

Investigating the Success of Spatial Coevolution

Nathan Williams
Veriwave, Inc
9600 SW Oak Street
Portland, OR 97223
503-473-8350
nathanw@rebeltribe.com

Melanie Mitchell
Department of Computer Science
Portland State University
Portland, OR 97207-0715
503-725-2412
mm@cs.pdx.edu

In H. G. Beyer et al. (editors), *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO-2005*, New York: ACM Press, 523–530.

ABSTRACT

We investigate the results of coevolution of spatially distributed populations. In particular, we describe work in which a simple function approximation problem is used to compare different spatial evolutionary methods. Our work shows that, on this problem, spatial coevolution is dramatically more successful than any other spatial evolutionary scheme we tested. Our results support two hypotheses about the source of spatial coevolution’s superior performance: (1) spatial coevolution allows population diversity to persist over many generations; and (2) spatial coevolution produces training examples (“parasites”) that specifically target weaknesses in models (“hosts”). The precise mechanisms by which the combination of spatial embedding and coevolution produces these results are still not well understood.

1. INTRODUCTION

Hillis [2] was the first to propose the notion of “host-parasite” coevolution. In his scheme, a population of candidate solutions (sorting networks) coevolved with a population of test sets (groups of sorting problems). The fitness of an individual sorting network (“host”) depended upon how many sorting problems in an individual group (“parasite”) it successfully sorted. Likewise, the fitness of an individual sorting-problem group depended upon how many problems in the group a network did not successfully sort. Hillis embedded the populations of hosts and parasites in a two-dimensional spatial grid with one host and one parasite per grid location. The fitness of each host was evaluated only on the parasite in the same grid location, and vice versa. This coevolution scheme was able to discover a smaller correct sorting network than a scheme using evolution alone.

Since Hillis’s work, several other examples of the advantages of coevolution have been demonstrated (e.g., [1, 3, 7, 9]). Coevolution has been shown, in many cases, to outperform traditional evolutionary methods in both quality

of final solutions and in efficiency, since coevolution allows for sparse training rather than relying on large training sets.

However, it is still not completely understood what factors lead to the success of coevolution. It has been hypothesized that coevolution provides an adaptable gradient for learning so that the host and parasite populations evolve to continually provide a challenge for one another. It is thought that each population is able to focus on strategic weaknesses in the other population, forcing the other population to overcome these weaknesses, in analogy with biological arms races. It is also thought that the requirement of continually adapting in response to continual changes in the opposing population produces persistent diversity in each population, which is known to be an important factor for success in evolutionary computation.

Some studies have shown that coevolution does not necessarily lead to better results for every task [10]. For example, coevolution can lead to a loss of gradient. This happens when one coevolving population evolves to greatly outperform the other population; uniformly high or low fitness values in the respective populations provide no gradient for selecting the fittest individuals. Additionally, coevolution can lead to over-fitting, in which one population focuses on the weaknesses in the other population without generalizing the solution. Finally, coevolution can lead to relativism, in which the populations oscillate between strategies but do not make progress towards a general solution.

Much of the work on coevolution since Hillis’s has investigated non-spatial coevolution: each host is evaluated on the complete population of parasites or on some non-spatially derived subset of parasites. Coevolution has been found to be successful in such non-spatial systems, but apparently only with the addition of certain other mechanisms for explicitly maintaining population diversity and dealing with the other problems listed above. These added mechanisms include competitive fitness sharing [9], resource sharing [3], a “phantom parasite” [8], explicit control of parasite virulence [1], and the use of domain-

specific knowledge in fitness evaluations [3, 7]. It is not clear to what extent these additional mechanisms are responsible for the success attributed to coevolution. For example, Werfel et al. [11] found that resource sharing, not coevolution, was the major factor underlying Juillé and Pollack’s [3] success on coevolving cellular automata.

Are there any circumstances under which coevolution would be successful without such additional mechanisms? Hillis’s work seems to be one example. Notably, Hillis’s system employed the additional factor of spatial embedding, whose contribution to the success of coevolution was not analyzed. His system also incorporated other complicating factors in the representation and initialization of sorting networks.

In this paper we extend the work of Pagie and Hogeweg [6] on investigating a spatially embedded coevolutionary system without the additional mechanisms described above. We attempt to isolate what factors give rise to the dramatic success of spatial coevolution.

We examine two hypotheses for the factors producing the success of spatial coevolution: (1) spatial coevolution promotes continued diversity in the population and (2) spatial coevolution produces parasites that target specific weaknesses in hosts.

2. Experimental Setup

Our experimental setup is identical to that used by Pagie and Hogeweg [6] except for some minor differences identified below.

We use the same function-approximation task studied by Pagie and Hogeweg and their tree representation of candidate solutions. The target function is the following two-dimensional equation:

$$t(x, y) = 1 / \left(1 + x^{-4} \right) + 1 / \left(1 + y^{-4} \right),$$

where $x, y \in [-5.0, 5.0]$. A plot of the target function is

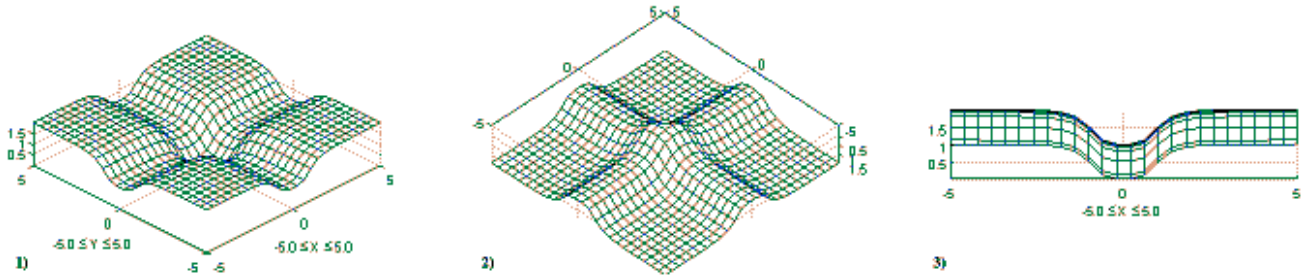


Figure 1: Plots of the target function with views from 1) the top, 2) the bottom, and 3) the side. The maximum of the function approaches 2.0 at the corners and the minimum approaches 0 at (0,0).

shown in Figure 1.

Each candidate solution (or “host”) is a tree representing a mathematical equation. The goal is to construct a tree that approximates the target equation within a certain tolerance. The problems upon which fitness is based are simple (x, y) values evenly distributed over the problem domain at intervals of 0.4. Therefore, the complete set consists of $26 \times 26 = 676$ total problems.

The fitness of a host is based on its weighted error when evaluated on a subset of problems. The particular subset of problems on which the hosts are evaluated depends upon the evolutionary method used. The *true fitness* of a host h is defined to be the average error of the host on the complete set of 676 problems:

$$tf_h = \left(\sum_{i=1}^{S_{\text{Parasites}}} |t(p_i) - h(p_i)| \right) / S_{\text{Parasites}},$$

where $S_{\text{Parasites}}$ is the number of problems in the complete set of problems, $t(p_i)$ is the value of the target function on problem p_i , and $h(p_i)$ is the value of a host on problem p_i .

The hosts are represented, using the genetic programming paradigm, as trees describing mathematical equations that can be evaluated on two-dimensional points. The function set is $\{+, -, *, \%\}$ and the terminal set is $\{X, Y, \mathfrak{R}\}$. The $+$, $-$, and $*$ operators are the standard arithmetic functions for addition, subtraction, and multiplication respectively; each takes two arguments. The protected divide 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. The terminal \mathfrak{R} is the “ephemeral random constant” [4], a randomly generated value between -1.0 and 1.0. The ephemeral random constant’s value for the node is randomly generated when the node is created and is fixed for the lifetime of the node. Note that the function and terminal sets contain unnecessary elements. In fact, the only necessary elements are the terminals X and Y, protected division, and either addition or subtraction. Additionally, while the constant 1.0 is not directly available as a terminal, it can be created by dividing a number by itself or by

dividing a number by zero. The initial population of hosts is randomly generated with a maximum depth of three nodes.

The coevolutionary methods utilize a population of “parasites”: (x, y) coordinate pairs from the complete set of all problems. The fitness of a parasite is defined by the evolutionary method used. Pagie and Hogeweg initialized all parasites to $(0.2, 0.2)$. However, we found that the coevolutionary methods were more successful when the initial members of the parasite population were randomly chosen with replacement from the complete set of 676 possible problems, and the results we report are from runs with this random initialization.

In our experiments, the host and the parasite populations are spatially embedded on a 50×50 two-dimensional toroidal lattice, with one host and one parasite per grid cell. Thus, the total population sizes are 2500 individuals. At each generation, the host population undergoes selection, crossover, and mutation, and the parasite population undergoes selection and mutation. These are described below.

Competition between individuals for reproduction is local in space. Each host or parasite competes for survival against the surrounding 8 hosts or parasites in its 3×3 neighborhood. After each population has been evaluated to determine fitness using one of the evolutionary methods (described in the next section), the individuals in each 3×3 neighborhood are ranked according to their fitness values. The i th ranked individual is selected with probability 0.5^i . The ninth element is selected with probability 0.5^8 to ensure that the nine probabilities sum to unity. The selected individual (acted upon further by crossover and mutation) replaces the individual in the center of the neighborhood in the next generation. All replacements are made simultaneously before the start of the next generation.

Following the selection step, 40% of the selected host population is chosen with replacement to undergo crossover with a randomly chosen neighboring host (including itself) from among the original neighboring individuals with which it competed for selection. Host crossover replaces a randomly chosen sub-tree in the selected host with a copy of a randomly chosen sub-tree from the other host, leaving the selected host modified but the other host unchanged. There is no parasite crossover.

Following the crossover step, 20% of the selected host population is chosen with replacement to undergo point mutation. Additionally, at each generation 10% of the parasites are chosen to undergo mutation. Parasite mutation modifies either the x or y component of the coordinate pair by adding or subtracting one step (0.4). However, when a parasite on the edge of the problem space is mutated so that its new value would be outside of the problem space, the parasite remains unchanged.

A host is considered to be *correct* if for each of the 676 problems in the complete problem set, the absolute

difference between the target function and the host function is less than or equal to a tolerance of 0.01:

$$\forall p, p \in \text{Parasites}, |t(p) - h(p)| \leq 0.01.$$

A *stable* correct host is one that has been discovered and then remains in the population for at least 50 generations. The algorithm stops when a stable correct host has been discovered or after 500 generations if none have been discovered. The goal of each run is to produce a stable correct host; therefore, runs that do so are considered to be successful.

The hosts have a rich representation space; thus equivalent equations can be expressed with different trees. The particular representing a host’s equation is called the host’s *genotype*. The actual set of problems the host solves is its *phenotype*. Thus, the hosts can have many genotype encodings for a single phenotype. In this work, we define phenotype as the subset of the complete set of 676 problems for which the error of the host with respect to the target function is less than or equal to the tolerance of 0.01:

Host h ’s phenotype: set of all p such that

$$p \in \text{Parasites} \text{ and } |t(p) - h(p)| \leq 0.01.$$

3. Evolutionary Methods

This research compares a variety of spatial evolutionary methods. We replicate the original methods used by Pagie and Hogeweg: coevolution, complete evaluation, and random evaluation. However, we also add two additional methods, host-against-all coevolution and resource sharing, as additional control experiments. All methods utilize spatially embedded hosts. Below we define the fitness functions of hosts and parasites for each of the different spatial evolutionary methods that we investigated. Note that since a host’s “fitness” is defined in terms of its error with respect to the target function, each evolutionary method attempts to *minimize* host fitness f_h and *maximize* parasite fitness f_p .

3.1 Coevolution

Let $p = (x, y)$, $x \in [-5.0, 5.0]$, $y \in [-5.0, 5.0]$ be a particular problem.

Host fitness:

$$f_h = \left(\sum_{i=1}^9 |t(p_i) - h(p_i)| \right) / 9,$$

such that p_i is in h ’s neighborhood.

The fitness of a host is the host’s average error on the nine problems defined by the neighboring parasites, where $t(p)$ is the target function value on problem p and $h(p)$ is the host function’s value on problem p . The fitness evaluation is sparse: at each generation, each host is evaluated on only 9 problems out of the complete set of 676 problems.

Parasite fitness:

$$f_p = |t(p) - h(p)|,$$

such that h is the host in the same grid cell as p .

The fitness of a parasite p is calculated with respect to the host h in the same grid cell and is the absolute value of the error of the host on that problem. Note that while a host's fitness is a function of its value on the nine parasites in its neighborhood, a parasite's fitness depends only on the single host in its same grid location. Pagie and Hogeweg claim that such an asymmetric fitness evaluation of parasites produces better results, in terms of time to discover a good solution, than a symmetric fitness evaluation in which the parasite's fitness depends upon the nine neighboring hosts [6].

3.2 Complete Evaluation

Complete evaluation (called "static evaluation" in [6]) is a non-coevolutionary spatial method to which coevolution is to be compared. Under complete evaluation, there is no parasite population. Rather, the host is evaluated against all 676 problems in the problem space ($S_{Problems}$) with equal weighting.

Host fitness:

$$f_h = \left(\sum_{i=1}^{S_{Problems}} |t(p_i) - h(p_i)| \right) / S_{Problems}$$

3.3 Random Evaluation

Random evaluation was chosen as a control to determine if coevolution's success as compared with complete evaluation was due only to sparse evaluation (evaluation against a small subset of the problem space) rather than other aspects of coevolution. Under random evaluation there is no parasite population, but sparse evaluation is still maintained. For each generation and for each host, nine random problems are chosen with replacement from the complete problem set on which each host's fitness is calculated.

Host fitness:

$$f_h = \left(\sum_{i=1}^9 |t(p_i) - h(p_i)| \right) / 9,$$

such that p_i is chosen randomly from $S_{Problems}$.

3.4 Host-Against-All Coevolution

Host-against-all coevolution was chosen as a control experiment to determine the effects of spatial locality of parasites in dynamically exploiting specific weaknesses in specific hosts. Under host-against-all coevolution, a parasite population exists, but parasites are no longer local to their hosts. Instead, each host is evaluated on all 2500 parasites in the parasite population ($n_{Parasites}$). However, the hosts and parasites remain embedded and the fitness of a parasite is

still the error of the host in the same grid cell. Host selection and crossover still occurs between neighboring individuals on the spatial lattice. Thus, some locality is maintained.

Host fitness:

$$f_h = \left(\sum_{i=1}^{n_{Parasites}} |t(p_i) - h(p_i)| \right) / n_{Parasites}$$

Parasite fitness:

$$f_p(h) = |t(p) - h(p)|,$$

such that h is the host in the same grid cell as p .

3.5 Resource Sharing

Resource sharing (or competitive fitness sharing) is a method for promoting diversity in the population by giving higher weight to problems that fewer hosts solve compared with problems that many hosts solve [3, 9]. We performed a control experiment using resource sharing alone to determine the role population diversity plays in evolving stable correct solutions. Resource sharing provides an alternative problem-weighting mechanism hypothesized to behave similarly to how parasites exploit the hosts during coevolution. The credit each host earns for solving a particular problem is shared equally among all the hosts that solve that particular problem. Thus, the more hosts that solve a problem, the less credit each host receives for solving that problem.

Host fitness:

$$f_h = \sum_{i=1}^{S_{Problems}} (weight(p_i) * covered(h, p_i)),$$

where

$$weight(p) = 1 / \left(\sum_{i=1}^{n_{Hosts}} covered(h, p) \right), \text{ and}$$

$$covered(h, p) = \begin{cases} 1 & \text{if } |t(p) - h(p)| \leq 0.01 \\ 0 & \text{if } |t(p) - h(p)| > 0.01 \end{cases}$$

The fitness of a host is the sum of the weights of each problem in the complete set of problems on which the host matches the target function within the accepted tolerance. The $weight(p)$ of each problem p depends upon how many other hosts solve p : the fitness a host receives from solving the problem is shared among all hosts that solve it. The function $covered(h, p)$ returns 1 if the error of host h on problem p is less than the accepted tolerance of 0.01, and 0 otherwise.

4. RESULTS

4.1 Pagie & Hogeweg's Experiments

The results of Pagie and Hogeweg's experiments are summarized in Table 1. The "success rate" is the percentage of successful runs (i.e., runs on which a stable,

correct solution was evolved). Coevolution was successful in approximately 45% of the time.

Complete evaluation did not produce any successful runs.

Evolutionary Method	Mean Number of Nodes in Best Evolved Hosts	Standard Deviation
Coevolution	101.28	133.56
Complete Evaluation	1301.92	1320.22
Random Evaluation	1548.76	2242.02
Host-Against-All Coevolution	628.58	824.65
Resource Sharing	955.44	1597.92

Table 1: Pagie and Hogeweg’s results comparing three different evolutionary methods. The success rate is the percentage of runs that produced a stable correct host, out of 20 total runs.

Random evaluation’s success rate was 35%, which is roughly comparable to that of coevolution. Pagie and Hogeweg reasoned that in both of these methods, sparse evaluation allows the evolutionary process to explore the solution space more freely than does complete evaluation. In particular, it frees hosts from the requirement of reducing the error on all problems at once; rather, it allows them to focus on particular subsets of problems.

4.2 Results of Our Experiments

We performed 50 independent runs of each of the five methods described above. The results are summarized in Table 2. During each run we recorded the best individual at each generation based upon true fitness (average error over the complete set of 676 problems). Ties were broken by which host had been stable the longest, then by which host had the fewest number of nodes, and finally by which host was encountered first in a pass over the spatial grid.

Evolutionary Method	Success Rate
Coevolution	39 / 50 (78%)
Complete Evaluation	0 / 50 (0%)
Random Evaluation	7 / 50 (14%)
Host-Against-All Coevolution	26 / 50 (52%)
Resource Sharing	6 / 50 (12%)

Table 2. Results of the five different evolutionary methods. The success rate is the percentage of runs that produced a stable correct host, out of 50 total runs.

Coevolution outperformed all other methods significantly, with a success rate of 78%. The next best method was host-against-all coevolution with a 52% success rate. Resource sharing and random evaluation performed approximately equally well with a 12% success rate for resource sharing and a 14% success rate for random evaluation. Complete evaluation did not produce any successful runs.

Evolutionary Method	Success Rate	Mean Number of Nodes in Best Evolved Host
Coevolution	9 / 20 (45%)	44
Complete Evaluation	0 / 20 (0%)	68
Random Evaluation	7 / 20 (35%)	not reported

Table 3: Results of the size of hosts evolved by the five different evolutionary methods averaged over 50 runs.

The average number of nodes in the best evolved hosts of each evolutionary method is summarized in Table 3. The average size of hosts in a population directly affects the time needed to calculate host fitnesses. Thus, methods that produce smaller hosts run faster and consume fewer resources. Coevolution produced significantly smaller hosts as compared with the other methods, largely due to the fact that stable correct hosts tend to be smaller than other hosts.

All five evolutionary methods are able to gradually improve the true fitness of the hosts to some degree during each run. We observed that in successful runs, the best true fitness in the population shows a very gradual increase, followed by an abrupt jump (down) to a better fitness. Figure 2 shows the fitness of the best individual in each generation for all 50 runs of coevolution. All evolutionary methods, with the exception of complete evaluation, exhibit similar behavior, with the sole difference being the number of stable correct solutions discovered. In all fitness evaluation methods except complete evaluation, there are times when the fitness evaluation method will cause a small loss of true fitness. That loss is generally temporary, and a host with equal or greater true fitness is subsequently evolved.

Complete evaluation, however, is notable because it was unable to evolve a single correct solution. While complete evaluation typically shows some improvement in true fitness as the population evolves, there are no large jumps in true fitness of the best individual in the population; rather, the evolutionary process is able to make only small improvements to existing individuals.

5. ANALYSIS

5.1 Comparison of Results

We found three major differences between our results and those previously reported by Pagie and Hogeweg [6]: the success rates of coevolution and random evaluation, and the mean number of nodes in the best evolved hosts.

Pagie and Hogeweg report a success rate for coevolution of 45%, compared with our success rate of 78%. The

difference seems to be due to how the parasites are initialized. In Pagie and Hogeweg's runs, all parasites were initialized to a single central value: (0.2,0.2). In our runs parasites were initialized to randomly chosen problems. Moreover, when we ran coevolution with parasites initialized to (0.2,0.2), we obtained success rates comparable to those reported by Pagie and Hogeweg.

Pagie and Hogeweg report a success rate for random evolution of 35%, whereas we find that random evolution succeeds in only 14% of runs. We also find that the trees that are evolved in all methods are typically much larger than the sizes reported by Pagie and Hogeweg (see Tables 1 and 3 above). In spite of a detailed review of Pagie and Hogeweg's algorithm, with the assistance of Ludo Pagie [personal communication], we have been unable to identify the cause of these differences.

5.2 Analysis of Results

Following Pagie and Mitchell [7], we investigate the following two hypotheses for the differences in success rate of spatial coevolution as compared with the other evolutionary methods investigated here:

- 1) Spatial coevolution promotes continued diversity in the population.
- 2) Spatial coevolution produces parasites that target specific weaknesses in hosts.

Diversity is thought to be important for evolutionary computation because low diversity decreases the chances that a useful crossover or mutation will occur. Here we measure diversity using the entropy of the different phenotypes exhibited in the population:

$$\text{Entropy} = -\sum_{i=0}^{n_{\text{phenotypes}}} p_i \log_2 p_i,$$

where $p_i = \text{phenotype}_i / n_{\text{hosts}}$, phenotype_i is the number of hosts that exhibit a specific phenotype, n_{hosts} is the total number of hosts in the population (here, 2500), and

$n_{\text{phenotypes}}$ is the number of different phenotypes exhibited in the population. A higher entropy value indicates greater phenotype diversity.

Figures 3 through 7 show the phenotype entropy of all 50 runs for each of the fitness evaluation methods. Because the host population is randomly initialized, all runs (whether successful or not) for all evaluation methods start with the same entropy (here, approximately 4.0). Successful runs are plotted in white and unsuccessful runs are plotted in black. Coevolution and host-against-all coevolution both attain high phenotype entropy immediately, and the entropy remains high until a correct solution is discovered, at which time the entropy drops as the correct host dominates the population. At times, one strategy will gain momentum to overtake the population, reducing entropy. However, entropy quickly rises as diversity is reestablished. Only when a correct host is discovered does the host phenotype entropy drop substantially as that host takes over the population.

While the spatial distribution of hosts plays a part in encouraging diversity, these results support our first hypothesis, that spatial coevolution consistently promotes continuing population diversity. Resource sharing also encourages high entropy as shown in Figure 5, although the entropy does not increase as immediately as the coevolutionary methods, nor does it go quite as high. As shown in Figures 6 and 7, some runs do achieve high entropy under complete evaluation and random evaluation. However, the entropy of the runs under these methods varies greatly. In short, the coevolutionary methods exhibit high entropy more reliably than the other methods, but high entropy alone cannot explain their relative success since a higher percentage of runs with high entropy does not directly correlate to a proportionately higher success rate.

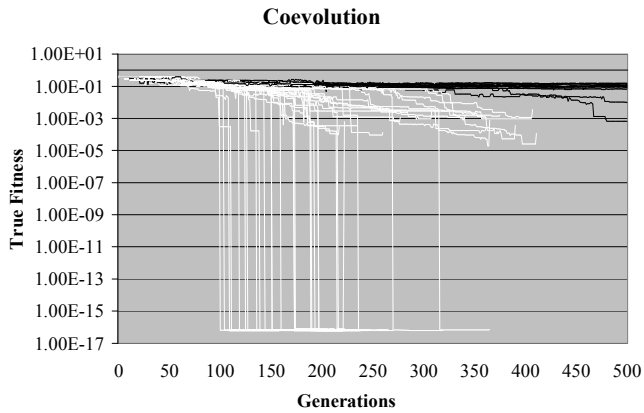


Figure 2: Fitness of the best individual in each generation for all 50 runs of coevolution.

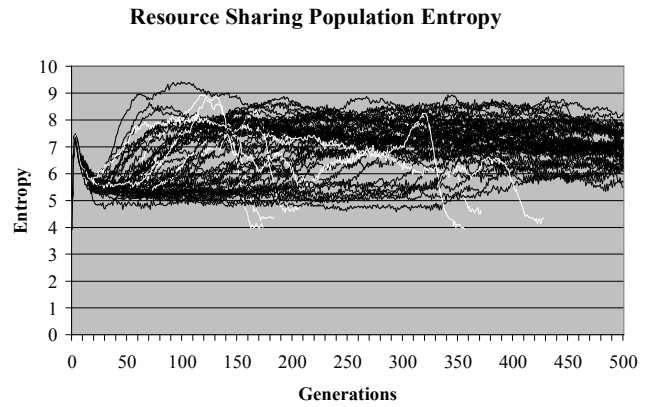


Figure 5: Graph of the entropy of the phenotype of the host population at each generation for all 50 runs of resource sharing.

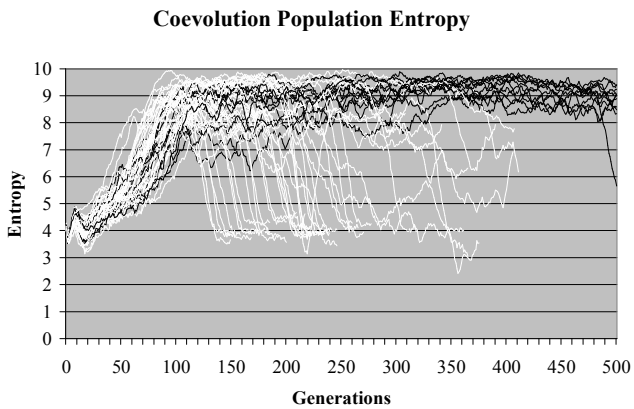


Figure 3: Graph of the entropy of the phenotype of the host population at each generation for all 50 runs of coevolution.

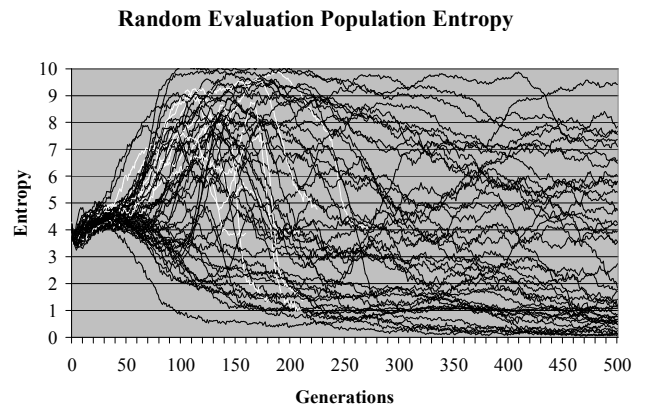


Figure 6: Graph of the entropy of the phenotype of the host population at each generation for all 50 runs of random evaluation.

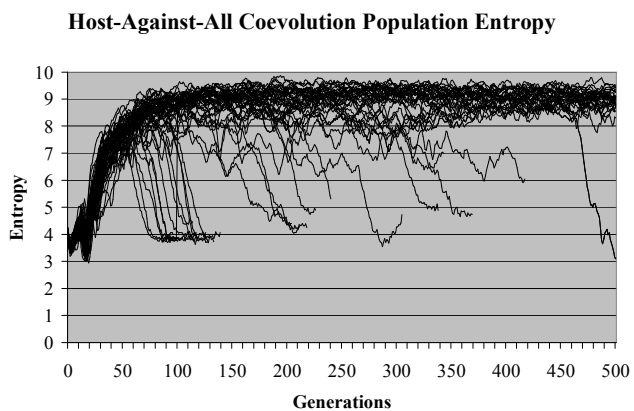


Figure 4: Graph of the entropy of the phenotype of the host population at each generation for all 50 runs of host-against-all coevolution.

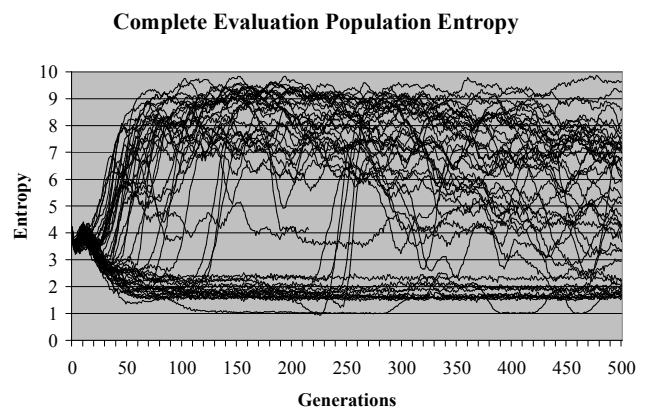


Figure 7: Graph of the entropy of the phenotype of the host population at each generation for all 50 runs of complete evaluation.

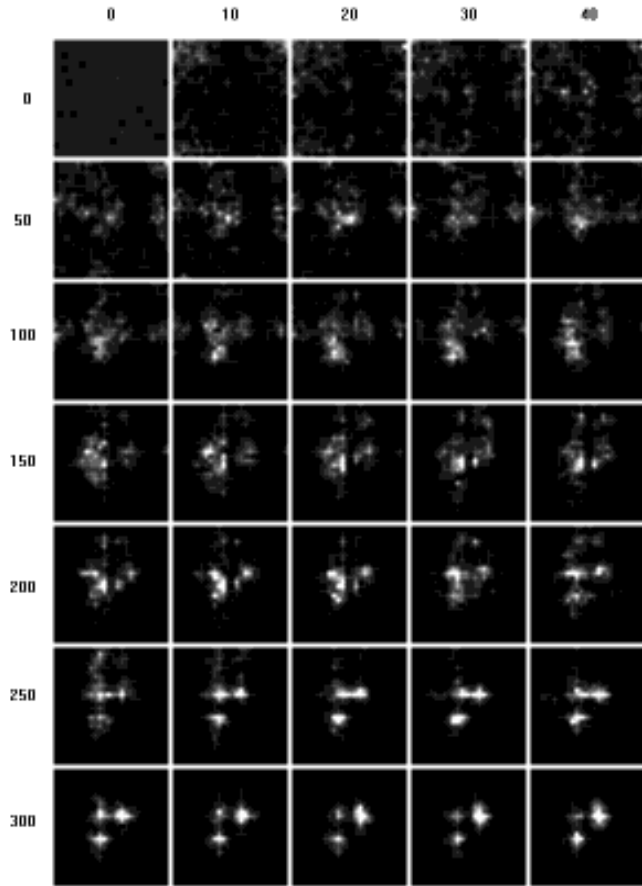


Figure 8: Graph of the concentration of coevolving problems in the problem space for a particular successful run of coevolution. Snapshot of parasites were taken every 10 generations.

Our second hypothesis is that spatial coevolution produces parasites that target specific weaknesses in hosts. Figure 8 shows snapshots, taken every 10 generations, of the concentration of problems that the parasite population uses to challenge the hosts for a typical successful run. Figures 9 and 10 show the true fitness of the best individual and the population phenotype entropy at these generations. In Figure 8, each box is a 26 x 26 grid where the grid coordinates are the x and y values of each problem in the complete set of 676 problems. The brighter the dot at the given coordinate (x, y) , the higher the concentration of problem (x, y) in the parasite population. The problems start out scattered evenly over the problem domain (as shown in the snapshot for generation 0) and evolve to an early focus on the edges of the problem domain. However, they soon start focusing on specific problems spread throughout the domain, with specific focus towards the center and in the corners. The original population of random hosts contains at least one host that solves each different problem, but none that solve all the problems. The initial random population of hosts solves more problems in the ridges of the target function; therefore, the parasites focus on the

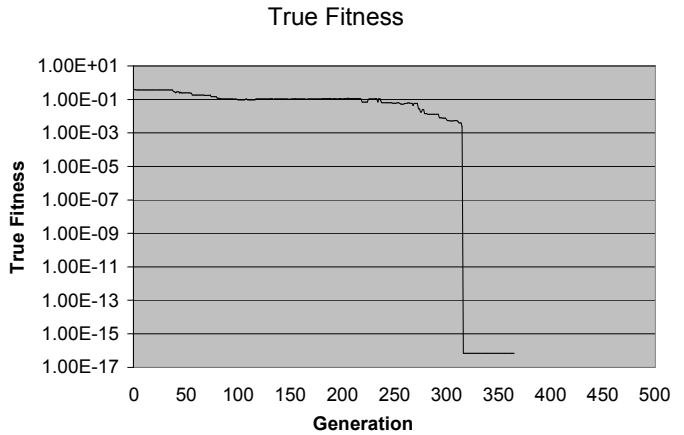


Figure 9: Graph of the true fitness for the particular successful run of coevolution in figure 8.

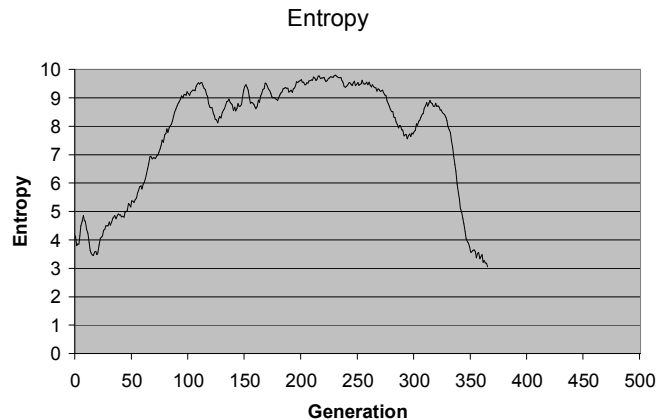


Figure 10: Graph of the entropy for the particular successful run of coevolution in figure 8.

problems in the corner of the problem domain initially. Nevertheless, since the hosts can easily solve those problems using a constant function, the parasites focus on the ridges of the target function in the center of the problem domain.

Eventually, as a successful host is discovered, all problems focus on the center of the problem domain. Similar focusing is exhibited by host-against-all coevolution. The changing focus of the parasite population supports our second hypothesis that the parasites are targeting the weaknesses of the hosts in the population during the different stages of learning.

6. CONCLUSION

In our experiments, coevolution had a substantially higher success rate than all other methods. In contrast with the results of Pagie and Hogeweg, random evaluation, which

uses sparse fitness evaluation but no coevolution, had a much lower success rate. Likewise, maintaining high diversity, by itself, did not indicate a high success rate, as shown by results of resource sharing. Host-against-all coevolution, which was meant as a control to test the effect of co-locating hosts and parasites, had an intermediate success rate.

Our results give support for our two hypotheses: that the success of spatial coevolution is largely due to maintained diversity and the ability of parasites to target specific weaknesses in neighboring hosts. These factors produce hosts that are able to completely solve the evolutionary target by induction from a small number of strategic hard problems. The precise mechanisms by which spatial embedding combined with coevolution produces these results are still not well understood.

In the future, more work needs to be done further analyzing the properties of the evolving spatial populations such as correlating changes in entropy with changes in true fitness. Further diversity measurements and analysis of the growth and spread of emerging strategies will lead to a better understanding of the principles at work. Additionally, comparative studies of other learning methods, such as boosting algorithms, can also lead to a better understanding of the mechanisms underlying the success of coevolution.

7. REFERENCES

- [1] Cartlidge, J. and Bullock, S. (2004). Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation* 12(2): 193-222.
- [2] Hillis, D. W. (1990); Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D* 42:228-234.
- [3] Juillé, H. and Pollack, J. B. (1998). Coevolutionary learning: A case study. *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, Wisconsin, July 24 - 26, 1998, pp 251-259.
- [4] Koza, J. R. (1990): *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*. Technical Report No. STAN-CS-90-314, Computer Sciences Department, Stanford University.
- [5] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.
- [6] Pagie, L. & Hogeweg, P. (1997); Evolutionary consequences of coevolving targets. *Evolutionary Computation* 5(4):401-418.
- [7] Pagie, L. & Mitchell, M. (2002). A comparison of evolutionary and coevolutionary search. *International Journal of Computational Intelligence and Applications*, 2(1), 53--69.
- [8] Rosin, C. D. (1997). *Coevolutionary Search Among Adversaries*. Ph.D. thesis, University of California, San Diego, Department of Computer Science and Engineering.
- [9] Rosin, C. D. & Belew, R. K. (1995). Methods for competitive coevolution: Finding opponents worth beating. In Eshelman, L. J. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 373-381. San Mateo, CA: Morgan Kaufmann.
- [10] Watson, R.A. and Pollack, J.B. (2001). Coevolutionary Dynamics in a Minimal Substrate. In Spector, L. et al. (Eds), *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pp. 702-709 San Mateo, CA: Morgan Kaufmann.
- [11] 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