A Coevolutionary Approach to Learning Sequential Decision Rules*

Mitchell A. Potter
Dept. of Computer Science
George Mason University
Fairfax, VA 22030
e-mail: mpotter@cs.gmu.edu

Kenneth A. De Jong
Dept. of Computer Science
George Mason University
Fairfax, VA 22030

John J. Grefenstette
NCARAI
Naval Research Laboratory
Washington, DC 20375

Abstract

We present a coevolutionary approach to learning sequential decision rules which appears to have a number of advantages over non-coevolutionary approaches. The coevolutionary approach encourages the formation of stable niches representing simpler subbehaviors. The evolutionary direction of each subbehavior can be controlled independently, providing an alternative to evolving complex behavior using intermediate training steps. Results are presented showing a significant learning rate speedup over a non-coevolutionary approach in a simulated robot domain. In addition, the results suggest the coevolutionary approach may lead to emergent problem decompositions.

1 Introduction

For both natural and artificial organisms the ability to learn complex behavior is desirable, but difficult to achieve. Techniques such as "shaping" are frequently used to construct complex behaviors in stages by breaking them down into simpler behaviors which can be learned more easily, and then using these simpler behaviors as building blocks to achieve more complex behavior (de Garis 1990; Singh 1992; Lin 1993). So, for example, if we were interested in training a robot to track an object in a room full of moving obstacles, we might begin by training it to follow the object in an obstacle free space. Once this simpler task is mastered, we could add fixed obstacles, and later moving obstacles.

In such approaches, the trainer plays a significant role in the decomposition of the tasks into simpler subtasks, in the training of the organisms on the subtasks, and in the synthesis of the desired behavior from the subtasks. In the case of robot learning it is frequently desirable to minimize such heavy dependency on trainers. There was some early optimism that evolutionary techniques could be used to evolve complex behavior directly without any intermediate training, but initial attempts have met with only limited success.

In this paper we present a coevolutionary approach to learning complex behaviors which are realized by sets of sequential decision rules. This approach is intended to require much less human intervention than the shaping approach. In our coevolutionary approach, multiple instances of a genetic algorithm (GA) are run in parallel, each instance of which evolves a species[1] of individuals which are good at particular tasks. These areas of expertise are not enforced explicitly by a trainer; rather, they emerge as a result of an initial seeding of the populations with some very general behavioral rules covering their respective areas of expertise. Once the populations are seeded, the evolution of complex behavior proceeds without a human in the loop and without explicit training on various subtasks.

This is accomplished by selecting a representative from each of the GA populations and combining them into a single composite agent capable of being evaluating on the top level goal task. We call these composite agents "collaborations". Credit from evaluating the composite agent flows back to the individual subcomponents reflecting how well they collaborate with the other subcomponents to achieve the top level goal. This credit is then used by the GA instances to evolve better subcomponents. We call such systems *Cooperative Coevolutionary Genetic Algorithms* (CCGAs).

We initially tested this CCGA architecture in the well-studied and well-understood domain of function optimization and achieved considerable performance improvements over the more traditional centralized architectures (Potter and De Jong 1994). In this paper, we describe a more ambitious on-going effort involv-

*To appear in Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), July 15-19, 1995, Pittsburgh, Pennsylvania USA, Morgan Kaufmann

[1] We define a species to be a collection of interbreeding individuals with similar characteristics

ing the use of CCGAs to evolve sequential decision rules which control the behavior of a simulated robot. Preliminary results are presented which show similar performance improvements in evolving viable rule sets when compared to a more traditional GA approach. These experiments also demonstrate a surprising stability of the niches formed by the species through collaboration. The implications of these results for the evolution of even more complex behavior is discussed.

2 Basic Coevolutionary Framework

The hypothesis underlying the ideas presented here is that, in order to evolve more and more complex behavior, explicit notions of modularity need to be introduced in order to provide reasonable opportunities for complex solutions to evolve in the form of interacting co-adapted subcomponents. The difficulty comes in finding reasonable computational extensions to our current evolutionary paradigms in which such subcomponents "emerge" rather than being hand designed as in (de Garis 1990). At issue here is how to represent such subcomponents and how to apportion credit to them for their contributions to the problem solving activity such that the evolution of complex behavior proceeds without a human in the loop.

Classifier systems attempt to accomplish this via a single population of interacting rules whose individual fitnesses are determined by their interactions with other rules via a simulated micro-economy (Holland and Reitman 1978). Other extensions to current evolutionary paradigms have been proposed to encourage the emergence of niches and species in a single population (DeJong 1975; Deb and Goldberg 1989; Davidor 1991; Forrest, Javornik, Smith, and Perelson 1993; Giordana, Saitta, and Zini 1994; Moriarty and Miikkulainen 1996) in which individual niches compete for the allocation of trials.

The use of multiple interacting subpopulations has also been explored as an alternate mechanism for coevolving niches using the so-called *island model* (Grosso 1985; Cohoon, Hegde, Martin, and Richards 1987; Pettey, Leuze, and Grefenstette 1987; Tanese 1989; Whitley and Starkweather 1990). In the island model a fixed number of subpopulations evolve competing rather than cooperating solutions. In addition, individuals occasionally migrate from one subpopulation (island) to another, so there is a mixing of genetic material. The previous work that has looked at cooperating rather than competing subpopulations has involved a user-specified decomposition of the problem into species (see, for example, (Husbands and Mill 1991) or (Hills 1990)).

Cooperative coevolutionary algorithms (CCAs) combine and extend ideas from these earlier systems in several ways. As shown in Figure 1, a CCA consists of a collection of evolutionary algorithm (EA) instances, each attempting to evolve species which are useful as modules for achieving higher level goals. For example, in the robotics domain one species may know how to track an object, another may know how to avoid an obstacle, and those skills can be used in combination to solve difficult surveillance tasks. Although nothing explicitly prevents multiple species from existing within the same subpopulation, in practice, a single species will tend to dominate each subpopulation as shown in Section 5.

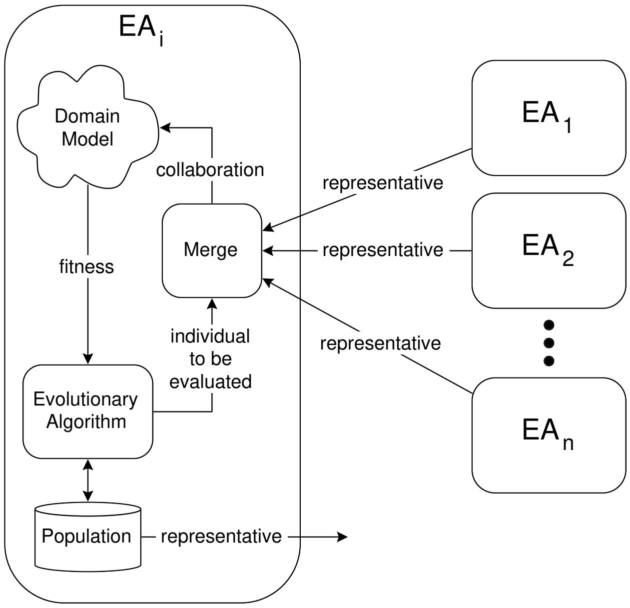

Figure 1: Cooperative coevolutionary architecture from the perspective of the *i*th EA instance

Complete collaborative solutions are obtained by assembling representative members from each of the subpopulations. Credit assignment within each EA instance is defined in terms of the fitness of the collaborations in which its subpopulation members participate. This provides evolutionary pressure for the species from separate subpopulations to cooperate rather than compete. However, competition still exists among individuals within the same subpopulation.

Unlike the island model, the individuals from the separate subpopulations do not migrate; therefore, interbreeding does not occur. This lack of migration eliminates haphazard and often destructive recombination between dominate species once niches are established.

Mechanisms exist for seeding initial subpopulations with a bias towards evolving certain kinds of species. However, once evolution begins there is nothing to prevent the roles of species from changing considerably. In addition, the general coevolutionary model places no restrictions on the number of subpopulations. In some domains it may be desirable to allow the dynamic creation of new subpopulations and the elimination of unproductive ones (i.e., the birth and death of species).

In the CCA instantiation described in this paper, the evolution of each subpopulation is handled by a standard GA (although we could just as easily have chosen some other evolutionary algorithm). We emphasize this choice by adopting the more specific terminology CCGA.

3 System Architecture

In the previous section we described cooperative coevolutionary algorithms in general terms. In this section we illustrate this in more detail by showing how CCGAs can be used to evolve sequential decision rules. The approach we have taken is to extend an existing genetic algorithm based system called SAMUEL with the necessary components required to support coevolution. SAMUEL, which stands for Strategy Acquisition Method Using Empirical Learning, is designed to evolve sets of sequential decision rules to be used by decision making agents (Grefenstette, Ramsey, and Schultz 1990).

3.1 Standard SAMUEL System

The architecture of SAMUEL consists of three main components: a domain module, a production system, and a genetic algorithm. The production system and domain module evaluate rule sets while the genetic algorithm evolves new rule sets.

The domain module consists of a world model, agents, and a critic. Each agent consists of a set of sensors and effectors and is controlled by a set of rules being processed by the production system. Multiple agents can be controlled by the same rule set or by different rule sets. The critic monitors the agents and evaluates their performance within the context of the world model.

The production system performs matching, conflict resolution, and credit assignment on the rules in each rule set. Pattern matching is done between rule antecedents and an agent's sensor values. When a rule fires it modifies an agent's effectors. Conflict resolution is a function of rule strength. The strength of the rules is adjusted by the credit assignment algorithm based on feedback from the critic.

The genetic algorithm evolves a population of rule sets, that is, each chromosome represents an entire rulebase for controlling an agent. The size of these chromosomes vary based on the number and complexity of the rules they represent. This can be contrasted with a *classifier system* which evolves a population of rules rather than a population of rule sets. To evaluate one of its rule sets, the GA sends it to the production system where it is used to control an agent for a number of *episodes*. An episode is defined to be the application of agents to a domain task. The complete set of episodes used to evaluate a rule set is called a *trial*. The GA receives feedback from the critic on the average performance of the agent over the entire set of episodes and uses this feedback as a fitness measure. It also receives a set of updated rule strengths from the production system at the end of the trial. The GA reproductive cycle consists of replicating rule sets based on fitness and applying crossover and several types of rule modification operators to them.

3.2 Coevolutionary SAMUEL System

We have extended SAMUEL to allow multiple instantiations (modules) to be run in parallel and communicate with each other for the purpose of forming collaborations as described in Section 2. The architecture of a SAMUEL module is shown in Figure 2. Each SAMUEL module evolves its own subpopulation of rule sets that, in practice, is dominated by a species with expertise in a particular area. For example, one species may consist of competing rule sets for moving the robot toward an energy source while another species may concentrate on rule sets for conserving energy when no energy source is present. Each module makes its best rule set available to the other modules so that collaborations may be formed. This allows rules covering the entire spectrum of required behaviors to be coevolved across multiple populations rather than "shaping" behaviors in stages or evolving all behaviors in a single chromosome.

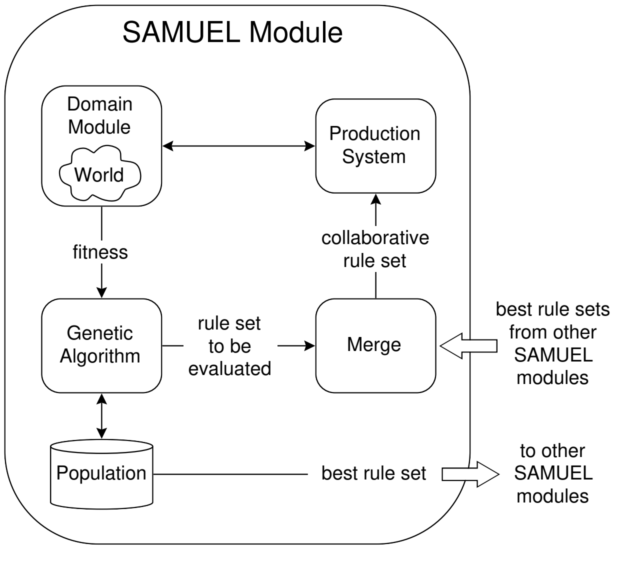

Figure 2: Coevolutionary SAMUEL module

More specifically, in order for a SAMUEL instantiation to send a rule set to the production system to be evaluated, the rule set must first be merged with the best rule set from each of the other SAMUEL instantiations. This merging process consists simply of concatenating the rule sets. Fitness in the local GA reflects how well the local rule set performs with the current best rule sets from the other instantiations to achieve the top level goal. In other words, the behaviors being evolved by a SAMUEL instantiation are evaluated within the context of the current best behaviors from each of the other instantiations.

SAMUEL also provides a simple mechanism for initially seeding each population with user-supplied rules. We use that facility to provide each of the populations with an initial bias to develop an area of expertise different from the others. This will be discussed in detail in the next section. Although ultimately we plan to include the birth of new species (subbehaviors) and the death of old, unproductive ones, in the current version of the system the number of populations is fixed.

4 Robot Domain

We have evaluated the CCGA approach to evolving complex behaviors in a simulated robot domain designed to provide a wide range of learning tasks. We are particularly interested in challenging classes of problems involving multiple robots in competitive or cooperative tasks, such as locating, tracking, or evading other agents in a cluttered environment, locating and retrieving hidden items, either in competition or in cooperation with other agent, and performing material handling tasks while avoiding collisions with other agents performing similar tasks. All of these tasks involve multiple levels of skills, and all appear to require some form of decomposition in order to be learned with current techniques.

In this paper we report initial results for a moderately complex task in which a robot must maneuver itself around an obstacle free room in which food pellets appear at random locations and times. The robot must consume these food pellets to replace lost energy. Energy loss is a function of the speed and turning rate of the robot. In addition, there is a small energy drain on the robot even when it remains stationary. Therefore, a lazy robot that does not seek food pellets will eventually run out of energy and die. To make the task more difficult, there is a second robot in the environment controlled by a fixed set of hand-crafted rules and which competes for the food pellets. We refer to this second robot as the "enemy robot" and the robot using the evolving rules as the "friendly robot".

Each robot has ten sensors and two effectors. The sensors consist of a food pellet detector, a clock indicating the amount of time that has passed since the appearance of the last food pellet, the robot's current energy level, heading, and speed, the range and bearing to the center of the room, the range and bearing to the other robot, and the range and bearing to the food pellet if one currently is present. The effectors consist of speed and turning rate.

The goal is to evolve a set of behaviors (rule sets) which enable the friendly robot to survive indefinitely in this stochastic, unfriendly environment. To evaluate one of these evolving rule sets, a number of food-gathering episodes are simulated in which the energy level of the robot using the rule set is monitored. The average energy level over the set of episodes is returned as the fitness of the rule set. Due to the random appearances of the food pellets, many of these training episodes must be run to accurately evaluate a rule set. For example, if by chance the food appears in close proximity to the enemy robot over a number of episodes the friendly robot may die even when controlled by a very good rule set.

A snapshot from one of these simulations is shown in Figure 3. The large circles with heading indicators represent the two robots. The small unicolor circles represent previously consumed foot pellets. Each of these pellets is color coded to indicate which robot consumed it. The small white circle with a black dot represents the currently active food pellet.

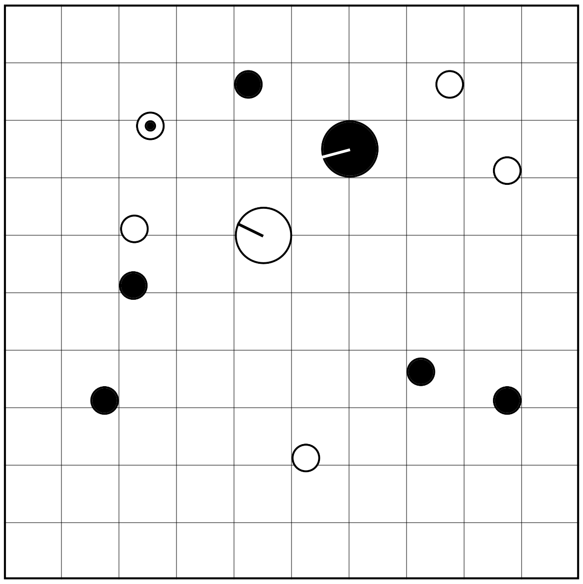

Figure 3: Snapshot from food gathering simulation

The hand-crafted rules used by the enemy robot cause it to move at a random speed on a slightly wandering path toward food pellets when they appear and to move to the center of the room when no food is present. The algorithm for positioning the food pellets places them at a fixed range (plus or minus a few grid locations) and random bearing from the center of the room; therefore, moving to the center of the room to wait for the appearance of the next pellet is a reasonably good behavior for a robot to have. The rules for controlling the turning rate of the enemy robot are also reasonably good. These rules provide for a challenging competitor, but they are not optimal behaviors. For example, it may be possible to do better by moving more directly to the food when present and not moving at all when the possibility of getting there first is small. Therefore, the GA has a chance to evolve behaviors that are superior to the enemy robot.

5 Experimental Results

We have evaluated the CCGA approach to learning sequential decision rules for controlling a simulated robot by running it on the task domain described in the previous section and comparing its performance with the performance of the standard non-coevolutionary SAMUEL system. We were particularly interested in studying two aspects of the CCGA approach: whether or not it would speed up rule learning, and whether or not it could evolve and maintain a set of cooperating behaviors. It could happen, for example, that subpopulations end up containing complete, monolithic solutions rather than subcomponents, and thus there is no advantage over SAMUEL (in fact, a considerable performance disadvantage in that parallel versions of SAMUEL were run with the same net result).

To explore these issues an experiment was performed consisting of ten runs of 7500 trials for both the coevolutionary and non-coevolutionary systems. Each trial consisted of 400 training episodes. The population size of both the coevolutionary and non-coevolutionary systems was set to fifty rule sets.

In order to encourage exploration by the GA, the initial population of rule sets being evolved by the standard SAMUEL system for use by the friendly robot was seeded with rules more general than the hand-crafted enemy robot rules. Specifically, they implemented behavior similar to that of the enemy robot when food was present; however, when no food was present they caused the friendly robot to wander randomly rather than returning to the center of the room.

The coevolutionary system consisted of two SAMUEL instances. The instances were seeded with rules intended to give them an initial biases corresponding to a fairly natural decomposition of the task into behaviors appropriate for when food was present and when it was absent. This was accomplished by dividing the set of rules used to initialize the standard SAMUEL system described in the previous paragraph into two subsets. One subset consisted only of rules matching situations in which food was present and the other subset consisted only of rules matching situations in which food was absent. One SAMUEL instance was then initialized using the "food present" subset and the other initialized with the "food absent" subset. We call these two instances Samuel_1 and Samuel_2 respectively. This method was intended to encourage, but not require, specialization along these behavioral lines. It should be noted that SAMUEL includes mutation operators that are capable of altering a rule from one that matches situations in which food is present to one that matches situations when food is absent, and vice versa.

Figure 4 illustrates the results obtained with respect to the speed of learning. The plot was generated from the average of ten runs. In this figure the best fitness evolved so far is plotted as a function of the number of trials. The unit of trials was used rather than selection cycles (generations) to provide a fairer comparison between the two systems. Note that a single generation of coevolutionary SAMUEL consists of twice the number of trials as in a generation of standard SAMUEL since two separate populations (twice the number of individuals) must be evaluated; therefore, the coevolutionary system was only run for half the generations as the standard system.

The first observation we make from this figure is that the coevolutionary system is able to evolve viable rule sets in significantly fewer training episodes than the standard SAMUEL system. Specifically, the coevolutionary system has achieved a reasonably good fitness of 70 in an average of about 800 trials while the non-coevolutionary system takes an average of about 1400 trials to achieve the same level of fitness. The second observation is that given a sufficiently large number of trials, both the coevolutionary and standard SAMUEL systems can evolve reasonably good rule sets for this particular task. This is illustrated by the eventual convergence of the two learning curves.

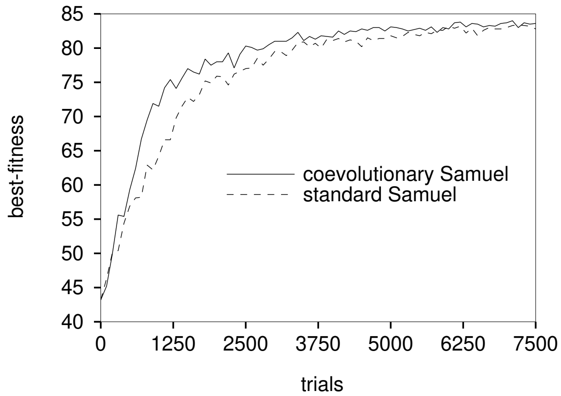

Figure 4: Performance comparison of best fitness averaged over ten runs

To determine the significance of the learning curve results the number of trials required to achieve a fitness of at least 70 was determined for each of the ten runs. This data is summarized in Table 1. The sample distributions from both the coevolutionary and standard systems were reasonably normal. In addition, the statistical significance of the results was computed using a one-sided t-test. This resulted in a statistically significant p-value of 0.034.

Table 1: Comparison of trials required to achieve a fitness of 70 over ten runs

Algorithm	*Mean*	*Min*	*Max*	*Stdev*
coevolution	830	300	1700	392
standard	1440	350	3550	909

To address the second issue, the ability of the coevolutionary system to evolve and maintain a set of cooperating behaviors, the best rule set from the final generation of each SAMUEL instantiation was inspected for the first five runs. The results are shown in Table 2. The table indicates the percentage of rules that will match situations in which food is present, food is absent, or both of these conditions (general behavior).

Table 2: Summary of behavior classes in best rule set from final CCGA generation averaged over five runs

Instance	*Food Present Behavior*	*Food Absent Behavior*	*General Behavior*
Samuel_1	92.3%	1.4%	6.2%
Samuel_2	4.5%	75.8%	19.7%

The first observation we make from the table is that the niches formed by the species through collaboration were relatively stable as shown by the high percentage of rules exhibiting behavior consistent with the initial seeding of the populations. This is significant given that the average Samuel_1 rule set size grew to eighty-three rules from an initial size of eight and the average Samuel_2 rule set grew to ninety-seven rules from an initial size of two. Because each individual is evaluated based on how well it collaborates with representatives from the other SAMUEL instances, there is evidently little incentive for a SAMUEL instance to explore a region of the search space already being exploited by one of the species from another SAMUEL instance.

The second observation (not obvious from the table) is that a third species emerged and co-habitated the population dominated by the species specializing in behavior related to food being absent. The rules of this third species matched situations in which the energy level of the robot had exceeded some value. Their consequent was generally to stop the robot from moving to conserve energy. In other words, if the robot is not hungry then it should not waste energy seeking food. Many of the 19.7% general behavior rules found in the best rule set from the final generation of the Samuel_2 instance were of this category. The appearance of this third species suggests a bifurcation of the Samuel_2 subpopulation may have been appropriate. Methods for splitting subpopulations will be explored in future studies.

6 Discussion and Conclusions

Evolving complex behavior directly without intermediate training steps is a difficult task. We believe that the CCGA architecture described here provides a promising approach for achieving this. We have provided here preliminary evidence that the CCGA approach can provide significant speedup over the non-coevolutionary approach in this complex robot domain.

Perhaps more important than speedup is the evidence that CCGAs encourage the formation of stable niches representing cooperating subcomponents (species). Evidently once a species establishes a niche, there is little evolutionary pressure for the other species to exploit the same niche. This is important for several reasons. First, it provides a way of controlling the evolutionary direction of each species. This can be accomplished by seeding the initial populations with some general rules in the areas of interest. By controlling the evolutionary direction of a number of collaborative species we should be able to coevolve more complex behavior than possible with a monolithic approach, although our current domain is not sufficiently complex to show this explicitly. Second, it suggests the possibility of an emergent decomposition of a problem. Given a number of coevolving species, when one of them discovers a desirable region of the search space there seems to be no evolutionary pressure for the other species to exploit the same region. Instead, they continue to search for complementary regions, thus leading to an emergent decomposition.

There are a number of logical extensions to this work. First, we intend to experiment with domains sufficiently complex to prevent the non-coevolutionary approach from successfully evolving a solution without intermediate training steps and show that cooperative coevolution is able to evolve reasonable behavior in these complex domains. Second, we intend to explore the use of coevolution for emergent decomposition of problems via the birth of new species and the death of old, unproductive ones.

Acknowledgements

This work was supported by the Navy Center for Applied Research in Artificial Intelligence. In addition, we wish to acknowledge the contribution of Bob Daley, who in a series of discussions suggested ways in which the SAMUEL system could be modified to support co-

evolution.

References

Cohoon, J., S. Hegde, W. Martin, and D. Richards (1987). Punctuated equilibria: A parallel genetic algorithm. In J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 148–154. Lawrence Erlbaum Associates.

Davidor, Y. (1991). A naturally occuring niche & species phenomenon: The model and first results. In R. Belew and L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 257–263. Morgan Kaufmann.

de Garis, H. (1990). Building artificial nervous systems using genetically programmed neural network modules. In B. Porter and R. Mooney (Eds.), *Proceedings of the Seventh International Conference on Machine Learning*, pp. 132–139.

Deb, K. and D. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. In J. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 42–50. Morgan Kaufmann.

DeJong, K. (1975). *Analysis of Behavior of a Class of Genetic Adaptive Systems*. Ph. D. thesis, University of Michigan, Ann Arbor, MI.

Forrest, S., B. Javornik, R. Smith, and A. Perelson (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation 1*(3), 191–211.

Giordana, A., L. Saitta, and F. Zini (1994). Learning disjunctive concepts by means of genetic algorithms. In W. Cohen and H. Hirsh (Eds.), *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 96–104. Morgan Kaufmann.

Grefenstette, J., C. Ramsey, and A. Schultz (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning 5*(4), 355–381.

Grosso, P. (1985). *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. Ph. D. thesis, University of Michigan, Ann Arbor, MI.

Hills, D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II*, pp. 313–324. Addison-Wesley.

Holland, J. and J. Reitman (1978). Cognitive systems based on adaptive algorithms. In D. Waterman and F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems*. Academic Press.

Husbands, P. and F. Mill (1991). Simulated coevolution as the mechanism for emergent planning and scheduling. In R. Belew and L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 264–270. Morgan Kaufmann.

Lin, L.-J. (1993). Hierarchical learning of robot skills by reinforcement. In *Proceedings of the 1993 International Joint Conference on Neural Networks*, pp. 181–186. IEEE Press.

Moriarty, D. E. and R. Miikkulainen (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning 22*(1). In Press.

Pettey, C., M. Leuze, and J. Grefenstette (1987). A parallel genetic algorithm. In J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 155–161. Lawrence Erlbaum Associates.

Potter, M. and K. De Jong (1994). A cooperative coevolutionary approach to function optimization. In Y. Davidor and S. H.-P. (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pp. 249–257. Springer-Verlag.

Singh, S. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning 8*, 323–339.

Tanese, R. (1989). Distributed genetic algorithms. In J. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 434–439. Morgan Kaufmann.

Whitley, D. and T. Starkweather (1990). Genitor II: a distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence 2*, 189–214.