CHAPTER 7

COEVOLUTIONARY LEARNING WITH SPATIALLY DISTRIBUTED POPULATIONS

Melanie Mitchell

I. INTRODUCTION

Many approaches to machine learning require a system to learn a model from a set of training examples that the model must classify, or training problems that the model must solve. How are these training cases to be chosen? Typically, due to the costs of acquisition or computation, only a relatively small sample from the universe of possible problems is available to the learner. How should that sample be chosen?

Clearly, if the problems are too easy or to difficult, the system will not make progress in learning. Ideally, the problems should be chosen to be optimally challenging for a system's current state of learning and to specifically target weaknesses in the system at a given time. In life, good teachers craft such problems for their students. The challenge in machine learning is to craft such problems automatically.

Approaches to this issue in the machine learning community include active learning (Cohen et al., 1996), boosting (Freund and Schapire, 1997), and coevolutionary learning. This last approach has been surprisingly successful on a number of tasks, but lacks the theoretical underpinning of the first two. It is not well understood why coevolution works, when it will succeed, and how best to apply it. The purpose of this chapter is to briefly review some of the results to date in coevolutionary learning, and to report on recent research that explores the role of spatially extended populations in the success of coevolution.

Coevolutionary learning, which builds on genetic algorithms (GAs) and other evolutionary computation techniques, has been explored by many people. The first explicit application of computational "host-parasite coevolution" was performed by Hillis (1990). Hillis's inspiration came from host-parasite coevolution in nature. One striking natural example is the phenomenon of "egg mimicry" in plants. Insects such as butterflies sometimes lay their eggs on plant leaves, providing a ready food source for newly hatched larvae. The passion flower plant has evolved a protection against such parasitism by producing toxic chemicals in its leaves. However, the genus of butterfly called *Heliconius* has evolved a counter-adaptation: its larvae are able to tolerate these chemicals. In response, the passion flower has evolved a remarkable counter-counter-adaptation: yellow spots on its leaves that resemble *Heliconius* eggs. In order to avoid too much food competition among larvae, butterflies will not lay eggs on leaves already crowded with eggs, and the spots on passion flower leaves fool at least some butterflies into thinking that there are already too many eggs on the leaves. Moreover, it turns out that the yellow spots are actually glands that produce nectar, thereby attracting ants and wasps, which also prey on the eggs and larvae (Turner, 1981).

Such "evolutionary arms races" among coevolving organisms abound in nature, and were recognized by Darwin as a major force driving evolutionary change.

Hillis used these ideas in his computer evolution of optimal parallel sorting networks. The evolving networks were the hosts, whose fitnesses were determined by their performance on training problems (lists of items to be sorted). The lists themselves — the parasites — evolved to be challenging for the evolving hosts. Hillis found significant improvement in the resulting sorting networks as compared with those evolved via a more standard genetic algorithm.

In general, coevolutionary learning systems consist of two populations of individuals which evolve concurrently, with the fitness of individuals in each population depending on their interactions with individuals in the other population. The interactions can be competitive, as in Hillis' host-parasite coevolution or the coevolution of game-playing strategies (e.g., Barricelli, 1962a, b; Reed et al., 1967; Rosin and Belew, 1997; Fogel, 2002), or cooperative, as in the cooperative coevolution of neural network weights (e.g., Potter and De Jong, 2000). This chapter will discuss competitive coevolution only.

II. PROBLEM DOMAINS SUITED FOR COEVOLUTIONARY LEARNING

For competitive coevolutionary learning to be applied to a given problem domain, both the candidate solutions and the environment used to train them (e.g., training examples) have to be evolvable. Game playing is one obvious domain in which this is the case: The host and parasite populations both consist of game-playing strategies. The fitness of an evolving strategy is its success in playing against a sample of other evolving strategies. Axelrod's (1987) and Lindgren's (1992) work on coevolving strategies for the prisoner's dilemma game followed this model, though neither approach used a separate host and parasite population. Rosin and Belew used host-parasite coevolution to evolve strategies for the games of nim and three-dimensional tic-tac-toe (1997). Pollack et al. (1996) used a simple coevolutionary hill-climbing strategy to evolve very successful strategies for backgammon.

Another potential problem domain is the coevolution of special-purpose algorithms and test cases. Hillis'work on coevolving sorting networks is an example of this. His coevolutionary learning approach was able to discover a correct 16-input sorting network with 61 comparisons, which is close to the presumed minimal number of 60 comparisons in a network designed by Green in 1969.

Coevolution has been applied to the design of desired behaviors for robots and autonomous vehicles, both simulated and physical (e.g., Sims, 1994; Ronge and Nordahl, 1996; Nolfi and Floreano, 1998; and Haith et al., 1999). It has also been applied to drug design: Rosin (1997) reported on a preliminary study in which he and colleagues coevolved simulated HIV protease inhibitors (hosts) with simulated protease enzymes (parasites). The idea was to evolve inhibitors that were effective against all possible protease mutants.

A final example of a potential application is in computer security, in which, say, intrusion detection or virus detection algorithms could be co-evolved with possible threats. While there has been substantial work on "computer immune systems" (e.g., Hofmeyer and Forrest, 2000), I am not aware of any work on this topic that uses coevolutionary learning.

The examples described above are only a sample of the previous and potential application areas for coevolutionary methods.

III. HYPOTHESIZED ADVANTAGES FOR COEVOLUTION

In spite of the volume of work on coevolutionary learning in recent years, there is still little understanding of why and under what conditions this approach will be useful. Why should we expect coevolutionary learning to have any advantages over more traditional evolutionary computation approaches? Several reasons have been suggested in the literature, similar to those suggested for biological host-parasite systems (Dawkins and Krebs, 1979).

One hypothesis is that coevolution will be better able to preserve diversity in its populations than non-coevolutionary methods, since the environment for each population (i.e., the other population) is continually changing. It has also been hypothesized that since parasites continually evolve to exploit the hosts' weaknesses, coevolutionary arms races between the host and parasite populations will produce better performing, more general hosts. To be optimally challenging for the hosts, the hosts will learn successfully with many fewer training examples than needed in traditional machine learning approaches.

These hypotheses are compelling and plausible, but have not yet been rigorously and generally tested in experiments on coevolutionary learning. Sections VI and VII will discuss some methods for testing these hypotheses and the results of these tests on two particular sets of coevolution experiments.

IV. POSSIBLE IMPEDIMENTS FOR COEVOLUTION

The success of an evolutionary algorithm is typically measured in one or more of three ways: the quality of the solution found, the probability of finding a high-quality solution (i.e., fraction of successful independent runs), and the computational effort required to find a high-quality solution.

Using these criteria, coevolution has met with mixed success on a number of problems. Hillis (1990) showed that coevolution found a better (smaller) correct sorting network than did evolution alone, but this seemed to require very large populations, as well as an initial host population that already possessed some features of optimal sorting networks. Rosin and Belew (1997) demonstrated that coevolution was able to produce optimal strategies for playing nim and three-dimensional tic-tac-toe, whereas evolution alone was not able to do so. However, this success was achieved only when additional heuristics were used in conjunction with coevolution, including competitive fitness sharing, shared sampling, and a "hall of fame" which saved the most successful parasite from each previous generation. Nolfi and Floreano (1998) were able to foster an arms race in the coevolution of simple robot controllers, but also needed to use the hall-of-fame heuristic for success. Paredis (1997) coevolved cellular automaton (CA) rules and initial configurations in order to discover a rule that performed well on the density classification problem (Das et al., 1994), but found coevolution to be unsuccessful at discovering a rule with good classification performance. Juillé and Pollack (1998) combined coevolution with resource sharing — a diversity preserving heuristic for GAs — on the CA density classification problem. Their algorithm was successful in finding a high-performance CA for this task. (They did not report the probability of finding high-performance CAs or the amount of computational effort required to do so.) However, Werfel et al. (2000) demonstrated that the success of Juillé and Pollack's algorithm was due largely to resource sharing rather than coevolution. De Jong and Pollack (2002) used notions of multi-objective optimization and "Pareto-coevolution" that improve the results of coevolution in some cases, but require large numbers of examples and interactions between all hosts and all parasites. There are several other examples in the literature.

The general conclusion from these examples is that coevolution often needs special heuristics or starting conditions in order to overcome several impediments associated with coevolution alone. These impediments have been pointed out by various researchers (Cartlidge and Bullock, 2004; De Jong and Pollack, 2004; Nolfi and Floreano, 1998; Paredis, 1997; Shapiro, 1998), sometimes using different terminology. The following is a partial list:

> *Loss of gradients*: Coevolution leads to a state in which the parasite population becomes too easy (called "disengagement") or too difficult (called "over-virulence") for the host population. The result is that every individual in the host population either defeats all parasites it samples (though not all possible parasites) or is defeated by all parasites it samples. In either case, all hosts are assigned the same fitness, as are all parasites, and there is no longer any gradient (i.e., difference in fitnesses) that can be used for selection.
>
> *Over-specialization*: The population of hosts gets stuck in a local optimum in which hosts are able to defeat a subset of the parasites, but are not general solutions to the problem at hand.
>
> *Red queen dynamics* (or *cycling* or *mediocre stable states*): The populations of hosts and parasites continue to change in response to one another but these changes do not force hosts to become more general solutions.

Can these impediments be overcome without adding computationally expensive, ad hoc heuristics to coevolution? A number of groups have investigated the hypothesis that spatial distribution of the host and parasite populations, with local interactions governing both fitness evaluations and selection, is a nature-inspired method for alleviating these problems in coevolutionary learning. As a short-hand, I will use the term "spatial coevolution" to mean coevolution with spatially distributed populations, and "non-spatial coevolution" to mean more traditional approaches to coevolution in which host and parasite populations either exhaustively test one another or use random sampling to effect interactions.

Several experiments have given evidence that spatial coevolution can alleviate the impediments described above (e.g., Hillis, 1990; Mitchell et al., 2006; Pagie and Hogeweg, 1997; Pagie and Mitchell, 2002; Ronge and Nordahl, 1996; Williams and Mitchell, 2005; Wiegand and Sarma, 2004). However, there is still no good understanding why spatial distribution significantly improves the performance of coevolution on numerous problems.

The next sections review the work of my own group on spatial coevolution and our investigations into its substantial success relative to other evolutionary methods. Our work attempts to isolate the factors that give spatially extended coevolution an advantage. Is spatial distribution or coevolution alone the main factor contributing to success, or must there be a combination of the two? Is resource sharing alone able to achieve the same success as coevolution? We try to answer these questions by comparing the results of spatial coevolution with that of five other evolutionary methods in which we eliminate coevolution or spatial extent, or both. Parts of this work have been reported in shorter form by Pagie and Mitchell, (2002), Williams and Mitchell (2005), and Mitchell et al. (2006).

V. PROBLEM DOMAINS USED IN OUR STUDY

Building on previous work, we have used two problem domains for studying coevolutionary learning: function induction via genetic programming and evolving cellular automata to perform computations.

Function Induction

The function induction task we use is the one studied by Pagie and Hogeweg (1997). The target function is:

$$f(x, y) = \frac{1}{1+x^{-4}} + \frac{1}{1+y^{-4}}$$

The population of candidate solutions (hosts) consists of genetic-programming-style trees created from the function set {+, –, *, %} and terminal set {x, y, $\Re$} where $\Re$ returns a random constant in [-1.0, 1.0] every time a node containing it is generated. The +, –, and * operators are the standard arithmetic functions for addition, subtraction, and multiplication, respectively; each takes two arguments. The protected-division function % takes two arguments, A and B, and returns 1 if $B = 0$ and A/B otherwise. The terminals x and y evaluate to the respective values of the coordinates on which the tree is being evaluated.

An example of a candidate solution is the tree shown in Fig. 1. Suppose this tree is given $x = 0.8$, $y = 0.4$ as input. When the tree is evaluated, the $\Re$ operator will generate a random constant, which will be passed on to offspring of this tree. Suppose $\Re$ generates 0.1. Then the tree will return (+ (/ 0.4 0.8) (* 0.4 0.1)) = 0.54. Given these same values for x and y, the target function would return $1/(1 + 0.8^{-4}) + 1/(1+0.4^{-4}) = 0.316$. The *error* of a tree on a problem $p = (x,y)$ is the absolute value of the difference between the target function's value and the host's value on p: $|f(x,y) - h(x,y)|$. In this example, the error is $|0.316 - 0.54| = 0.224$.

Training problems (parasites) are (x,y) values evenly distributed over $x \in [-5.0, 5.0]$, $y \in [-5.0, 5.0]$ at regular intervals of 0.4. The total number of examples (x,y) in this domain is 676. The goal is to discover a tree h representing a good approximation of the target function such that the error over all 676 training problems (x,y) is close to zero. That is, a successful tree will return values $h(x,y)$ that are close to the target $f(x,y)$.

A host h is defined to be *correct* on a parasite p if its error on p is less than or equal to 0.01. The fitness of a host h over a sample of n parasites is the inverse of its average error over those n parasites. The lower the average error, the higher the fitness. For coevolutionary methods, the fitness of a parasite p with respect to a single host h is equal to the error of h on p.

A *successful* run of evolution on this problem is defined to be a run in which at least one host is discovered that is correct over the complete set of 676 problems, and at least one such host remains in the population for 50 successive generations. Unsuccessful runs terminate after some maximum number of generations.

Evolving Cellular Automata

Several groups have explored genetic algorithms as a means to design cellular automata with collective computational abilities (e.g., Das et al., 1994, 1995). Most of these studies use one-dimensional, binary-state cellular automata. Such a CA is a one-dimensional lattice of N two-state machines ("cells"), each of which changes its state as a function only of the current states in a local neighborhood. As is illustrated in Fig. 2, the lattice starts out with an initial configuration (IC) of cell states (0s and 1s) and this configura-

Figure 1. An example of a function tree.

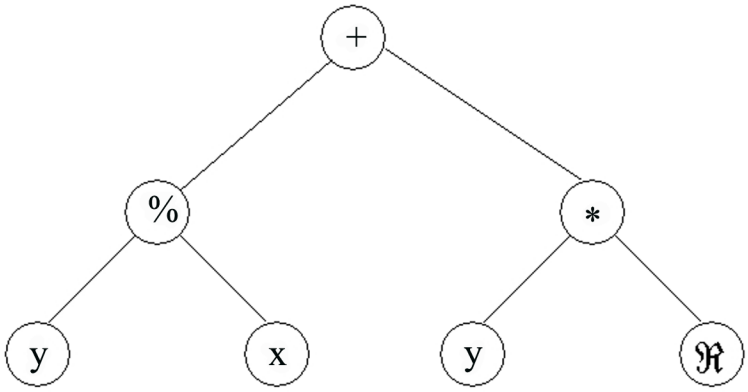


Figure 2. Illustration of a one-dimensional, binary-state, nearest-neighbor ($r = 1$) cellular automaton with $N = 11$. Both the lattice and the rule table ϕ for updating the lattice are illustrated. The lattice configuration is shown at two successive time steps.

Rule table ϕ:

Neighborhood η:	000	001	010	011	100	101	110	111
Output bit:	0	0	0	1	0	1	1	1

Lattice:

r = 1

Neighborhood $\eta \rightarrow$

t = 0	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
t = 1	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**

tion changes in discrete time steps. At each time step, all cells are updated simultaneously according to the CA *rule* ϕ. (Here we use the term "state" to refer to the value of a single cell. The term "configuration" will refer to the collection of local states over the entire lattice.)

A CA's rule ϕ can be expressed as a lookup table (or "rule table") that lists, for each local neighborhood, the state that is taken on by the neighborhood's central cell at the next time step. For a binary-state CA, these update states are referred to as the "output bits" of the rule table. In a one-dimensional CA, a neighborhood consists of a cell and its r ("radius") neighbors on either side. (In Fig. 2, $r = 1$.) Here we describe CAs with periodic boundary conditions—the lattice is viewed as a circle.

Das et al. (1994) used a GA to evolve rule tables for one-dimensional, binary-state, $r = 3$ CA (with periodic boundary conditions) that would perform the "density classification" task. The goal was to find a CA that decides whether or not the IC contains a majority of 1s (i.e., has high density). If so, then within $2N$ time steps (where N is the number of cells in the lattice), the CA should go to the fixed-point configuration of all 1s (i.e., all cells in state 1 for all subsequent iterations); otherwise, within $2N$ time steps it should produce the fixed-point configuration of all 0s. A CA is *correct* on an IC if the CA starting with that IC produces the correct final fixed point configuration within $2N$ time steps.

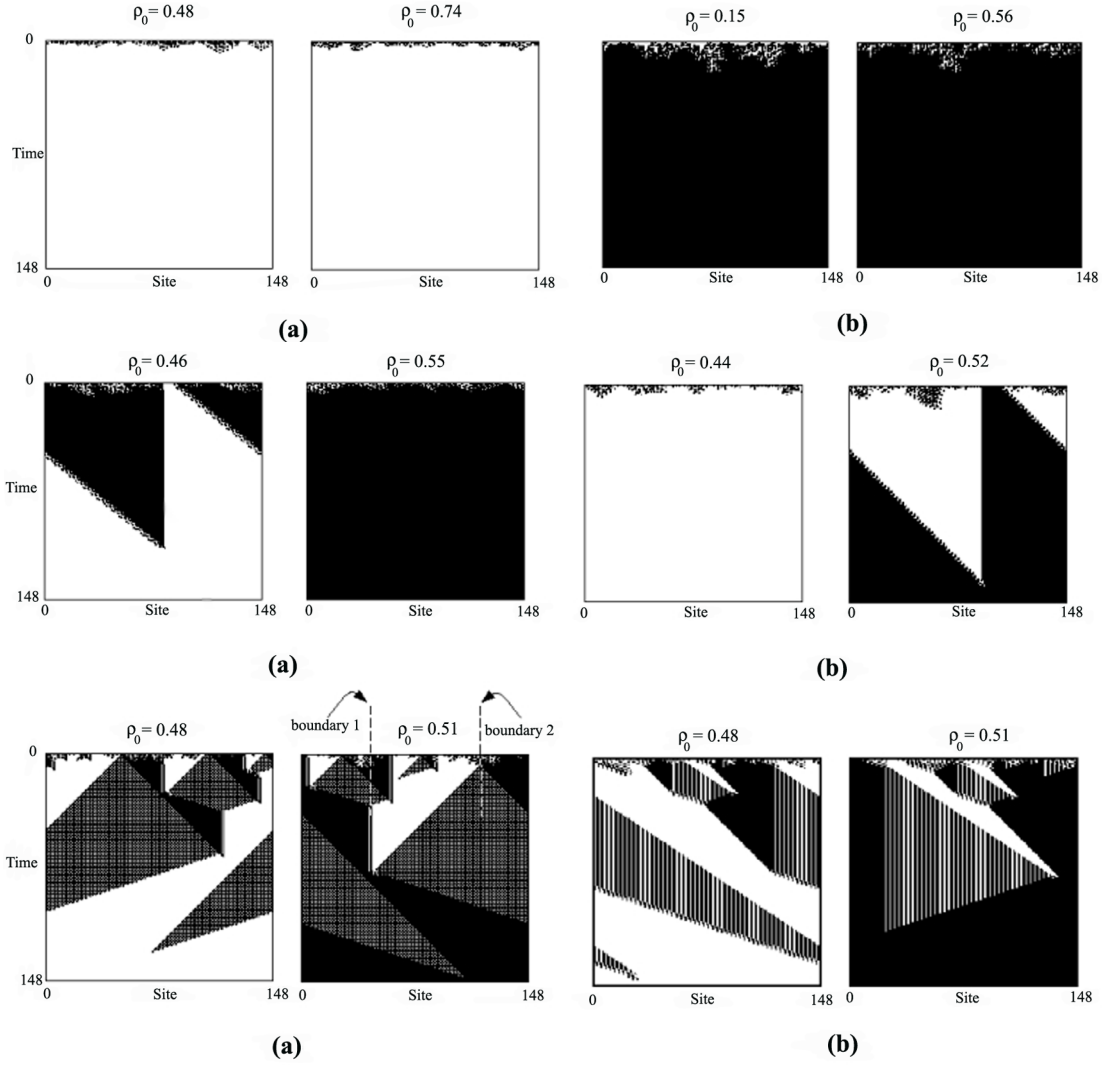
Designing an algorithm to perform this task is trivial if one is using a system with a central controller or central storage of some kind, such as a standard computer with a counter register or a neural network in which all input units are connected to a central hidden unit. However, it is nontrivial to design a small-radius ($r << N$) CA to perform this task, since a small-radius CA relies only on local interactions. Since the 1s can be distributed throughout the CA lattice, the CA must transfer information over large distances ($\approx N$), and process information collected from different parts of the lattice. To do this requires the global coordination of cells that are separated by large distances and that cannot communicate directly. This coordination must, of course, happen in the absence of any central processor or central memory directing the coordination. Land and Belew (1995) showed that no two-state CA exists that can correctly classify all possible ICs of arbitrary lengths, but did not give an upper bound on possible classification accuracy.

Crutchfield et al. (2003) describe three different CA "strategies" discovered by the GA in the course of evolution, illustrated in Fig. 3. Each plot in this figure shows a 149-cell one-dimensional lattice on the horizontal axis, with black representing cells with state "1" and white representing cells with state "0". The vertical axis shows this lattice iterating according to the CA rule, with time increasing down the page.

The three types of CA strategies are:

> *Default*: The CA always iterates to a fixed point of all-1s or always iterates to a fixed point of all-0s. A default strategy correctly classifies half of all initial configurations.
>
> *Block-expanding*: Like Default, unless the IC contains a sufficiently large block of adjacent cells of the color opposite to the default color—if so, the CA expands this block until it fills up the lattice. Block expanding strategies correctly classify between (approximately) 60–70% of all initial configurations on a 149-cell lattice, depending on the details of the particular strategy. The performance of such strategies decreases steeply with increasing lattice size.

Figure 3. Space-time diagrams illustrating the behavior of two "default" strategies (top a and b), two "block expanding" strategies (middle a and b), and two "particle" strategies (bottom, a and b), in each case starting from random initial configurations. Each (a) pair shows the behavior for a single CA rule, but starting with different ICs. Likewise for each (b) pair, but for a different CA rule. The ρ_0 value for each plot gives the density (fraction of 1s) in the IC. In the bottom "particle" strategy (a), the non-black shaded area consists of a checkerboard pattern of alternating black and white states, and in (b) the non-black shaded area consists of alternating black and white vertical stripes. (Adapted from Crutchfield et al., 2003.)



Particle: The CA uses sophisticated signals and interactions between signals to solve the problem. The particle strategies are the only ones that have high classification accuracy on a wide range of initial configurations. Particle strategies correctly classify from approximately 72% to 86% of all initial configurations on a 149-cell lattice. The performance of particle strategies decreases with increasing lattice size, but less steeply than the block expanding strategies.

CA rules also evolved that are not examples of any of these three strategies; these CAs exhibited random-like behavior and had very low performance. We will call these "random" strategies.

A *successful* run of evolution on this problem is defined here to be one in which at least one particle strategy is present in the final generation.

Using a traditional GA to evolve CA rules, Mitchell et al. (1996) reported that most runs of the GA produced both default and block-expanding strategies, but very few runs produced particle strategies. However, Pagie and Mitchell (2002) performed experiments with coevolution, using spatially distributed populations, in which the hosts were CA rules and the parasites were initial configurations. In their experiments, the fitness of a host h on a sample of n parasites is the fraction of correct classifications of h over these n parasites. The fitness of a parasite p with respect to a single host h is defined as 0 if h classifies p correctly, and $|\,\text{density}(p) - 1/2\,|$ otherwise. This is a domain-specific way to control the virulence of parasites, since the most difficult initial configurations to classify will have density $\approx 1/2$.

Pagie and Mitchell reported that spatial coevolution was significantly more successful at evolving particle strategies than traditional evolution alone, or than evolution alone with a spatially distributed population. Mitchell et al. (2006) extended Pagie and Mitchell's experiments and analysis; their methods and results are described below.

VI. EXPERIMENTS

In order to assess and understand the success of spatial coevolution compared with other evolutionary approaches, we implemented six different evolutionary methods and ran experiments using each of them. Each experiment consisted of a number of independent runs (starting with different random number seeds) of a particular evolutionary method with a given set of parameter values, described below.

The success rate of an experiment is defined as the percent of successful runs in the experiment. As was described above, a successful run on the function-induction task is one in which at least one host h has been found that is correct on each of the 676 training examples in the complete set, and has been in the population for at least 50 generations. A successful run on the cellular-automaton task is one in which at least one particle strategy is in the host population in the final generation.

Six evolutionary methods were tested.

Spatial Coevolution: The host and parasite populations are distributed on a two-dimensional grid of $M \times M$ sites, with wrap-around at the boundaries to form a torus. Each site contains a single host h and a single parasite p. At each generation, the following steps take place for all hosts and for all parasites, constituting a single generation:

1. *Fitness calculation*: Calculate the fitness of each host h in the population using nine parasites: the parasite at the same site as h and the parasites at

the eight neighboring sites. Calculate the fitness of each parasite p in the population with respect only to the host in the same site as p. The fitnesses of all individuals in a population are computed synchronously.

2. *Selection*: For each host h, rank h along with the other eight hosts in its neighborhood according to fitness, with the highest fitness host having rank 1 and the lowest having rank 9. Each of these 9 hosts has probability of being selected equal to 0.5^{rank} (except for the bottom-ranking host, which has probability 0.5^8, so that the nine probabilities will sum to 1). The selected host h' replaces the host h in the center site of this neighborhood (if h itself is selected, no change is made). The same selection procedure is applied to each parasite p, which competes similarly with the eight other parasites in its neighborhood. The replacement of hosts and parasites at each site are done synchronously.

3. *Crossover of hosts*: Each site in the grid now contains a selected host h' and a selected parasite p'. At each site, decide whether or not to perform a crossover according to the crossover probability. In our experiments, only hosts are subject to crossover. To cross over a host h', randomly choose a second host h'' at a different site in the same neighborhood, and cross over h' and h'' at a randomly selected point (in the function-induction task, exchange randomly chosen subtrees) to form two offspring. Discard one of the offspring at random; the other one replaces h' in the center site.

4. *Mutation of hosts*: Apply mutation to all hosts, according to the host mutation probability. In the function-induction task, choose a node from the tree at random and replace the function it contains by another randomly chosen function that takes the same number of arguments (and always replace a terminal by another terminal). In the cellular-automaton task, choose one or more bits and flip their values.

5. *Mutation of parasites*: Apply mutation to all parasites, according to the parasite mutation probability. For the function-induction task, choose either the x or y component and add or subtract one step of 0.4. If this mutation would result in moving the parasite outside of the [–5.0, 5.0] boundaries, the mutation is not applied and the parasite remains unchanged. For the cellular-automaton task, choose one or more bits in the parasite (initial configuration) and flip their values. Again, all crossovers, mutations, and replacements are done synchronously throughout the grid.

For the function-induction task, this process repeats until a successful host is discovered or for a maximum of 500 generations, whichever comes first. For the cellular-automaton task, this process repeats for 5000 generations.

Following Pagie and Hogeweg (1997), Pagie and Mitchell (2002), and Williams and Mitchell (2005), the parameter values we used are as follows. For the function-induction task, the grid size is 50×50. The initial population of hosts is created by generating 2500 random trees of maximum depth 3. The initial population of parasites is generated by creating 2500 random pairs (x, y) with values chosen from the domain [–5.0, 5.0] at steps of 0.4. At each generation, 40% of the host population is selected (with replacement) to undergo crossover, 20% of the host population is chosen (with replacement) to undergo a single mutation, and 10% of the parasite population is chosen (with replacement) to undergo a single mutation.

For the cellular-automaton task, the grid size is 20×20. The initial population of hosts is created by generating 400 random bit strings. The initial population of parasites is created by generating 400 bit strings of all zeros. The crossover probability is zero (i.e., no crossover is used). The host mutation probability is 0.0016 per bit and the parasite mutation probability is 0.0034 per bit.

These details could, of course, be defined differently. However, a major goal here is to better understand the results of previous work, so we adopted the same parameter values that were used in that previous work. It is important for future work to understand the sensitivity of the results to these various parameters.

Non-Spatial Coevolution: Here all parameters are as described for spatial coevolution above, but the populations of hosts and parasites are not arrayed on a grid. At each generation the fitness of each host is calculated using nine parasites randomly chosen with uniform probability across the parasite population. The fitness of each parasite is calculated with respect to a host randomly chosen from the host population. In the selection step each host is replaced via a tournament among itself and eight other hosts randomly chosen from the host population. The tournament is carried out with ranking as described above for spatial coevolution. The selection step for each parasite is done analogously. Thus, for both fitness calculation and selection, spatial locality plays no role. Crossover and mutation are carried out as described above, with the crossover partner being chosen from the eight other hosts in the same tournament.

Spatial Evolution: This method follows the same procedure for hosts as in spatial coevolution—the hosts and parasites are again arrayed in a grid. The only difference is that the parasites do not evolve. Instead, at each generation, a new randomly generated set of $M \times M$ parasites is arrayed on the grid. For the function-induction task, these parasites are generated at random as before. For the cellular-automaton task, these parasites are generated at random from a distribution that is uniform over density (all densities in [0,1] are equally likely), as described by Pagie and Mitchell (2002).

Non-Spatial Evolution: This is similar to a traditional evolutionary algorithm. For the function-induction task, in each generation the fitness of each host is calculated using 9 parasites randomly drawn from the complete set of 676 parasites. For the cellular-automaton task, in each generation the fitness of each host is calculated using 100 parasites randomly drawn from the distribution that is uniform over density. For both tasks the hosts are selected and mutated as in the non-spatial coevolution method described above. There is no evolution of parasites; at each generation a new set is randomly drawn.

Spatial Resource Sharing: Resource sharing (also called "competitive fitness sharing") was proposed by Rosin and Belew (1997) and further developed by Juillé and Pollack (1998). Under resource sharing, a host gets more credit for solving a parasite that few other hosts solve than for solving one also solved by many other hosts. Resource sharing has been shown to be necessary for success in some applications of non-spatial coevolution (Rosin and Belew, 1997; Juillé and Pollack, 1998). In particular, Werfel et al. (2000) showed that resource sharing was the mechanism largely responsible for the

success reported by Juillé and Pollack (1998). We experimented with both a spatial and non-spatial form of resource sharing. For the function-induction task the spatial form is the same as spatial evolution described above, except in this case at each generation the fitness of each host is based on all 676 (x, y) cases. For the cellular-automaton task, the spatial form is the same as spatial evolution described above, except in this case at each generation the fitness of each host is based on 100 parasites drawn at random from the uniform distribution. For both cases, let *covered* (h, p) mean that h is correct on p. Following earlier work (Juillé and Pollack, 1998), the definition of fitness used here is:

$$fitness\ (h) = \sum_{p \in parasites} \text{covered}\ (h, p) * weight_{p}$$

where

$$weight_{p} = \sum_{h \in hosts} \frac{1}{\text{covered}\ (h, p)}$$

In this method the parasite population does not evolve but is redrawn at random at each generation.

Non-spatial Resource Sharing: This is the same as spatial resource sharing, except that the hosts are selected and mutated as in the non-spatial evolution method.

VII. RESULTS

Comparing the performance of these various methods on these tasks can give us some insight into what features of spatial coevolution give rise to its success. Table 1 gives the percent of successful runs for each experiment on each task. For the function-induction task, the percentages are out of 50 runs of each method. For the cellular-automaton task, the percentages are out of either 20 or 30 runs of each method.

It can be seen that spatial coevolution is the most successful method by far for both tasks. For the function-induction task, its success rate was 78%. Non-zero success rates were also attained by spatial evolution, non-spatial evolution, and spatial resource sharing, but the rates were considerably less than that of spatial evolution. For the cellular-automaton task, spatial coevolution, with 67% successful runs, was the only method to succeed at all.

These results show that the combination of spatial distribution and coevolution is considerably more successful than either alone, or than resource sharing alone. Particularly notable is the fact that spatial coevolution produced a majority of successful runs even though each host is evaluated on only 9 parasites. Other methods, some using many more parasites to evaluate hosts, achieved much lower success than spatial coevolution (and in some cases, no success at all).

In Section 4 above, I stated the hypotheses that the success of spatial coevolution is due to two factors: its ability to maintain diversity in the populations for long periods of time and to foster continuing arms races, in which parasites continually evolve to target weaknesses in the evolving hosts. Pagie and Mitchell (2002) and Mitchell et al. (2006) gave evidence in the cellular automata experiments for both claims, which I describe below.

Table 1. Percent (and fraction) of successful runs for each experiment on the function-induction and cellular automaton tasks.

	Function Induction	Cellular Automata
Spatial Coev.	**78%** (39/50)	**67%** (20/30)
Non-Spatial Coev.	**0%** (0/50)	**0%** (0/20)
Spatial Evol.	**14%** (7/50)	**0%** (0/30)
Non-Spatial Evol.	**6%** (3/50)	**0%** (0/20)
Spatial Res. Sharing	**12%** (6/50)	**0%** (0/20)
Non-Spatial Res. Sharing	**0%** (0/50)	**0%** (0/30)

Evidence for preservation of diversity in the host population

Figure 4 gives evidence that diversity is maintained by spatial coevolution and no other method in the cellular automata experiments. Here we measure diversity at each generation in terms of the number of individuals in the population that encode each type of evolved strategy: Default, Block-expanding, and Particle. Each figure plots those numbers at each generation for a typical run of one of the six evolutionary methods. Only spatial coevolution displays all three strategies co-existing in significant numbers (i.e., exhibiting high diversity) for an extended period (approximately the last 1000 generations). In the other plots, either one strategy can be seen to dominate the population (spatial methods), or two different strategies exhibit oscillating domination (non-spatial methods). Note that in some runs, there are some generations in which the population contains very low-performance Random-strategy CAs that are not included in these plots.

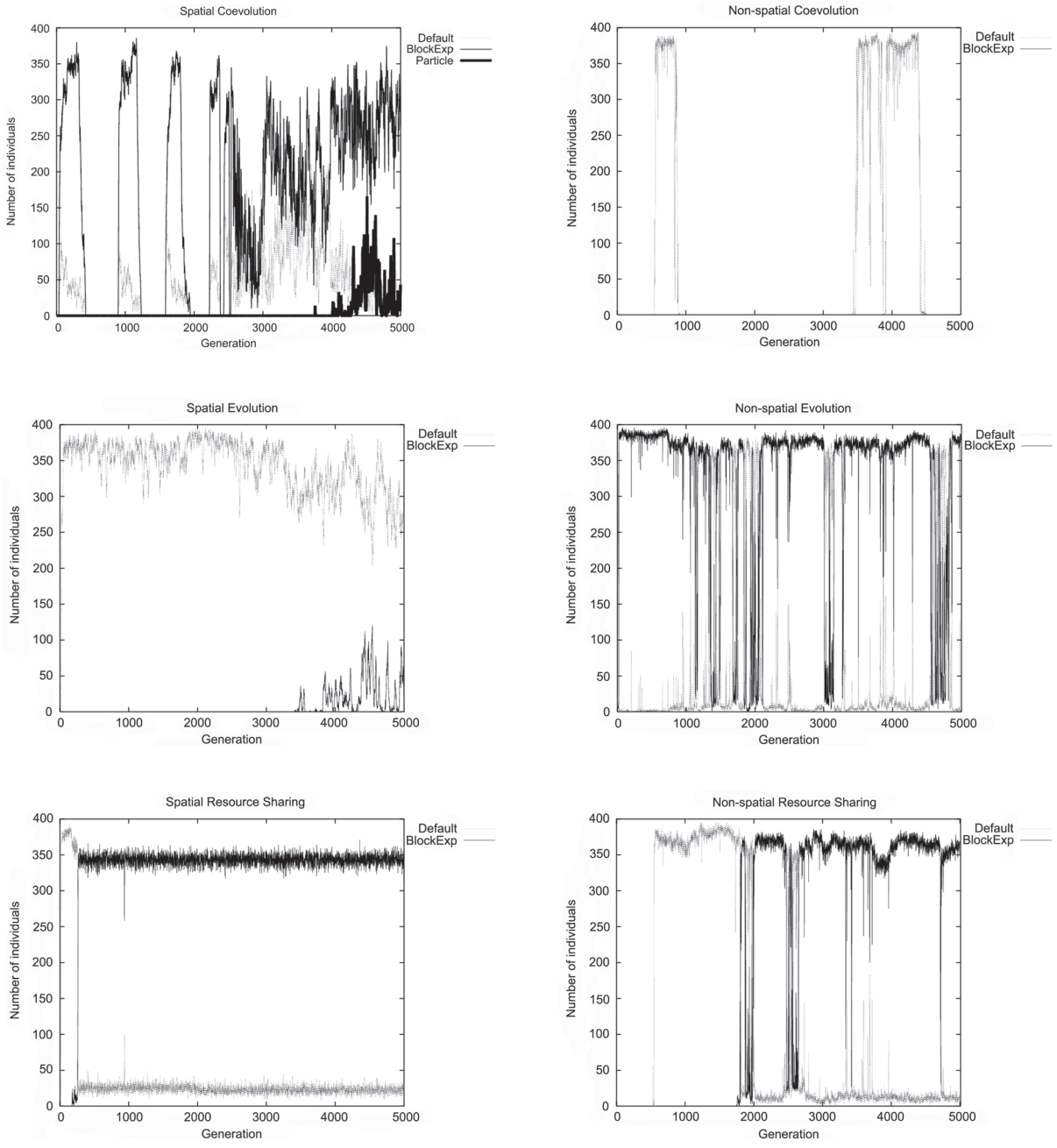
In short, spatial coevolution appears to preserve diversity in the host population more effectively than the other evolutionary methods.

Evidence for arms races between hosts and parasites

As was described above, a block-expanding strategy works by expanding sufficiently large blocks of 1s (or 0s) in the IC to produce an all-1 (all-0) state. In coevolutionary settings, such strategies can be targeted by parasites that evolve to contain "deceptive blocks": blocks of 1s in a majority-0s IC or blocks of 0s in a majority-1s IC. In Pagie and Mitchell (2002), we defined a *deceptive block* as the occurrence of a block of seven or more adjacent 0s or 1s in the IC, but only when the probability of the occurrence of such a block in a randomly generated string of the same density is less than 0.01.

Figure 5 plots, at every 10 generations, the frequency of parasites with deceptive blocks in the parasite population in a typical run of spatial coevolution. (Pagie and Mitchell's runs, from which this plot was obtained, were the same as the spatial coevolution runs described above except that the size of the spatial array was 30×30 instead of 20×20). The sections of the plot labeled "default," "block-expander," and "particle" correspond to generations at which that particular strategy was the highest performing one in the host population. It can be seen that in the period at which the best strategies

Figure 4. Number of individuals in the host population encoding the Default, Block Expanding, and Particle strategies in a typical run from each of the six experiments. The number of Particle strategies is zero for all generations in all plots except Spatial Coevolution, in which Particle strategies are discovered at approximately generation 4000 and persist for the rest of the run.



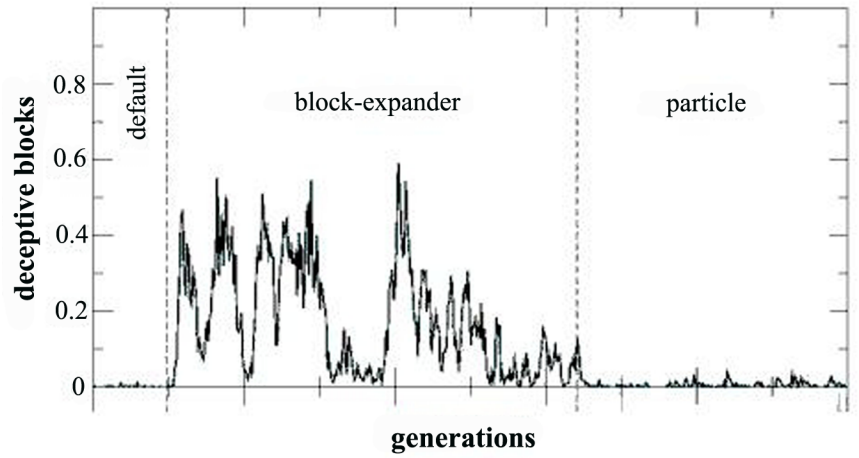


Figure 5. The frequency of ICs in the particle population that contain deceptive blocks, versus generation, for a typical run of spatial coevolution. (Values are plotted every 10 generations, for a run lasting 5000 generations.) The labels "default," "block-expander," and "particle" correspond to generations at which that particular strategy was the highest performance one in the host population. (Adapted from Pagie and Mitchell, 2002.)

are block-expanders, the parasites exploit this fact by evolving significantly higher numbers of deceptive blocks. These deceptive blocks in turn seem to push the host population to eventually evolve more sophisticated Particle strategies that are not susceptible to this exploitation. The frequency of deceptive blocks rises and falls during the block-expanding period, presumably in response to the host population switching from "expand-1s" strategies to "expand-0s" strategies, and vice versa.

In short, by tracing the dynamics of parasites with deceptive blocks, we have identified the presence of one kind of arms race between the host and parasite populations. There are also likely to be other types of arms races present in these runs, which have not yet been identified.

VIII. CONCLUSIONS AND FUTURE WORK

The results reported here show that spatial coevolution is considerably more successful on two non-trivial learning tasks than several other evolutionary methods, requires fewer training problems for success, and may be a generally effective method for overcoming impediments of the kind described in Section IV. One reason seems to be that spatial coevolution, alone among these methods, is able to maintain diversity in the host population for long periods during a run. Another reason seems to be that parasites evolve to specifically target weaknesses in host strategies, forcing hosts to evolve new strategies without those weaknesses—in short, spatial coevolution seems to foster effective arms races between hosts and parasites.

Clearly there is still much to be done to understand these results and their generality. We plan to take the following five steps in the near future: (1) Extend the earlier work of Williams and Mitchell (2005) on measuring diversity in the function-induction task; (2) Extend the methods of Pagie and Mitchell (2002) for identifying specific arms races in both tasks; (3) Understand the spatio-temporal dynamics of host innovations in all the spatial methods, as well as the spatio-temporal dynamics of parasite innovation in spa-

tial coevolution; (4) Extend our comparisons to include boosting, a machine learning method which has goals similar to coevolution in that it adapts distributions of training examples over many runs of learning; and (5) Explore the effects of the network structure of the spatial grid on the behavior of spatial coevolution. For example, how would occasional long-range interactions in the spatial grid (producing a "small world network," Watts, 1999) affect the behavior of spatial coevolution?

ACKNOWLEDGMEMTS

Many thanks to Ludo Pagie, Michael Thomure, and Nathan Williams for their contributions to this project.

REFERENCES

Axelrod, R. (1987). The evolution of strategies in the iterated Prisoner's Dilemma. In L. D. Davis (editor), *Genetic Algorithms and Simulated Annealing*, pp. 32–41. Los Altos, CA: Morgan Kaufmann.

Barricelli, N. A. (1962a). Numerical testing of evolution theories. Part I: Theoretical introduction and basic tests. *Acta Biotheoretica*, 16(1-2), 69-98.

Barricelli, N. A. (1962b). Numerical testing of evolution theories. Part II: Preliminary tests of performance, symbiogenesis, and terrestrial life. *Acta Biotheoretica*, 16(1-2), 69-98.

Cartlidge, J. and Bullock, S. (2004). Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation*,12 (2), 193–222.

Cohn, D. A., Ghahramani, Z. and Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4, 129–145.

Crutchfield, J. P., Mitchell, M., and Das, R. (2003). The evolutionary design of collective computation in cellular automata. In J. P. Crutchfield and P. K. Schuster (editors), *Evolutionary Dynamics—Exploring the Interplay of Selection, Neutrality, Accident, and Function*, pp. 361–411. New York: Oxford University Press.

Das, R., Crutchfield, J. P., Mitchell, M., and Hanson, J. E. (1995). Evolving globally synchronized cellular automata. In L. J. Eshelman (editor), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann.

Das, R., Mitchell, M., and Crutchfield, J. P. (1994). A genetic algorithm discovers particle-based computation in cellular automata. In *Parallel Problem Solving from Nature—PPSN III*, LNCS Volume 866, pp. 344–353. Berlin: Springer.

Dawkins, R. and Krebs, J. R. (1979). Arms races between and within species. *Proceedings of the Royal Society of London: Biological Sciences*, 205, 489–511.

De Jong, E. D. and Pollack, J. B. (2002). Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2), 159–192.

Fogel, D. B. (2002). *Blondie24*: *Playing at the Edge of AI*. San Francisco, CA: Morgan Kaufmann.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139.

Haith, G. L., Colombano, S. P., Lohn, J. D., and Stassinopoulos, D. (1999). Coevolution for problem simplification. In Banzhaf, W. et al. (editors), *GECCO-99: Proceedings*

of the Genetic and Evolutionary Computation Conference, pp. 244–251. San Francisco, CA: Morgan Kaufmann.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42, 228–234.

Hofmeyr, S. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation* 7(1), 1289–1296.

Juillé, H. and Pollack, J. B. (1998). Coevolving the 'ideal' trainer: Application to the discovery of cellular automata rules. *Genetic Programming 1998: Proceedings of the Third Annual Conference*. San Francisco, CA: Morgan Kaufmann.

Land, M. and Belew, R. K. (1995). No perfect two-state cellular automaton for density classification exists. *Physical Review Letters*, 74(25), 5148–5150.

Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. In C. G. Langton et al. (editors), *Artificial Life II*, pp. 295–312. Reading, MA: Addison-Wesley.

Mitchell, M., Crutchfield, J. P., and Das, R. (1996). Evolving cellular automata to perform computations: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA '96)*. Moscow, Russia: Russian Academy of Sciences.

Mitchell, M., Thomure, M. D., and Williams, N. L. (2006). The role of space in the success of coevolutionary learning. In L. M. Rocha et al. (editors), *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pp. 118–124. Cambridge, MA: MIT Press.

Nolfi, S. and Floreano, D. (1998). How co-evolution can enhance the adaptation power of artificial evolution: Implications for evolutionary robotics. In P. Husbands and J-A. Meyer (editors), *Proceedings of the First European Workshop on Evolutionary Robotics*. Lecture Notes in Computer Science, Vol. 1468. Berlin: Springer.

Pagie, L. and Hogeweg, P. (1997). Evolutionary consequences of coevolving targets. *Evolutionary Computation*, 5(4), 401–418.

Pagie, L. and Mitchell, M. (2002). A comparison of evolutionary and coevolutionary search. *International Journal of Computational Intelligence and Applications*, 2(1), 56–69.

Paredis, J. (1997). Coevolving cellular automata: Be aware of the red queen! In *Proceedings of the Seventh International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Pollack, J. B., Blair, A. D., and Land, M. (1996). Coevolution of a backgammon player. In C. G. Langton (editor), *Proceedings of the Fifth Artificial Life Conference*, pp. 92–98. Cambridge, MA: MIT Press.

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

Reed, J., Toombs, R., and Barricelli, N. A. (1967). Simulation of biological evolution and machine learning. I. Selection of self-reproducing numeric patterns by data processing machines, effects of heredity control, mutation type, and crossover. *Journal of Theoretical Biology*, 17, 319-342.

Ronge, A. and Nordahl, M. G. (1996). Genetic programs and co-evolution: Developing robust general purpose controllers using local mating in two dimensional populations. In H.-M. Voigt et al. (editors), *Parallel Problem Solving from Nature—PPSN IV*. LNCS Volume 1141, pp. 81–90. Berlin: Springer.

Rosin, C. D. (1997). *Coevolutionary Search Among Adversaries*. Ph.D. Dissertation, Computer Science Department, U. C. San Diego, La Jolla, CA.

Shapiro, J. L. (1998). Does data-model co-evolution improve generalization performance of evolving learners? In *Lecture Notes in Computer Science*, Vol. 1498, pp. 540–549. Berlin: Springer.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. In Brooks, R. and Maes, P. (editors), *Proceedings of Artificial Life IV*, pp. 28–39. Cambridge, MA: MIT Press.

Turner, J. R. G. (1981). Adaptation and evolution in *Heliconius*: A defense of Neo-Darwinism. *Annual Review of Ecology and Systematics*, 12, 99–121.

Watts, D. J. (1999). *Small Worlds: The Dynamics of Networks Between Order and Randomn*ess. Princeton, NJ: Princeton University Press.

Werfel, J. and Mitchell, M. and Crutchfield, J. P. (2000). Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, 4(4), 388–393.

Wiegand, R. P. and Sarma, J. (2004). Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In Xin, Y. et al. (editors), *Parallel Problem Solving from Nature—PPSN VIII*. LNCS Vol. 3242, pp. 912-921. Berlin: Springer.

Williams, N. L. and Mitchell, M. (2005). Investigating the success of spatial coevolutionary learning. In H.-G. Beyer et al. (editors). *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO-2005*, pp. 523–530. New York: ACM Press.