Improving Coevolutionary Search for Optimal Multiagent Behaviors

Liviu Panait
Department of Computer Science
George Mason University
Fairfax, VA 22030
lpanait@cs.gmu.edu

R. Paul Wiegand
Krasnow Institute
George Mason University
Fairfax, VA 22030
paul@tesseract.org

Sean Luke
Department of Computer Science
George Mason University
Fairfax, VA 22030
sean@cs.gmu.edu

Abstract

Evolutionary computation is a useful technique for learning behaviors in multiagent systems. Among the several types of evolutionary computation, one natural and popular method is to coevolve multiagent behaviors in multiple, cooperating populations. Recent research has suggested that coevolutionary systems may favor stability rather than performance in some domains. In order to improve upon existing methods, this paper examines the idea of modifying traditional coevolution, biasing it to search for maximal rewards. We introduce a theoretical justification of the improved method and present experiments in three problem domains. We conclude that biasing can help coevolution find better results in some multiagent problem domains.

1 Introduction

Multi-agent learning is an area of intense research, and is challenging because the problem dynamics are often complex and fraught with local optima. These difficulties have made evolutionary computation (EC) an attractive approach to learning multiagent behaviors (for example, [Iba, 1996; Luke *et al.*, 1998; Wu *et al.*, 1999; Bull *et al.*, 1995; Bassett and De Jong, 2000; Bull, 1997]). This work has led to interesting research questions in applying EC in a multiagent setting, including communication, representation, generalization, teamwork, and collaboration strategies.

As it is very general (and relatively knowledge-poor), evolutionary computation is particularly useful in problems that are of high dimensionality, are non-Markovian, or yield few heuristic clues about the search space that otherwise would make reinforcement learning or various supervised learning methods good choices. We believe that multiagent learning problem domains often exhibit such features. These problem domains are often complex and "correct" actions cannot be known beforehand in a given situation. Further, even relatively simple problems can require large numbers of external and, even more challenging, internal state variables. Last, many such problems exhibit changing environments, even ones that adapt to make the problem harder for the learner (due to the presence of co-learning opponents).

EC fits nicely with multiagent systems because it is already *population-oriented*: it searches over a set of multiple agents (the individuals). Further, an EC population may be broken down into distinct subpopulations, each yielding agents to be tested together in a multiagent environment, with each subpopulation "evolving" in parallel. This notion of separately evolving, interacting populations of agents is known as *coevolution*. Coevolution has proven a useful technique for multiagent problems where the quality of agent is typically assessed in the context of competing or cooperating peers.

But coevolution is no panacea. Recent research has shown that a coevolutionary system does not necessarily search for better teams of agents, but can instead search for agent populations that represent *stable equilibria* in the cooperative search space [Ficici and Pollack, 2000; Wiegand *et al.*, 2002b]. This paper will explore this problem, then introduce a method for biasing coevolution so that the search for stability coincides with optimization for improvement.

We continue this paper with a brief description of coevolution and present an experimental and a theoretical framework. We then suggest a method for biasing the coevolutionary process, describe a theoretical investigation on how biasing modifies the search space, and discuss experimental results on three problem domains. The paper ends with a set of conclusions and directions for future work.

2 Evolutionary Computation and Coevolution

Evolutionary computation is a family of techniques, known as *evolutionary algorithms*, widely used for learning agent behaviors. In EC, abstract Darwinian models of evolution are applied to refine populations of agents (known as individuals) representing candidate solutions to a given problem. An evolutionary algorithm begins with an initial population of randomly-generated agents. Each member of this population is then evaluated and assigned a fitness (a quality assessment). The EA then uses a fitness-oriented procedure to select agents, breeds and mutates them to produce child agents, which are then added to the population, replacing older agents. One evaluation, selection, and breeding cycle is known as a generation. Successive generations continue to refine the population until time is exhausted or a sufficiently fit agent is discovered.

Coevolutionary algorithms (CEAs) represent a natural approach to applying evolutionary computation to refine multiagent behaviors. In a CEA, the fitness of an individual is based on its interaction with other individuals in the population: thus the fitness assessment is context-sensitive and subjective. In *competitive* systems, agents benefit at the expense of other agents; but in *cooperative* systems, agents succeed or fail together in collaboration. The focus of this paper is in cooperative coevolutionary algorithms. Interesting CEA issues include communication [Bull *et al.*, 1995], teamwork, and collaboration [Bull, 1997].

A standard approach [Potter, 1997] to applying cooperative coevolutionary algorithms (or CCEAs) to an optimization problem starts by identifying a static decomposition of the problem representation into subcomponents, each represented by a separate population of individuals. For example, if a task requires two agents whose collaboration must be optimized, one might choose to use two populations, one per agent in the task. The fitness of an individual in a population is then determined by testing the individual in collaboration with one or more individuals from the other population. Aside from this collaborative assessment, each population follows its own independent evolution process in parallel with other populations.

2.1 Formalizing the CCEA

An appealing abstract mathematical model for this system comes from the Biology literature: Evolutionary Game Theory (EGT) [Maynard-Smith, 1982; Hofbauer and Sigmund, 1998]. EGT provides a formalism based on traditional game theory and dynamical systems techniques to analyze the limiting behaviors of interacting populations under long-term evolution. For specifics about applying EGT to the analysis of multi-population cooperative coevolutionary algorithms, see [Wiegand *et al.*, 2002a].

In this paper, we consider only two-population models. In such a model, a common way of expressing the rewards from individual interactions is through a pair of *payoff matrices*. We assume a symmetric model such that when individuals from the first population interact with individuals from the second, one payoff matrix A is used, while individuals from the second population receive rewards defined by the transpose of this matrix (A^T). In our theoretical exploration of EGT in this paper, we will use an *infinite population*: thus a population can be thought of not as a set of individuals, but rather as a finite-length vector $\vec{x}$ of proportions, where each element in the vector is the proportion of a given individual configuration (popularly known as a *genotype* or, as we will term it, a *strategy*) in the population. As the proportions in a valid vector must sum to one, all legal vectors make up what is commonly known as the *unit simplex*, denoted Δ^n, where n here is the number of distinct strategies possible, $\vec{x} \in \Delta^n : x_i \in [0,1], \sum_{i=1}^{n} x_i = 1$.

Formally we can model the effects of evaluation and proportional selection over time using a pair of difference equations, one for each population. The proportion vectors for the two populations are $\vec{x}$ and $\vec{y}$ respectively. Neglecting the issue of mutation and breeding and concentrating only on the effects of selection, we can define the dynamical system of a two-population cooperative coevolutionary algorithm as:

$$\begin{aligned}
\vec{u} &= A\vec{y} && (1)\\
\vec{w} &= A^T\vec{x} && (2)\\
x_i' &= \left(\frac{u_i}{\vec{x}\cdot A\vec{y}}\right)x_i && (3)\\
y_i' &= \left(\frac{w_i}{\vec{y}\cdot A^T\vec{x}}\right)y_i && (4)
\end{aligned}$$

...where $\vec{x}'$ and $\vec{y}'$ represent the new population distributions for the next generation. Here it is assumed that an individual's fitness is assessed through pair-wise collaborations with *every* member of the cooperating population. We call this idea *complete mixing*. The equations above describe a two-step process. First, the vectors $\vec{u}$ and $\vec{w}$ are derived; these represent the fitness assessments of strategies in the generations $\vec{x}$ and $\vec{y}$ respectively. Note that an infinite population model considers the fitness assessment for a strategy, and not for a particular instance of that strategy (an individual). Then selection is performed by computing the proportion of the fitness of a specific strategy over the sum fitness of the entire population.

2.2 Optimization versus Balance

CCEA researchers apply these algorithms hoping to *optimize* the collaborations between the populations, but it isn't clear that this system is meant to do this. In fact, the system seeks a form of *balance* between strategies, which may not correspond with what we, as external viewers of the system, would consider optimal. In the context of a payoff matrix, an optimal position is the pair of strategies that yield the highest payoff for the cooperating agents. This position is a stable attracting fixed point of such a system; but it is also the case that there are other suboptimal points, which can also attract trajectories [Wiegand *et al.*, 2002b]. Indeed, it is possible that most, if not all, trajectories can be pulled toward suboptimal spots. These points correspond to Nash equilibria: suboptimal combinations of strategies where if any *one* strategy is changed, the net reward for both agents will decrease.

As a result, individuals in a CCEA are not necessarily refined to be the optimal subcomponent of the optimal component; instead they are refined to be jacks-of-all-trades that dovetail nicely with the *current individuals from the other population*. What does this mean for practitioners wanting to coevolve "optimal" (or perhaps, even "good") cooperative strategies using a coevolutionary algorithm? It means that CEAs are *not* necessarily optimizers in the sense that one might intuitively expect them to be. Something must be done to modify the existing algorithms or our expectations of what these algorithms really do.

3 Biasing for Optimal Cooperation

One reason CCEAs tend toward "balance" is that an individual's fitness is commonly assessed based on how well it performs with immediate individuals from the other population. To find optimal cooperation, the search process may need to be more optimistic than this: assessing fitness based more on the *highest-reward* interactions between an individual and various members of the other population. A previous investigation in this direction is reported in [Wiegand *et al.*, 2001]:

Agent 1		Agent 2 a	Agent 2 b	Agent 2 c
	a	11	penalty	0
	b	penalty	7	6
	c	0	0	5

Agent 1		Agent 2 a	Agent 2 b	Agent 2 c
	a	10	0	penalty
	b	0	2	0
	c	penalty	0	10

Table 1: Joint reward matrixes for the Climb (left) and Penalty (right) domains.

assessing an individual's fitness based on its *maximum* performance with other agents in a collaborative domain was shown to yield better results than when using the mean or minimum performance. The idea presented in this paper is relatively simple: base an individual's fitness on a combination of its immediate reward while interacting with individuals in the population, and on an estimate for the reward it would have received had it interacted with its *ideal collaborators*. The fraction of reward due to the immediate (as opposed to the ideal) interaction changes during the course of the run.

We note that this notion of bias towards maximum possible reward has also been used in the reinforcement learning literature in subtly different ways than we use it here. For example, maximum reward was used by [Claus and Boutilier, 1998] to modify the exploration strategy of the agent, and by [Lauer and Riedmiller, 2000] to modify the update rule for the Q table. To some extent, the "Hall of Fame" method introduced by [Rosin and Belew, 1997] for competitive coevolution is also related to biased cooperative coevolution.

We justify the use of such a bias in a CCEA as follows. Recall that if an individual's fitness is based on its immediate interaction with individuals from the other population, then $\vec{u} = A\vec{y}$ and $\vec{w} = A^T\vec{x}$, as described in equations 1 and 2. Now, let us consider a function $\max_A$ that returns a column vector corresponding to the maximum value of each row in matrix A. Now, if an individual's fitness is based on its maximum possible performance in conjunction with any individual from the other population, then we may modify equations 1 and 2 to be $\vec{u} = \max_A{}^T$ and $\vec{w} = \max_{A^T}{}^T$.

In this modified system, the tendency to optimize performance is clear. At each iteration of the model, the fitness of each strategy will be its best possible fitness. If there is a unique maximum, that result will have the highest fitness and so the proportion of the corresponding strategy will increase in the next step. When the global maxima are not unique, the resulting fixed point is a mixed strategy, with weights split between those maxima.

The reason for this is straightforward: the problem has lost the dimensionality added due to the nature of the interactions between the agents. Without this, the problem reduces to a simple evolutionary algorithm: regardless of the content of the opposing population, the fitness measure for a given strategy is the same. As shown in [Vose, 1999], an infinite population model of this reduced evolutionary algorithm will converge to a unique global maximum.

But it is difficult to imagine how a real CCEA algorithm would know the maximum possible reward for a given individual *a priori*. One approach is to use historical information during the run to approximate the maximum possible collaborative fitness for an individual. However, if the approximation is too large (or has too strong an effect on the overall fitness), and if it appears too early in the evolutionary run, then it can deform the search space to drive search trajectories into suboptimal parts of the space from which they cannot escape. On the other hand, if the approximation affects the fitness measurement very weakly, and too late in the run, then it may not be of much help, and the system will still gravitate towards "balance".

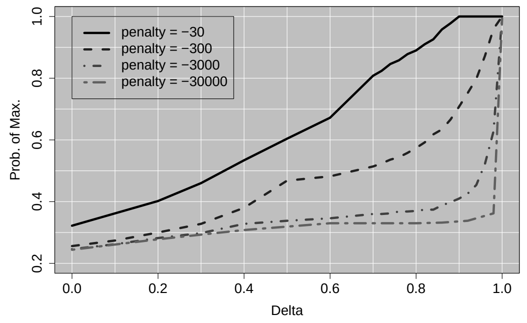

Figure 1: Probability of converging to the optimum as the bias parameter δ is varied between 0 and 1.

To better see this tradeoff, we again alter equations 1 and 2, this time adding a bias weight parameter δ. Now, $\vec{u} = (1-\delta)\cdot A\vec{y} + \delta\cdot\max_A{}^T$ and $\vec{w} = (1-\delta)\cdot A^T\vec{x} + \delta\cdot\max_{A^T}{}^T$. Varying δ between 0 and 1 will control the degree to which the model makes use of the bias. Consider the *Climb* payoff matrix on the left side of Table 1. We select 500 initial points of the dynamical system uniformly at random from $\Delta^n \times \Delta^m$, and iterate the system until it converges. While convergence is virtually guaranteed in traditional two-matrix EGT games [Hofbauer and Sigmund, 1998], it is not necessarily guaranteed in our modified system. In our experimental results, however, we obtained convergence in all cases to within some degree of machine precision. Figure 1 shows the probability, for various levels of δ, of the dynamical system converging to the optimum when the penalty is set to -30, -300, -3000 or -30000. Notice that, as the penalty worsens, the transition between optimal and suboptimal convergence becomes more severe. This suggests that for some problems, any benefits provided by this type of bias may be quite sensitive to the degree of bias.

4 Experiments

While this theoretical discussion helps justify our intuition for including a performance bias in fitness evaluation, it is not immediately applicable to real problems. In a more realistic setting, simplifying model assumptions such as infinite populations, lack of variational operators, complete mixing, and *a priori* knowledge of the maximum payoff are not possible. To convert theory into practice, we have adopted an approximation to the performance bias that is based on historical information gathered during the evolutionary run. We also decreased the bias through the course of a run to take advantage of the fact that initial partners are likely to be weak, while later partners are stronger.

We performed several experiments to compare simple co-

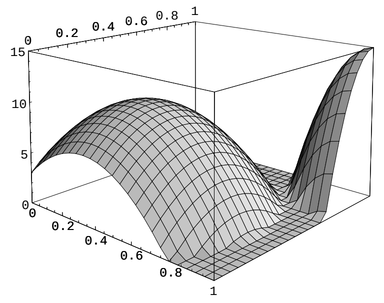

Figure 2: Joint reward in the continuous Two Peaks domain

evolution (SC) with biased coevolution (BC) in three problem domains detailed later. Both SC and BC base fitness on the immediate performance of an individual in the context of individuals from one other cooperating population. BC additionally includes a bias factor: part of the fitness is based on an approximation of what an individual's fitness would be were it to cooperate with its ideal partners.

We compared these two techniques in combination with two approaches to representing an individual. In the Pure Strategy Representation (PSR), an individual represented a single strategy. PSR individuals stored a single integer representing the strategy in question. A PSR individual bred children through mutation: a coin was repeatedly tossed, and the individual's integer was increased or decreased (the direction chosen at random beforehand) until the coin came up heads. In the Mixed Strategy Representation (MSR), an individual represented not a single strategy but a probability distribution over all possible strategies. When evaluating an MSR individual with a partner agent, 50 independent trials were performed, and each time each agent's strategy was chosen at random from the the agent's probability distribution. MSR individuals used one-point crossover, followed by adding random Gaussian noise ($\mu = 0, \sigma = 0.05$) to each of the distribution values, followed by renormalization of the distribution. Observe that using MSR creates a potentially more difficult problem domain than using PSR, for reasons of search space size and stochasticity of the fitness result.

We chose a common approach to cooperative coevolution fitness assessment. An individual is assessed twice to determine fitness: once with a partner chosen at random, then once partnered with the individual in the other population that had received the highest fitness in the previous generation. An individual's fitness is set to the maximum of these two assessments. During a fitness assessment, an individual receives some number of *rewards* for trying certain strategies in the context of partners. For a PSR individual, the assessment was simply the single reward it received for trying its strategy with its partners. As an MSR individual tried fifty strategies, its assessment was the mean of the fifty rewards it received.

SC and BC differ in that BC adds into the reward a bias term, that is, $Reward \leftarrow (1-\delta) \cdot Reward + \delta \cdot MaxReward$, where δ is a decreasing bias rate that starts at 1.0 and linearly decreases until it reaches 0 when 3/4 of the maximal run length has passed. Ideally, the *MaxReward* bias factor would be the highest possible reward received for trying that particular strategy, over all possible partner strategies. In the experiments in this paper, we chose to approximate *MaxReward* by setting it to the maximum reward seen so far in the run for the given strategy.

	Penalty			
	-30	-300	-3000	-30000
SC+MSR	21% 7.58	16% 7.04	16% 4.61	15% -7.69
BC+MSR	100% 10.6	100% 10.0	100% 9.17	97% 1.25
SC+PSR	100% 11.0	100% 11.0	100% 11.0	100% 11.0
BC+PSR	100% 11.0	100% 11.0	100% 11.0	100% 11.0

Table 2: Proportion of runs that converged to global optimum and average best individual fitness, Climbing Domain

	Penalty			
	-10	-100	-1000	-10000
SC+MSR	100% 9.69	66% 6.92	60% 6.24	61% 3.55
BC+MSR	100% 9.71	100% 9.40	100% 8.99	100% 3.36
SC+PSR	100% 10.0	100% 10.0	100% 10.0	100% 10.0
BC+PSR	100% 10.0	100% 10.0	100% 10.0	100% 10.0

Table 3: Proportion of runs that converged to global optimum and average best individual fitness, Penalty Domain

In all experiments, the most fit individual survived automatically from one generation to the next. To select an individual for breeding, we chose two individuals at random with replacement from the population, then selected the fitter of the two. Each experiment was repeated 100 times. The experiments used the ECJ9 software package [Luke, 2002].

4.1 Problem Domains

We experimented with three different single-stage game domains: two simpler ones (Climb and Penalty) introduced in [Claus and Boutilier, 1998], and a more complex artificial problem (Two Peaks). Evolutionary runs in the Climb and Penalty problem domain lasted 200 generations and used 20 individuals per population. Runs in the Two Peaks domain lasted 500 generations and used populations of 100 individuals each.

The joint reward matrices for the Climb and the Penalty domains are presented in Table 1. The domains are difficult because of the penalties associated with miscoordinated actions and the presence of suboptimal collaborations that avoid penalties. Figure 2 presents a continuous version of a the Two Peaks coordination game, where the x and y axes represent the continuous range of actions for the two agents, and the z axis shows the joint reward. The reward surface has two peaks, one lower but spread over a large surface, and the other one higher but covering a small area. Because an agent's strategy space is continuous over $[0,1]$, we discretized it into increasingly difficult sets of 8, 16, 32, 64 or 128 strategies. The discretizations result in slightly different optimal values.

5 Results

Tables 2–4 present the proportion (out of 100 runs) that converged to the global optimum, plus the mean fitness of the best individuals in the runs. MSR individuals were considered optimal if and only if the optimal strategy held over 50

	Discretization Level (Number of Actions)				
	8	16	32	64	128
SC+MSR	0% 8.98	0% 7.48	0% 6.70	0% 6.47	0% 6.36
BC+MSR	0% 8.99	0% 7.51	0% 6.70	0% 6.47	0% 6.36
SC+PSR	100% 11.0	41% 11.6	32% 11.5	42% 12.1	48% 12.4
BC+PSR	100% 11.0	71% 12.8	72% 13.4	61% 13.0	70% 13.0

Table 4: Proportion of runs that converged to global optimum and average best individual fitness, Two Peaks Domain

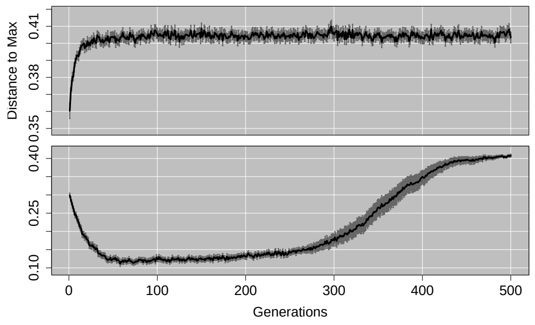

Figure 3: Distance from best-of-generation individuals to optimal strategy for the 8 actions Two Peaks domain using SC (top) and BC (bottom).

percent of the distribution (in fact, most optimal MSR individuals had over 90 percent).

Biased coevolution consistently found the global optima as often as, or more often than, standard coevolution. The only times where standard coevolution held its own was in the Climbing and Penalty domains, where PSR individuals found the optimum 100% of the time, as well as in the harder Two Peaks domain, where no MSR individuals found the optimum. For those problems when individuals found the optimum less than 100% of the time, we also compared differences in mean best fitness of a run, using a two-factor ANOVA with repetitions, factored over the method used and the problem domain.

The ANOVA results allow us to state with 95% confidence that biased coevolution is better than simple coevolution when MSR is used in the Climbing domain, and also in the Two Peaks domain when PSR is used; the tests give only a 90% confidence for stating that BC+MSR is better than SC+MSR in the Penalty domain.

In order to better understand what happens when using MSR in the Two Peaks domains, we plotted the average euclidian distance from the best individual per generation to the known global optima (Figures 3 and 4). The graphs present the 95% confidence interval for the mean of the fitnesses. Investigations showed that SC converged to suboptimal interactions (the lower, wider peak in Figure 2) in all cases. On the other hand, the trajectories of the search process are radically different when using BC. Let's take a closer look as to why this might be so.

As we learned from our discussion surrounding Figure 1,

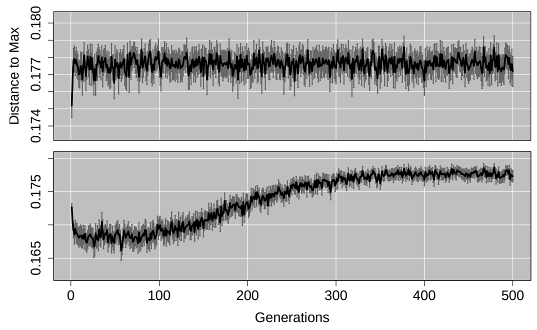

Figure 4: Distance from best-of-generation individuals to optimal strategy for the 32 actions Two Peaks domain using SC (top) and BC (bottom).

successful applications of this biasing method are tied to successfully determining the appropriate degree of bias to apply. Due to MSR's increased difficulty, it may be more challenging to find an appropriate balance for the bias. Figure 3 suggests exactly this. Notice that, in the early part of the run (when δ is strong), the algorithm tends towards the optimal solution; however, as the bias is reduced, it becomes overwhelmed and the trajectories are eventually drawn toward the suboptimal local attractor. Moreover, as the problem becomes larger (i.e., Figure 4, as well as others not shown), this failure occurs earlier in the run. This suggests more careful attention is needed to set the parameters and to adjust the bias rate when using MSR versus PSR. Indeed, by running longer and allowing for more interactions during evaluation, we were able to obtain convergence to the global optimum when using MSR (not shown).

6 Conclusions and Future Work

Although cooperative coevolution has been successfully applied to the task of learning multiagent behaviors, as research about these algorithms advances, it becomes increasingly clear that these algorithms may favor stability over optimality for some problem domains. In this paper, we develop a very simple idea: improve coevolution through the use of a maximum reward bias. We introduce a theoretical justification for the idea, then present experimental evidence that confirms that biasing coevolution can yield significantly better results than standard coevolution when searching for optimal collaborations. Our work further reveals that domain features greatly influence the levels of biasing necessary for convergence to optima: for some problems the performance changes slowly when the level of bias is modified, while for other domains there is a rapid degradation in results. This suggests that, while adding some kind of maximum reward bias can be helpful, there is still work to be done in understanding how best to apply this bias in different problem domains.

Our initial experimental results in this paper suggest that it is effective to use a history as an approximation to the true maximal collaborative reward for a given strategy. For future work we intend to extend these experiments to problem domains with search spaces much larger than the ones used in these experiments. In such domains, the number of strategies may be very large, even infinite. Keeping an effective history of strategies may thus be infeasible in certain circumstances; we intend to explore ways to sample the space or cache the most significant strategy results. Repeated games, such as the Iterated Prisoner's Dilemma, or stochastic games, may also require different approaches to biasing coevolution. Understanding these issues, we hope, can lead to significant improvements in cooperative coevolution's effectiveness as a multi-agent optimization technique.

7 Acknowledgements

This research was partially supported through Department of Army grant DAAB07-01-9-L504, ONR grant N00014-01-1-0703, and a gift from SRA International.

References

[Bassett and De Jong, 2000] Jeffrey K. Bassett and Kenneth A. De Jong. Evolving behaviors for cooperating agents. In Z. Ras, editor, *Proceedings from the Twelfth International Symposium on Methodologies for Intelligent Systems*, pages 157–165, Charlotte, NC., 2000. Springer-Verlag.

[Bull *et al.*, 1995] Lawrence Bull, Terrence C. Fogarty, and M. Snaith. Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA)*, pages 382–388. Morgan Kaufmann, 1995.

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

[Claus and Boutilier, 1998] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Sixteenth National Conference on Aritificial Intelligence (AAAI-98)*. MIT Press, 1998.

[Ficici and Pollack, 2000] S. Ficici and J. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 467–476. Springer-Verlag, 2000.

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

[Iba, 1996] H. Iba. Emergent cooperation for multiple agents using genetic programming. In J. Koza, editor, *Late Breaking Papers of the Genetic Programming Conference*, pages 66–74, Stanford, CA, 1996. Stanford University Bookstore.

[Lauer and Riedmiller, 2000] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference in Machine Learning*, 2000.

[Luke *et al.*, 1998] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup I (Lecture Notes in AI No. 1395)*, pages 398–411, Berlin, 1998. Springer-Verlag.

[Luke, 2002] Sean Luke. ECJ 9: A Java EC research system. http://www.cs.umd.edu/projects/plus/ec/ecj/, 2002.

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

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

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

[Vose, 1999] M. Vose. *The Simple Genetic Algorithm*. MIT Press, 1999.

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

[Wiegand *et al.*, 2002a] R. Paul Wiegand, William Liles, and Kenneth De Jong. Analyzing cooperative coevolution with evolutionary game theory. In D. Fogel, editor, *Proceedings of CEC 2002*, pages 1600–1605. IEEE Press, 2002.

[Wiegand *et al.*, 2002b] R. Paul Wiegand, William Liles, and Kenneth De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In Richardo Poli, Jonathan Rowe, and Kenneth De Jong, editors, *Foundations of Genetic Algorithms (FOGA) VII*, pages 231–248. Morgan Kaufmann, 2002.

[Wu *et al.*, 1999] Annie Wu, Alan C. Schultz, and Arvin Agah. Evolving control for distributed micro air vehicles. In *IEEE International Symposium on Conference on Computational Intelligence in Robotics and Automation (CIRA-99)*, 1999.