The Role of Space in the Success of Coevolutionary Learning

Melanie Mitchell[1], Michael D. Thomure[1], and Nathan L. Williams[2]
[1]Portland State University, Portland, OR 97207
[2]Veriwave, Inc., Beaverton, OR 97008
mm@cs.pdx.edu

To appear in *Proceedings of ALIFE X—The Tenth International Conference on the Simulation and Synthesis of Living Systems, 2006*.

Abstract

We explore the role of spatial distribution of populations in coevolutionary learning by comparing the performance of several different evolutionary methods on two different tasks. Our experiments show that spatial coevolution is substantially more successful than any of the other methods we tried. One reason for this, demonstrated by our results, is the ability of spatial coevolution to maintain diversity in the host population for long periods during a run.

Introduction

Many approaches to adaptive computation require that a system learn from examples, often in the form of problems that the system must solve. How are these training problems to be chosen? If the problems are too easy or to difficult, the system is often not able to 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. Approaches to this issue in the machine learning community include active learning (Cohn 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 paper is to increase this understanding by exploring the role of spatially extended populations in the success of coevolution.

In coevolutionary learning, two populations of individuals 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 the coevolution of game-playing strategies (Rosin and Belew, 1997), or cooperative, as in the cooperative coevolution of neural network weights (Potter and De Jong, 2000). The idea of competitive "host-parasite" coevolutionary learning was originally due to Hillis (1990), who proposed it as a way to optimally adapt training problems ("parasites") to evolving programs ("hosts"). This is the form of coevolution we will examine here.

It has been hypothesized that the success of coevolution is due to three factors: the constantly changing environment produced for each population by the other, the maintenance of diversity in the host and parasite populations, and "arms races" that develop between hosts and parasites. In such an arms race, the hosts evolve to successfully solve the problems encoded as parasites, forcing the parasites to evolve to be more difficult, targeting specific weaknesses of the hosts. This forces the hosts to evolve to solve these new problems, and so on. The hoped-for result is that the hosts will be forced to continue improving, evolving to overcome weaknesses, and finally be able to solve general problems in their domain of learning.

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 an initial host population that already possessed some features of optimal sorting networks. Rosin and Belew (1997) demonstrated that the 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 fit-

ness 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 the heuristic of resource sharing 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 largely due to resource sharing rather than coevolution. De-Jong and Pollack (2004) use 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 problems associated with coevolution alone. These problems have been pointed out by various researchers (Cartlidge and Bullock, 2004; De Jong and Pollack, 2002; Nolfi and Floreano, 1998; Paredis, 1997), sometimes using different terminology. The following is a partial list of these problems:

Loss of gradients: Coevolution leads to a state in which the parasite population becomes too easy (called "disengagement") or too difficult (called "overvirulence") 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 (or *over-fitting*): 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.

Intransitivity: The host or parasite population reaches a state in which fitness is not transitive: that is, there exist hosts or parasites A, B, and C such that fitness(A) $>$ fitness(B) and fitness(B) $>$ fitness(C), but fitness(C) $>$ fitness(A).

Can these problems be overcome without adding computationally expensive, ad hoc heuristics to coevolution? Here we investigate 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.

Several experiments employing spatial coevolution have given evidence for this hypothesis (Hillis, 1990; 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 extent significantly improves the performance of coevolution on numerous problems.

This paper is an extension of work by Pagie & Mitchell (2002) and Williams & Mitchell (2005). Both these groups found dramatic improvements in success when using spatial coevolution, as compared to other evolutionary methods. Here we continue the attempt 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. Our contribution in this paper is to give results from comparisons beyond those reported by Pagie & Mitchell and Williams & Mitchell, and to give evidence for the generality of our conclusions by correlating the behavior of these various methods between the two different tasks.

Problem domains

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 $\{x,y,\mathcal{R}\}$, where $\mathcal{R}$ returns a random constant in [-1.0, 1.0] every time a node containing it is generated. Details of the representation can be found in (Williams and Mitchell, 2005). Training examples (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 training examples (x,y) are assigned values $h(x,y)$ that are close to the target $f(x,y)$.

The "error" of a host h on a parasite $p = (x,y)$ is the absolute value of the difference between the target function's value on p and the host's value: $|f(x,y) - h(x,y)|$.

A host h is said to be "correct" on 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.

Evolving cellular automata

We use the same task that was described in detail in (Pagie and Mitchell, 2002). The goal is to discover a one-dimensional, radius 3, binary-state cellular automaton rule h that produces the following behavior, given binary initial configuration p:

- If p contains a majority of 1s (i.e., p is high density), then h should iterate to a global fixed-point configuration of all 1s. This fixed point must be reached within $2L$ time steps for a lattice of L cells.
- If p contains a majority of 0s (i.e., p is low density), then h should iterate to a global fixed-point configuration of all 0s. Again, this fixed point must be reached within $2L$ time steps.

If host h exhibits this behavior on parasite p, h is said to "correctly classify" p or, alternatively, h is said to be "correct" on p.

As discussed by Das et al. (1994), performing this task correctly over a wide range of p is a non-trivial task for a CA, which can use only local communication to discover a global property of the initial lattice.

The population of candidate solutions (hosts) consists of 128-bit strings representing cellular-automaton rule tables h. Training examples (parasites) are binary initial configurations p of length 149.

The fitness of a host h on a sample of n parasites $\{p_1, p_2, ..., p_n\}$ is the fraction of correct classifications of h over these n parasites.

For coevolutionary methods, the fitness of a parasite p with respect to a single host h is defined as 0 if h classifies p correctly, and $|density(p) - 1/2|$ 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 described three types of CA strategies that are discovered by evolutionary methods for this task: *Default* (the CA always iterates to a fixed point of all-1s or always iterates to a fixed point of all-0s); *Block-expanding* (like Default, unless p 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); and *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 (Das et al., 1994).

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. For the function-induction task, a successful run is one in which at least one host h has been found that is correct on each of the 676 parasites in the complete set, and has been in the population for at least 50 generations. For the cellular-automaton task, a successful run is one in which a Particle strategy is in the host population in the final generation. It would possibly have been better for our definition of success on the cellular-automaton task to be similar to that on the function induction task (discovery of a Particle strategy in *any* generation followed by its persistence for at least 50 generations). However, due to the computational expense of determining the strategies of a large population of CA rules, we used a rougher indicator of success: the presence of a Particle strategy in the final generation. We believe this indicator is well correlated with the more precise indicators, and plan to test this in future research.

In defining the methods and parameters described below, we follow (Williams and Mitchell, 2005) and (Pagie and Mitchell, 2002).

Spatial Coevolution: The host and parasite populations are distributed on a two-dimensional square grid of *NxN* 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 para-

site 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. 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:* Each site in the grid now contains the selected host h' and the 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 it by another node 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 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; Williams and Mitchell, 2005) and (Pagie and Mitchell, 2002), the parameter values we used are as follows. For the function-induction task, the grid size is 50*x*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*x*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, our 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 9 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 8 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 8 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 *NxN* 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 in (Pagie and Mitchell, 2002).

Non-spatial Evolution: This is equivalent 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 was calculated using 100 parasites randomly drawn from the distribution that is uniform over density. For both tasks the hosts were selected and mutated as in the non-spatial coevolution method described above. There was no evolution of parasites; at each generation a new set was 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; Werfel et al., 2000). In particular, Werfel, Mitchell, and Crutchfield (2000) showed that resource sharing was the mechanism largely responsible for the success reported in (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 was the same as spatial evolution described above, except in this case at each generation the fitness of each host was based on all 676 (x,y) cases. For the cellular-automaton task, the spatial form was the same as spatial evolution described above, except in this case at each generation the fitness of each host was 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; Werfel et al., 2000; Williams and Mitchell, 2005), the definition of fitness used here was

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

where

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

In this method the parasite population did not evolve but was redrawn at random at each generation.

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

	Function Induction	Cellular Automaton
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. Shar.	12% (6/50)	0% (0/20)
Non-Spatial Res. Shar.	0% (0/50)	0% (0/30)

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

Results and Discussion

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. The results for spatial coevolution, spatial evolution, and spatial resource sharing are taken from (Williams and Mitchell, 2005). The implementation of and results of experiments with non-spatial methods are reported for the first time here. For the cellular-automaton task, the percentages are out of either 20 or 30 runs of each method. To obtain these results, we independently re-implemented the spatial coevolution and spatial evolution methods described in (Pagie and Mitchell, 2002); the results for these experiments are from these new simulations. The only difference between our implementation and that of (Pagie and Mitchell, 2002) was the grid size: 20x20 in our case and 30x30 in theirs. This is what likely accounts for the lower success rates reported here. The implementation of and results of experiments with all the other methods are reported for the first time here.

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, using much higher numbers of parasites to evaluate hosts, achieved much lower success than spatial coevolution (and in some cases, no success at all).

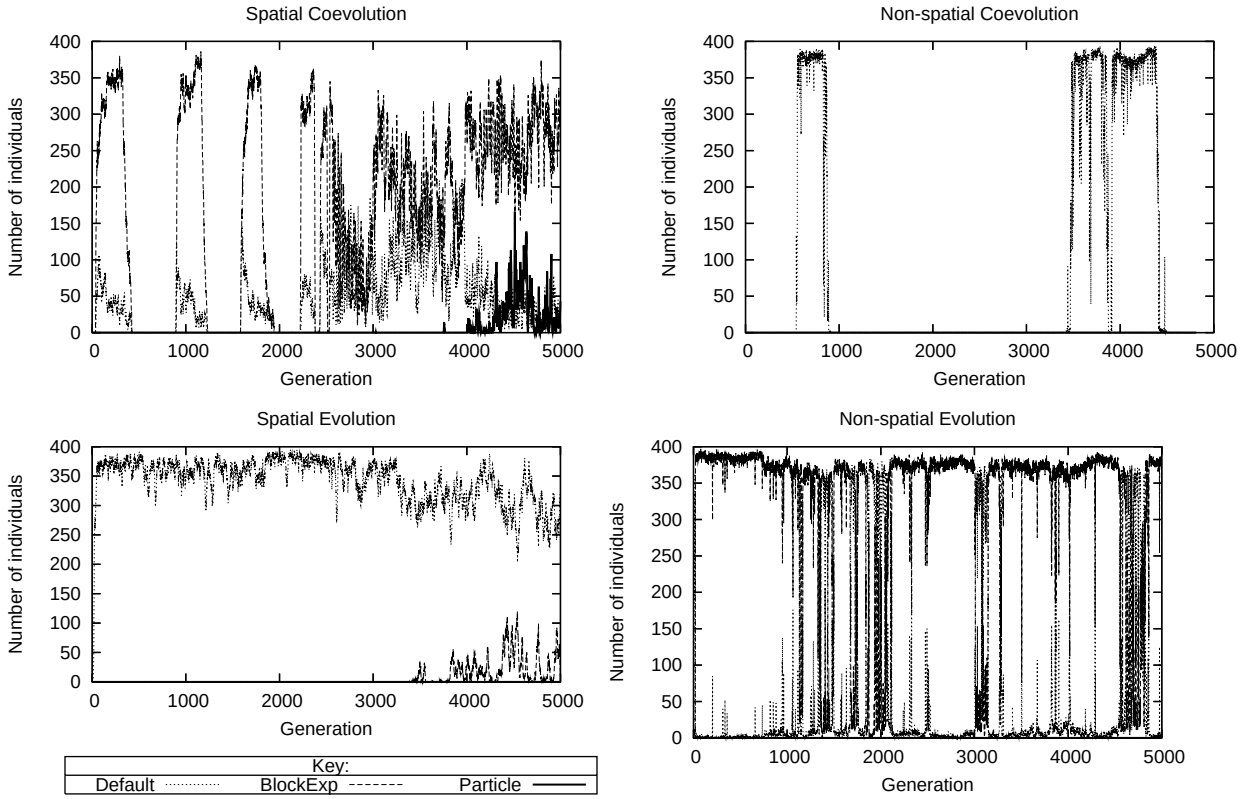

Figure 1: Number of individuals in the host population encoding the Default, Block Expanding, and Particle strategies in a typical run from each of the six experiments. The number of Particle strategies is zero for all generations in all plots except Spatial Coevolution, in which Particle strategies are discovered at approximately generation 4000 and persist for the rest of the run. (See next page for continuation of this figure.)

Pagie & Mitchell (2002) and Williams & Mitchell (2005) give evidence that there are two main factors giving rise to the success of spatial coevolution: 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. Here we give additional evidence for the diversity claim. The arms-race claim and the specific role of spatial distribution in fostering arms races will be further explored in a longer paper.

Figure 1 gives evidence that diversity is maintained by spatial coevolution and no other method in the cellular-automaton-induction 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, or two different strategies exhibit oscillating domination. (Note that in some runs, there are some generations in which the population contains very low-performance CAs that do not encode any of the three strategies and are not included in these plots.)

Conclusions and Future Work

We have shown that spatial coevolution is considerably more successful on two non-trivial learning tasks than several other evolutionary methods. 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. Others (Pagie and Mitchell, 2002; Williams and Mitchell, 2005) have given evidence that spatial coevolution also fosters 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 & Mitchell (2005) on measuring diversity in the function-induction task; (2) Extend the methods of Pagie & Mitchell (2002) for identifying specific arms races in both tasks; (3) Understand

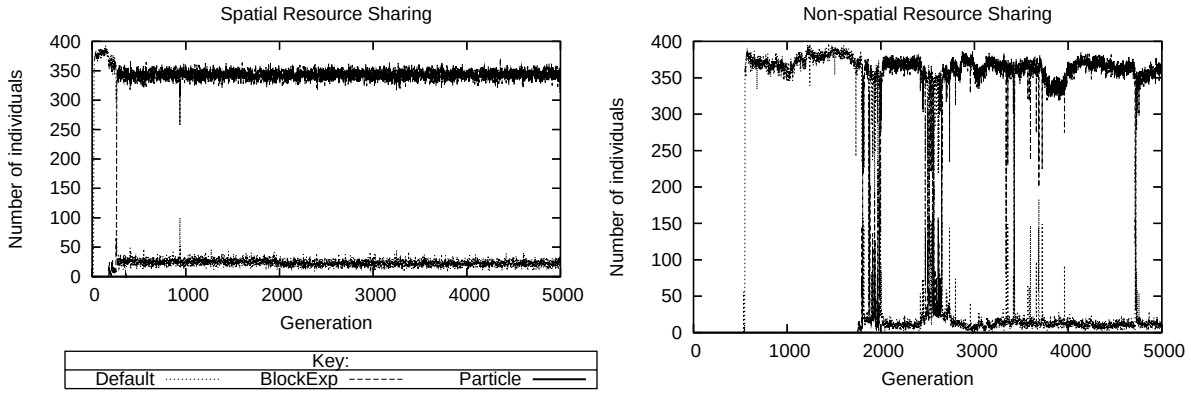

Figure 1: continued

the spatio-temporal dynamics of host innovations in all the spatial methods, as well as the spatio-temporal dynamics of parasite innovation in spatial 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?

References

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.

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*, volume 866, pages 344–353, Berlin. Springer-Verlag (Lecture Notes in Computer Science).

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

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.

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

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

Nolfi, S. and Floreano, D. (1998). How co-evolution can enhance the adaptation power of artificial evolution: Implications for evolutionary robotics. In *Proceedings of the First European Workshop on Evolutionary Robotics*.

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.

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

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 *Parallel Problem Solving from Nature—PPSN IV*, pages 81–90. Springer-Verlag.

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

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

Werfel, J., 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 *Parallel Problem Solving from Nature—PPSN VIII*, pages 912–921. Springer-Verlag.

Williams, N. L. and Mitchell, M. (2005). Investigating the success of spatial coevolutionary learning. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO-2005*, pages 523–530, New York. ACM Press.