dy_j}{dt}. \end{aligned}$$

Experiments used the following settings: $\theta = 0.001$, the learning rate $\alpha = 0.1$, and the exploration parameter $\tau = 0.01$. We iterate the RD model 100000 times, or until both agents have a probability exceeding $1 - 10^{-10}$ to select one of their actions over the other two.

The basins of attraction for the optimal $(1,1)$ (both play their first action) and suboptimal $(2,2)$ (both play their second action) equilibria in the Climb domain are visualized in Figure 5. Given

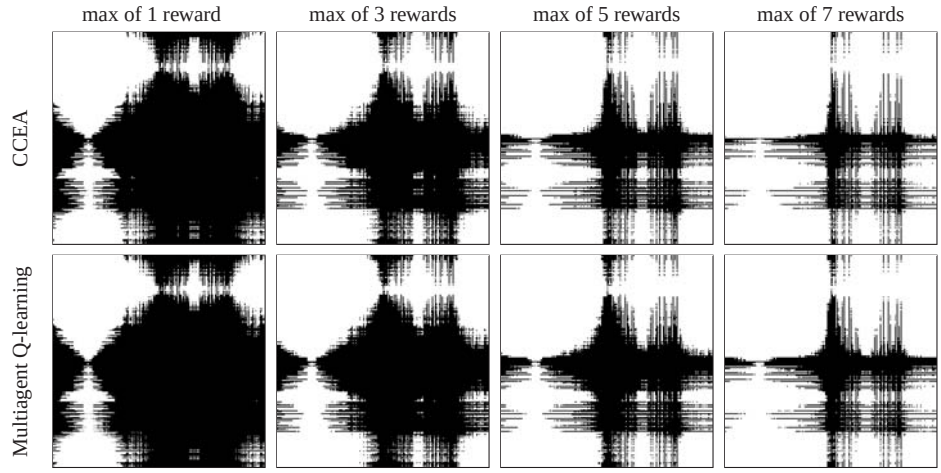

Figure 7: Basins of attraction in the Climb problem domain for cooperative coevolution and multiagent reinforcement learning, both ignoring all but the maximum of 1, 3, 5, and 7 rewards. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

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 suggests that a straightforward extension of Q-learning to multiagent domains may converge to the suboptimal solution when started from many initial conditions. 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 6.

Note that multiagent Q-learning improves with more lenience in nearly an identical (but not *quite* identical) fashion to coevolution. Figure 5 shows the basins of attraction in the Climb coordination game. Once again, the images show that the difficulty of the problem domain decreases as each population is provided with more accurate estimates for the utilities of actions. Similarly, Figure 6 presents the basins of attraction in the Penalty game. As was the case for coevolution, increasing numbers of collaborators are able to essentially eliminate convergence to the suboptimal equilibrium. Also, Q-learning likewise has a thin diagonal line of suboptimal convergence in the top-left graph of Figure 6, which once again is due to one agent selecting action 1 in equal proportions to the second agent selecting action 3, and vice-versa.

6. Discussion and Conclusions

At first glance one would think that coevolution and multiagent Q-learning, being fairly different algorithms, with fairly different settings and aimed at different kinds of problems, would respond differently in these scenarios. However, not only do they perform nearly identically in the "classic" case (no lenience), but they also respond in nearly the same fashion when subjected to varying degrees of lenience as well. To highlight this similarity, Figures 7 and 8 sample the diagonal graphs in Figures 3 – 5 and Figures 4 – 6: these are the cases where both agents ignore the same number of lower rewards. This is not really that surprising however: as the theoretical analysis has shown,

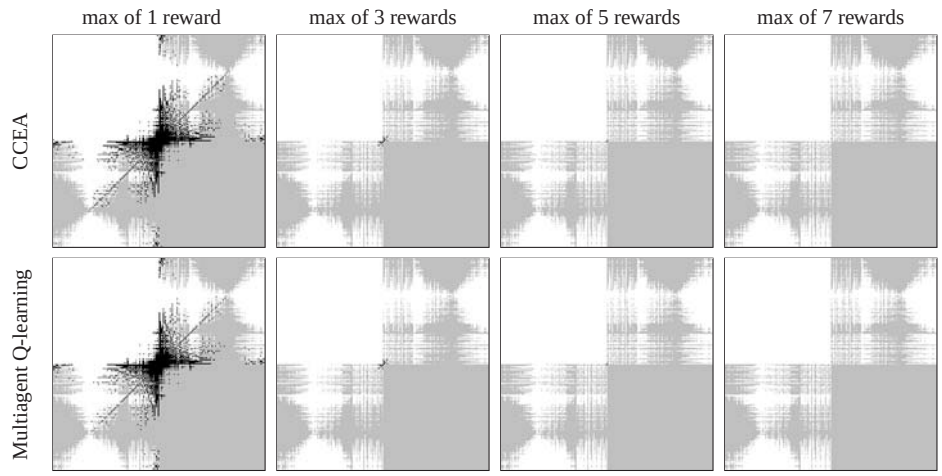

Figure 8: Basins of attraction in the Penalty problem domain for cooperative coevolution and multiagent reinforcement learning, both ignoring all but the maximum of 1, 3, 5, and 7 rewards. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

the dynamics of these equations is quite similar; so much so, indeed, that each of these respective communities would do well to more closely examine the existing work of the other.

Keep in mind that both evolutionary algorithms and Q-learning are guaranteed to converge to the global optimum in single-agent scenarios, if properly set and if given enough resources. We have argued that the attraction of these algorithms towards suboptimal solutions is caused primarily by poor estimates, and that it can be dealt with if learners simply ignore lower rewards. Figures 7 and 8 suggest that if the multiagent learning algorithm is heading towards suboptimal solutions, it is usually caused by the estimation procedure for the quality of an action of the learning algorithm. These algorithms do not start to suffer from undesirable pathologies just because they are used in multiagent scenarios. Rather, poor estimates are the primary cause for the suboptimal solutions. Ignoring lower rewards improves the learners' estimate for the quality of their behaviors, resulting in an increased probability that the multiagent learning system will converge to the global optimum.

In summary, this paper presented an extended formal model for a new class of multiagent learning algorithms, namely those involving lenient agents that ignore low rewards observed upon performing actions in the environment. The paper also illustrated the basins of attraction to different optimal and suboptimal Nash equilibria, and showed how intuitive graphs might reveal valuable information about the properties of multiagent learning algorithms that was lacking in the summarizations of empirical results. The paper provided theoretical support for previous reports that lenience helps learners achieve higher rates of convergence to optimal solutions, and also proved that properly-set lenient learners are guaranteed to converge to the Pareto-optimal Nash equilibria in coordination games. Finally, the results strengthened the use of Evolutionary Game Theory to study the properties of multiagent reinforcement learning algorithms.

This work has also opened many avenues of future research. We plan to extend these formal models to help analyze the properties of multiagent learning algorithms in more complex domains,

such as those involving more actions and players, stochasticity, multiple states, and partial observability. We hope that this formal analysis will provide the foundation for new learning algorithms guaranteed to converge to optimal solutions.

Acknowledgments

The authors would like to thank R. Paul Wiegand, Dana Richards, Keith Sullivan, Dominic Mazzoni, and the anonymous reviewers for helpful discussions and suggestions.

References

Adrian K. Agogino and Kagan Tumer. Evolving distributed agents for managing air traffic. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1888 – 1895, 2007.

Adrian K. Agogino and Kagan Tumer. Handling communication restrictions and team formation in congestion games. *Journal of Autonomous Agents and Multi Agent Systems*, 13(1):97–115, 2006.

Stephane Airiau, Kagan Tumer, and Adrian K. Agogino. Learning agents for distributed and robust spacecraft power management. In *AAMAS-06 Workshop on Adaptation and Learning in Autonomous Agents and Multi Agent Systems*, 2006.

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

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

Roger Eriksson and Björn Olsson. Cooperative coevolution in inventory control optimisation. In G. Smith, N. Steele, and R. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference, ICANNGA97*, 1997.

Nancy Fulda and Dan Ventura. Predicting and preventing coordination problems in cooperative learning systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2001.

Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Efficient non-linear control through neuroevolution. In *Lecture Notes in Computer Science (LNCS 4212)*, pages 654–662. 2006.

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

Phil Husbands and Frank Mill. Simulated coevolution as the mechanism for emergent planning and scheduling. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270, 1991.

Thomas Jansen and R. Paul Wiegand. Exploring the explorative advantage of the CC (1+1) EA. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 310–321, 2003.

Thomas Jansen and R. Paul Wiegand. The cooperative coevolutionary (1+1) ea. *Evolutionary Computation*, 12(4):405–434, 2004.

Kenneth De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.

Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-02)*, 2002.

Matt Knudson and Kagan Tumer. Effective policies for resource limited agents. In *Proceedings of the AAMAS-07 Workshop on Adaptive and Learning Agents Workshop (ALAg-07)*, 2007.

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, 2000.

John Maynard Smith and George R. Price. The logic of animal conflict. *Nature*, 146:15–18, 1973.

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

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*, 2005b.

Liviu Panait and Sean Luke. Selecting informative actions improves cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*, 2006.

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.

Liviu Panait, R. Paul Wiegand, and Sean Luke. A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature – PPSN-2004*, pages 892–901, 2004.

Liviu Panait, Keith Sullivan, and Sean Luke. Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*, 2006.

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

Mitchell Potter and Kenneth De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pages 249–257, 1994.

Mitchell Potter and Kenneth De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

Satinder P. Singh, Michael J. Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548, 2000.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

Kagan Tumer and Adrian K. Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 371–388. Springer, 2007.

Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. In *The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM Press, Melbourne, Australia*, 2003.

Karl Tuyls, Pieter Jan 't Hoen, and Bram Vanschoenwinkel. An evolutionary dynamical analysis of multi-agent learning in iterated games. *The Journal of Autonomous Agents and Multi-Agent Systems*, 12:115–153, 2006.

Michael Vose. *The Simple Genetic Algorithm*. MIT Press, 1999.

Christopher J. Watkins. *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge University, Cambridge, United Kingdom, 1989.

Christopher J. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Jorgen W. Weibull. *Evolutionary Game Theory*. MIT Press, 1996.

R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2004.

R. Paul Wiegand and Mitchell Potter. Robustness in cooperative coevolution. In *GECCO '06: Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation*, pages 369–376, 2006.

R. Paul Wiegand, William Liles, and Kenneth De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1235–1242, 2001.

R. Paul Wiegand, William Liles, and Kenneth De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In *Foundations of Genetic Algorithms VII*, pages 231–248, 2002.