Resource Sharing and Coevolution in Evolving Cellular Automata

Justin Werfel* Melanie Mitchell† James P. Crutchfield‡

Abstract

Coevolution between a population of candidate solutions and a population of test cases has received increasing attention as a promising biologically inspired method for improving the performance of evolutionary computation techniques. However, the results of studies of coevolution have been mixed. One of the seemingly more impressive results to date was the improvement via coevolution demonstrated by Juillé and Pollack on evolving cellular automata to perform a classification task. Their study, however, like most other studies on coevolution, did not investigate the mechanisms giving rise to the observed improvements. In this paper we probe more deeply into the reasons for these observed improvements and present empirical evidence that, in contrast to what was claimed by Juillé and Pollack, much of the improvement seen was due to their "resource sharing" technique rather than to coevolution. We also present empirical evidence that resource sharing works, at least in part, by preserving diversity in the population.

1 Introduction

Using evolutionary algorithms to design problem-solving strategies often involves the use of *test cases* to estimate fitness, since the space of possible problems is typically too large to evaluate a given strategy's performance exhaustively. An important issue for improving statistical estimates of fitness in such situations is determining how to sample test cases and how to weight their contribution to fitness estimates. This is particularly significant if one wishes to avoid premature convergence, in which a

*Department of Electrical Engineering and Computer Science, E25-210, Massachusetts Institute of Technology, Cambridge, MA 02139 (email: jkwerfel@mit.edu)

†Biophysics, P-21, MS D454, Los Alamos National Laboratory, Los Alamos, NM 87545 (email: mm@biophysics.lanl.gov)

‡Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501 (email: chaos@santafe.edu)

mediocre solution strategy with no nearby fitter variants takes over the population and prevents the emergence of better solutions.

Techniques that have been proposed to ameliorate this difficulty include *shared sampling*, in which test cases are chosen so as to be unsolvable by as many of the strategies in the population as possible [16, 17]; *competitive fitness functions*, in which a tournament-style selection scheme determines that one strategy is fitter than another if the number of test cases solved by the first, but not by the second, is greater than the number solved by the second, but not by the first [8]; and *resource-sharing fitness functions*, in which strategies receive a higher fitness if they are able to solve test cases that are unsolvable by a large fraction of other strategies. Resource sharing has produced promising results on a number of tasks [7, 8, 16, 17].

The motivation behind resource sharing is to promote diversity by rewarding strategies that can solve test cases that few other strategies are also able to solve. In this way strategies receive less payoff for pursuing approaches that put them into "niches" already heavily occupied. Instead, they are encouraged to explore new approaches, particularly those which allow solving test cases that the rest of the population finds difficult. Presumably, the population ends up more spread out over the space of possible strategies. In other words, resource sharing is intended to preserve diversity, to prevent mediocre solutions from taking over the population, and to make more likely the emergence of exceptional new strategies through recombinations of dissimilar, previously discovered strategies.

Another technique that has been proposed to improve the strategies discovered by evolutionary search methods is that of coevolution, as introduced by Hillis [5]. Any particular static method for generating test cases can strongly bias the evolutionary course of strategies and produce over-fitting to the class of test cases that are generated. Moreover, there appears to be no single best static method. If the test cases are too easy, there is no strong pressure for high-performance strategies to emerge; if the test cases are too hard, then all low-performance strategies appear equally poor, reducing fitness variance, and evolution cannot proceed.

In a coevolutionary scheme, a population of test cases is maintained and evolves along with the evolving population of strategies. The fitness of a strategy is then some function of the number of test cases it is able to solve and the fitness of a test case is some inverse function of the number of strategies that are able to solve it, often with some penalty for being too difficult a test. The desired effect is that the test-case population will evolve so as to present an incrementally increasing but appropriate level of difficulty for the evolving population that forces strategies to become successively more capable of solving hard problems.

Past work by Paredis [15], in accord with our own earlier investigations, showed that a straightforward version of coevolution, on its own, fails to produce high-performing strategies for a cellular-automaton task investigated earlier by Packard [14] and Crutchfield et al. [1, 4, 11]. These researchers used genetic algorithms (GAs) to evolve one-dimensional, two-state cellular automata (CAs) to perform a classification task. This type of CA consists of a one-dimensional lattice of cells, each of which can be in state 0 or 1 at a given time step. The lattice starts out with an initial configuration of states, and at each time step each cell updates its state depending on its current state and

the states of its neighboring cells. In Packard's and Crutchfield et al.'s studies, the "neighboring cells" of a cell were defined to be the three cells on either side of the cell. Thus each neighborhood contained seven cells. The update rules can be given as a look-up table ("rule table") containing all possible configurations of seven cells and the associated update state for the center cell in each configuration. Any given rule table can be specified uniquely by ordering the entries in lexicographic order of neighborhood configuration (0000000 to 1111111) and then listing the $2^7 = 128$ update states in this order. This produces a bit string of length 128. These bit strings were the individuals in the GA's population.

Following [14] and [1, 4, 11], Paredis evolved cellular automata to perform a density classification task, in which an initial configuration of the CA lattice consisting of 1s and 0s was to be classified as "high-density" or "low-density" depending on whether or not it contained a majority of 1s. (The density of an initial configuration is defined as the fraction of 1s in that configuration.) A "high-density" classification was represented by the CA reaching a fixed point of all 1s, a "low-density" classification by a fixed point of all 0s. In this task, the "strategies" are CA rule tables and the "test cases" are initial configurations of a CA lattice.

In [15], a population $P_{\rm CA}$ of cellular automaton rule tables, encoded as bit strings, coevolved with a population $P_{\rm test}$ of initial-configuration test cases, also encoded as bit strings. The fitness of a CA rule table was calculated by running the corresponding cellular automaton (with a lattice of 149 cells) on each initial configuration in $P_{\rm test}$, and determining the number of these initial configurations that it classified correctly. The fitness of each test case was the number of CA rule tables in $P_{\rm CA}$ that classified it *incorrectly*. Paredis found that the two populations entered temporal oscillations in which each in turn performed well against the other population. The individuals in both populations, however, performed poorly against opponents chosen from outside the populations.

Combinations of different approaches for improving performance often work better than each approach alone [17]. In particular, Juillé and Pollack [9, 10] investigated a combination of coevolution and resource sharing in the evolving cellular automata framework described above, and found that the use of both techniques together led to the production of significantly better CA strategies than did the use of a standard GA. They attributed this success to the effectiveness of coevolution.

Since the version of coevolution studied by Paredis [15] has been shown not to produce effective strategies for this problem when used alone, it seems natural to ask whether the success in [9, 10] is due more to coevolution or to resource sharing, or to the particular combination of the two.

It should be noted that the results of both Paredis [15] and Juillé and Pollack [9, 10], as well as the new results we present below, were obtained in the context of the evolving cellular automaton framework and have not yet been generalized to other problems. This framework, however, was designed to be general in that it captures the important features of evolving systems in which global coordination emerges when only local interactions are possible. In addition, this framework has been found to have a number of features common to a wide class of evolutionary systems, including moderate-to-high degrees of epistatic interactions among loci, identifiable "building blocks" that

contribute to high-fitness solutions, a demonstratable advantage for crossover versus mutation alone, and metastable periods of fitness stasis punctuated by rapid periods of innovation [2, 12, 19]. Moreover, these features have been shown to generalize to other cellular-automaton tasks beyond one-dimensional density classification [3, 13]. The generality of these features will, we believe, allow the results of research in this framework to inform work on a wider class of evolving systems. Thus we believe the results in [9, 10, 15], as well as our results described below on coevolution and resource sharing, will have implications beyond the evolving cellular automaton framework.

2 Methods

In [1, 4, 11], a genetic algorithm (GA) was used to evolve cellular automaton rule tables (strategies) to perform the density classification task described above. The fitness of each strategy was a function of its classification performance on a random sample of test cases: initial configurations (ICs) of the CA lattice. The classification performance was defined as the fraction of ICs in a training sample or test sample that were correctly classified. The ultimate success of the GA was measured in two ways: (1) The *performances* $\mathcal{P}_N$ of the best evolved strategies—the fraction of correct classifications on N randomly generated test cases. For the results reported here, we used $\mathcal{P}_{10^4}$. (2) The GA's *search efficiency* $\mathcal{E}_s$—the percentage of runs on which at least one instance of a given type of strategy s was evolved.

In [1, 4, 11] we identified three classes of CA computational strategy s evolved by the GA:

- *Default:* The CA always iterates to all 1s or all 0s.
- *Block-expanding:* The CA always iterates to all 0s (1s) unless there is a sufficiently large block of 1s (0s) in the IC, in which case that block grows until it fills the lattice.
- *Particle:* The CA uses localized moving signals—"particles"—and their collisions to transfer and combine information from different parts of the IC.

These classes were identified on the basis of both $\mathcal{P}_N$ and by extensive analysis of space-time patterns produced by the CAs of each type. Only the particle strategies resulted in high performance and generalized well to large lattice sizes; only the particle strategies are examples of what we would want to call sophisticated collective computation emerging from local rules.

The three different strategies were easily distinguished by the performance $\mathcal{P}$ they generated: on 149-cell lattices (the size used in the experiments reported here) the default strategies had $\mathcal{P} = 0.5$, the block-expanding strategies had $\approx 0.6 < \mathcal{P} < 0.68$, and the particle strategies had $\mathcal{P} \geq 0.7$. A small number of *high-performance particle* strategies evolved with $\mathcal{P} \geq 0.8$. As the lattice size was increased, the performance of block-expanding strategies quickly went down to approximately 0.5, whereas the performance of particle strategies declined much more slowly. The space-time behavior of the high-performance particle strategies was qualitatively similar to that of the

lower-performance particle strategies; why the former's performance was higher is still an open question.

In [4], $\mathcal{E}_{\text{particle}}$ was approximately 3%. For reference, we note that to date the best known CAs for density classification, evolved or designed by hand, have performances approximately $0.8 \leq \mathcal{P}_{10^4} \leq 0.86$ on 149-cell lattices.

Juillé and Pollack [9, 10] showed that a particular combined form of resource sharing and coevolution resulted in higher performances (up to $\mathcal{P} \approx 0.86$) and high-performance search efficiencies ($\mathcal{E}_{\text{particle}} > 30\%$) [6] than were found in earlier evolving cellular automata experiments.

For comparison, Paredis's version of coevolution [15] alone produced only low-performance CAs that did no better than default strategies and had search efficiencies $\mathcal{E}_{\text{particle}} = 0\%$—substantially worse than that of a GA without coevolution. To investigate what aspects of Juillé and Pollack's method were responsible for the improved performance and search efficiency, we performed a series of experiments to replicate their results and analyze them more deeply than was reported in [9] and [10].

The experiments described here used GA and CA parameters, resource sharing fitness functions, and a coevolution scheme similar to those of [9, 10], and identical to those of a follow-up study by Juillé [6]. The populations of CAs and ICs each had 200 members. The CAs were tested on 149-cell lattices. We performed four experiments, each consisting of 50 GA runs initiated with independent random number seeds, where each run consisted of 1000 generations. The experiments evaluated four search techniques: (1) GA: the GA alone, with neither resource sharing nor coevolution, with ICs drawn at each generation from a density-uniform distribution (i.e., a probability distribution which is uniform with respect to IC density[1])[2]; (2) GA+C: the GA with coevolution only, with ICs initially drawn at each generation from a density-uniform distribution and allowed to evolve thereafter; (3) GA+RS: the GA with resource sharing only, with ICs drawn at each generation from a density-uniform distribution; and (4) GA+RS+C: the GA with resource sharing and coevolution combined, with ICs initially drawn from a density-uniform distribution and allowed to evolve thereafter.

In the GA without resource sharing, the fitness function for a CA was simply the number of ICs it was able to classify correctly:

$$f(\text{CA}_i) = \sum_{j=1}^{N_{\text{IC}}} \mathit{correct}(\text{CA}_i, \text{IC}_j)$$

where N_{IC} was the number of ICs in the population, and $\mathit{correct}(\text{CA}_k, \text{IC}_j)$ was 1 if the CA correctly classified the jth IC and 0 otherwise.

When coevolution was used without resource sharing, the fitness function for ICs was defined analogously, with the addition of a term $E(\text{CA}_i, \rho(\text{IC}_j))$ to artificially lower

[1]The density-uniform distribution can be contrasted with the binomially distributed densities of ICs generated by choosing each bit randomly, as is done when calculating $\mathcal{P}_N$. The latter produces a distribution of densities strongly peaked around 0.5—the hardest cases to classify. Using the density-uniform distribution to generate ICs for evaluating fitness markedly improved the GA's success in all cases where ICs are chosen from a distribution.

[2]The same algorithm was used in [1, 2, 4] but with smaller values for population size and number of generations, and consequently lower search efficiencies.

the fitnesses of especially difficult ICs with densities near $\rho = 1/2$:

$$E(\mathrm{CA}_i, \rho(\mathrm{IC}_j)) = \ln(2) + p\ln(p) + (1-p)\ln(1-p)$$

where p was the probability that the ith CA could solve a randomly generated IC of density $\rho(\mathrm{IC}_j)$, the density of the jth IC. The IC fitness function was then

$$f(\mathrm{IC}_j) = \sum_{i=1}^{N_{\mathrm{CA}}} E(\mathrm{CA}_i, \rho(\mathrm{IC}_j)) \cdot (1 - \mathit{correct}(\mathrm{CA}_i, \mathrm{IC}_j))$$

When resource sharing was used without coevolution, ICs did not evolve but were generated at each generation from a density-uniform distribution. For the purpose of calculating CA fitnesses, each IC was assigned a weight based on how many CAs it defeated:

$$W_{\mathrm{IC}_j} = \frac{1}{\sum_{k=1}^{N_{\mathrm{CA}}} \mathit{correct}(\mathrm{CA}_k, \mathrm{IC}_j)}$$

Each CA was then assigned a fitness according to which ICs it could solve, based on those weights:

$$f(\mathrm{CA}_i) = \sum_{j=1}^{N_{\mathrm{IC}}} W_{\mathrm{IC}_j} \cdot \mathit{correct}(\mathrm{CA}_i, \mathrm{IC}_j)$$

These definitions provided a limited fitness resource, equal to the total number of ICs in the population, which was divided up among all CAs.

When coevolution was used with resource sharing, CA weights and IC fitnesses were defined analogously:

$$\begin{aligned} W_{\mathrm{CA}_i} &= \frac{1}{\sum_{k=1}^{N_{\mathrm{IC}}} E(\mathrm{CA}_i, \rho(\mathrm{IC}_k)) \cdot (1 - \mathit{correct}(\mathrm{CA}_i, \mathrm{IC}_k))} \\ f(\mathrm{IC}_j) &= \sum_{i=1}^{N_{\mathrm{CA}}} W_{\mathrm{CA}_i} \cdot E(\mathrm{CA}_i, \rho(\mathrm{IC}_j)) \cdot (1 - \mathit{correct}(\mathrm{CA}_i, \mathrm{IC}_j)) \end{aligned}$$

CA_i was considered to have correctly classified IC_j if after a maximum of $2.15 \cdot 149$ successive applications of the CA rule, CA_i had reached a fixed point of all 1s if $\rho(\mathrm{IC}_j) > 0.5$ and a fixed point of all 0s otherwise. (The case $\rho(\mathrm{IC}_j) = 0.5$ was not possible on a lattice of 149 cells.)

The initial population of CAs was drawn from a uniform distribution over the density of the 128-bit update rule (i.e., all densities were equally likely). The elite CAs (fittest 20%) each generation survived to the next; 60% of the new generation was created from single-point crossovers between pairs of randomly chosen elite individuals, with mutation probability 0.02 per bit; 20% was created by copying single elite chosen randomly (with replacement) with mutation probability 0.02 per bit. Mutation flipped the bit at the chosen locus.

With coevolution, the evolving IC population was represented as a set of densities, rather than specific ICs; each generation, a new set of ICs was generated with the specified densities. At each generation 97% of the IC population survived intact, with

the remaining 3% chosen from a density-uniform distribution. As in Juillé and Pollack's experiments, no crossover or mutation was applied to the IC population.

As described above, the performance $\mathcal{P}_{10^4}$ of a CA, evaluated after a run, was defined as the fraction it classified correctly of 10000 ICs drawn at random from an unbiased distribution (i.e., each cell in each IC had an equal probability of being 0 or 1).

3 Results

For each experiment, we recorded the number of runs in which block-expanding, particle, and high-performance particle strategies ("particle+") were discovered and the mean number of generations it took to discover each strategy.

The search efficiency $\mathcal{E}_s$ and the mean generation of first occurrence t_s for each strategy are given in Table 1. The standard deviation σ_{t_s} of t_s across the 50 runs of each alternative GA is also reported there.

The results for GA+RS and GA+RS+C agree, within statistical uncertainty, with results found by Juillé [6].

As expected from [15], runs with coevolution alone almost never produced particle strategies ($\mathcal{P}_{10^4} > 0.7$). In addition, the use of coevolution increases the average time taken by the GA to find even low-performance, block-expanding strategies (e.g., $t_s = 104$, rather than 14), and likewise increases the variance in that time ($\sigma_{t_s} = 133$, rather than 6).

Runs with resource sharing produce CAs with high performance more consistently across runs than does the GA alone ($\mathcal{E}_{\text{particle}} = 43\%$, rather than 29%). Moreover, runs with resource sharing tend to take longer ($t_s = 33$, rather than 14) to find block-expanding CAs. They also vary more ($\sigma_{t_s} = 63$, rather than 6) in how long they take to do so.

Comparing runs using both resource sharing and coevolution to those using resource sharing alone, the addition of coevolution appears to heighten these effects of resource sharing. Runs using both techniques take longer ($t_s \approx 84$, rather than 33) to find block-expanding CAs, vary more in how long it takes them to do so ($\sigma_{t_s} \approx 157$ rather than 63), and find particle CAs more frequently (e.g., $\mathcal{E}_{\text{particle}} = 47\%$ rather than 43% and $\mathcal{E}_{\text{particle+}} = 27\%$ rather than 10%) than do runs with resource sharing alone.

By contrast, comparing runs with resource sharing and coevolution to those with coevolution alone and to those with neither, we see that coevolution has entirely different effects in the presence or absence of resource sharing. Coevolution alone greatly decreases the effectiveness of the basic GA in discovering high-performance CA rules, while if resource sharing is also present, the success of the GA in discovering high-performance rules is improved considerably.

As an aside, note that the large variances σ_{t_s} in mean time to find a given strategy are typical of and to be expected in evolutionary search algorithms. The nature of such fluctuations is discussed in [18]. What is notable here is that for the discovery of particle strategies, the GA using resource sharing has much less variation than seen in the GA alone. The addition of coevolution to resource sharing appears to have little

(beneficial) effect in reducing the variations for the appearance times of particle CAs. In fact, in reaching high-performance particle CAs, the addition of coevolution roughly doubles the variance in t_s.

In short, for increased efficiency in finding particle strategies, the differences between resource sharing with coevolution (GA+RS+C) and resource sharing alone (GA+RS) are much less pronounced than the corresponding differences between GA+RS and GA+C, and between GA+RS and GA. What the addition of coevolution does most clearly is improve the search efficiency for high-performance particle strategies, and that only in the presence of resource sharing. Thus, while Juillé and Pollack [9, 10] attributed all the improvements they observed to the addition of coevolution, our results make it clear that resource sharing plays a major, if not *the* major role in producing these improvements.

4 The Operation of Resource Sharing

We also investigated whether the effectiveness of resource sharing is actually due, as was intended in its design, to a preservation of diversity in the GA population, or whether its success results from some other mechanism entirely.

One rough measure of diversity in a population is the average pairwise Hamming distance $\langle d \rangle$. The Hamming distance d between two CAs is simply the number of bits by which the genetic specification of their rule tables differ. CAs with different strategies are likely to differ in more bits and thus to be separated by a greater Hamming distance than CAs with similar strategies. When averaged over the population, $\langle d \rangle$ is greater if a population is more strategically diverse overall and its members are more spread out across the genotype space.[3]

To give a sense of the scale of Hamming distances here, recall that a CA's rule table is specified by $2^7 = 128$ binary update states. Thus, $0 \leq \langle d \rangle \leq 128$. Since the initial CA population was randomly initialized, $\langle d \rangle \approx 128/2 = 64$ bits at the start of a run.

Figure 1 shows $\langle d \rangle$ at each generation for sample runs that evolved high-fitness particle-based CAs, with (a) neither resource sharing nor coevolution, (b) coevolution alone, (c) resource sharing alone, and (d) both techniques. In all cases, as expected, $\langle d \rangle$ starts out at approximately 64 and then quickly decreases over a few generations, as the fittest CAs and their descendants take over the population, which settles down to CAs with similar strategies.

Beyond this transient phase, over each run $\langle d \rangle$ fluctuates about 10% to 20% as evolution progresses. Nonetheless, as the plots show, each run does follow an overall trend in population diversity. We measured these trends using a least-squares fit to estimate the average rate of change in population diversity, $\langle \dot{d} \rangle$. We also estimated the standard deviation $\sigma_{\langle d \rangle}$ of fluctuations in $\langle d \rangle$ about the fit. Both estimates for each run are reported in Table 1. Since such trends are interrupted as the GA discovers

[3]Concerned about possible long-tailed distributions governing d, we calculated the median, in addition to the average, pairwise Hamming distances. There was no qualitative change to the results. Moreover, the median distance never differed from the average by more than a single bit after the first few generations. For these reasons, we report here only average pairwise Hamming distances.

progressively more effective CAs, the fits were made only over a stationary epoch—a period in which average population fitness remains roughly constant. For Fig. 1(a), the fit is made for an epoch lasting from generation 50 to generation 800; in Fig. 1(b), from generation 100 to 1000; in Fig. 1(c), from 200 to 1000; and in Fig. 1(d), from 100 to 750.

In runs with neither resource sharing nor coevolution, $\langle d \rangle$ decreases slowly over time, with temporary increases each time a new, more effective type of strategy is discovered. In Figure 1(a), for example, $\langle d \rangle$ declines slowly over nearly 800 generations, from close to 18 bits to a minimum of approximately 12 bits. At that point, a new strategy appears around generation 900 and $\langle d \rangle$ increases again. The estimated trend shows a negative slope, and one concludes that with this algorithm, diversity steadily decreases during an epoch.

When the GA with coevolution is used, as in Fig. 1(b), $\langle d \rangle$ remains roughly constant over a period of several hundred generations. However, its value here is only about 10 bits, implying that in this rare case where coevolution alone does manage to produce high-performance CA rules, the diversity of the CA population is considerably lower than in any of the other versions of the GA. This fact suggests that part of the reason coevolution by itself finds high-performance rules so infrequently may be because its search through the space of possible CAs is relatively narrow.

When the GA with resource sharing is used, as illustrated by the run in Fig. 1(c), $\langle d \rangle$ remains roughly constant at approximately 17 bits. There may also be slightly greater fluctuation in the population diversity about this trend than in the alternative GAs; see $\sigma_{\langle d \rangle}$ in Table 1.

When coevolution and resource sharing are both used, as shown by the run in Fig. 1(d), $\langle d \rangle$ increases over time, after the population initially settles down. Here, $\langle d \rangle$ goes from about 15 bits to about 21 bits over the course of 700 generations.

It would appear then that resource sharing maintains diversity, as it was intended to do. Its use prevents the slow decrease in, and lower values of, total Hamming distance that otherwise occur as the population converges on a narrower range of strategies. In other words, it maintains a wider variation in the space of CAs. The addition of coevolution appears to enhance the effect of resource sharing when the latter is also used: the total Hamming distance increases and reaches markedly higher values over a similar number of generations.

5 Conclusions and Further Work

We have presented evidence that when a combination of resource sharing and coevolution improves GA performance, the improvement is largely (though not wholly) due to resource sharing rather than to coevolution on its own. This contradicts the conclusion offered in [9, 10] that coevolution was the driving force behind improved GA performance. We have also presented evidence that resource sharing works by preserving diversity in the population.

Like [9, 10] and other published work on coevolution and resource sharing, our analysis was confined to a single example, here evolving cellular automata to perform

density classification. We do not know if the results will generalize to other evolutionary computation applications, but we believe that they will for the reasons given in the introduction. We hope that our results will spur other researchers to examine carefully the mechanisms by which claimed improvements in evolutionary computation methods occur, especially when more than one improvement mechanism is being used in combination. Coevolution has been widely proposed as a promising mechanism for improvement, and we particularly want to understand how, and in what cases, it can lead to better performance. The work described here is a step in that long-term project.

Underlying these overall questions is a complicated problem in nonlinear population dynamics. Like many other examples in the evolutionary computation literature, high-performance CAs evolve via a series of epochs of stasis punctuated by sudden innovations [1, 2, 4], whether resource sharing, coevolution, both, or neither are employed. The dynamics of epochal evolution has recently been analyzed mathematically in some detail [18, 19, 20]. It would be useful, therefore, to bring the current investigations together with this mathematical analysis to understand why epochal evolution with resource sharing affects, as it does, the variance in the time it takes moderate- and high-performance individuals to emerge in the population, why higher-performance individuals appear more frequently with resource sharing, and how it is that coevolution increases these effects. Here we have begun to understand more systematically how resource sharing and coevolution affect the evolutionary process, but not yet the details of their underlying mechanisms.

Acknowledgments

We thank Rajarshi Das and Wim Hordijk for their help in performing these experiments and Hugues Juillé for helpful discussions. We thank Wim Hordijk for comments on the manuscript. This work was supported by the Santa Fe Institute, by National Science Foundation grants PHY-9531317 (Research Experiences for Undergraduates) and IIS-9705830, and by Keck Foundation grant 98-1677.

References

[1] J. P. Crutchfield and M. Mitchell. The evolution of emergent computation. *Proceedings of the National Academy of Science U.S.A.*, 92:10742–10746, 1995. Available at http://www.santafe.edu/projects/evca/evabstracts.html#evec.

[2] J. P. Crutchfield, M. Mitchell, and R. Das. The evolutionary design of collective computation in cellular automata. Technical Report 98-09-080, Santa Fe Institute, Santa Fe, NM, 1998. Available at http://www.santafe.edu/projects/evca/evabstracts.html#EvDens.

[3] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*,

pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann. Available at http://www.santafe.edu/projects/evca/evabstracts.html#EGSCA.

[4] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature—PPSN III*, volume 866, pages 344–353, Berlin, 1994. Springer-Verlag (Lecture Notes in Computer Science). Available at http://www.santafe.edu/projects/evca/evabstracts.html#particle.

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

[6] H. Juillé. Personal communication, 1998.

[7] H. Juillé and J. B. Pollack. Dynamics of co-evolutionary learning. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, 1996. MIT Press.

[8] H. Juillé and J. B. Pollack. Semantic niching and coevolution in optimization problems. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, Cambridge, MA, 1997. MIT Press.

[9] H. Juillé and J. B. Pollack. Coevolutionary learning: A case study. In *ICML '98—Proceedings for the International Conference on Machine Learning*, San Francisco, CA, 1998. Morgan Kaufmann.

[10] H. Juillé and J. B. Pollack. Coevolving the 'ideal' trainer: Application to the discovery of cellular automata rules. In J. R. Koza, W. Banzhaf, K. Chellapilla, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, San Francisco, CA, 1998. Morgan Kaufmann.

[11] M. Mitchell, J. P. Crutchfield, and R. Das. 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, 1996. Russian Academy of Sciences. Available at http://www.santafe.edu/projects/evca/evabstracts.html#evca-review.

[12] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994. Available at http://www.santafe.edu/projects/evca/evabstracts.html#compCA.

[13] F. Jimenez Morales, J. P. Crutchfield, and M. Mitchell. Evolving two-dimensional cellular automata to perform density classification: A report on work in progress. *Parallel Computing*, 2000. In Press.

[14] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293–301, Singapore, 1988. World Scientific.

[15] J. Paredis. Coevolving cellular automata: Be aware of the red queen! In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 393–400, San Francisco, CA, 1997. Morgan Kaufmann.

[16] C. D. Rosin and R. K. Belew. Methods for competitive coevolution: Finding opponents worth beating. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Francisco, CA, 1995. Morgan Kaufmann.

[17] C. D. Rosin and R. K. Belew. New methods for competitive evolution. *Evolutionary Computation*, 5(1), 1997.

[18] E. van Nimwegen and J. P. Crutchfield. Optimizing epochal evolutionary search: Population-size dependent theory. *Machine Learning*, In press, 1999. Available at http://www.santafe.edu/projects/evca/evabstracts.html#oeespsdt.

[19] E. van Nimwegen, J. P. Crutchfield, and M. Mitchell. Finite populations induce metastability in evolutionary search. *Phys. Lett. A*, 229:3:144–150, 1997. Available at http://www.santafe.edu/projects/evca/evabstracts.html#fpimies.

[20] E. van Nimwegen, J. P. Crutchfield, and M. Mitchell. Statistical dynamics of the Royal Road genetic algorithm. *Theoret. Comp. Sci.*, 229:41–102, 1999. Available at http://www.santafe.edu/projects/evca/evabstracts.html#sdrrga.

Search Technique	CA Strategy s	Search Efficiency $\mathcal{E}_s$	Time to s		Diversity	
			t_s	σ_{t_s}	$\langle \dot{d} \rangle$	$\sigma_{\langle d \rangle}$
GA	block-expanding	100%	14	6	-4.7	2.30
	particle	29%	397	380		
	particle+	6%	488	442		
GA+C	block-expanding	100%	104	133	0.24	2.39
	particle	4%	90	96		
	particle+	0%	N/A	N/A		
GA+RS	block-expanding	100%	33	63	1.3	2.50
	particle	43%	316	287		
	particle+	10%	479	127		
GA+RS+C	block-expanding	100%	84	157	7.7	1.90
	particle	47%	289	266		
	particle+	27%	438	235		

Table 1: Statistics for the evolutionary emergence of CAs with different strategies: block-expanding (here defined as $0.65 < \mathcal{P} < 0.7$), particle ($\mathcal{P} \geq 0.7$), and particle+ ($\mathcal{P} \geq 0.8$). The four main rows give results for four experiments of 50 runs each: "GA" refers to the GA alone, "GA+C" refers to the GA with coevolution only, "GA+RS" refers to the GA with resource sharing only, and "GA+RS+C" refers to the GA with resource sharing and coevolution. Search efficiency $\mathcal{E}_s$ is given for each CA strategy s over the 50 runs of each experiment. t_s is the mean number of generations to first occurrence of strategy s across the 50 runs. σ_{t_s} is the standard deviation in t_s measured across the runs. $\langle \dot{d} \rangle$ is the rate of change in population diversity $\langle d \rangle$ (quoted in bits per 1000 generations), and $\sigma_{\langle d \rangle}$ is the standard deviation of the fluctuation in $\langle d \rangle$ about the best-fit line estimated in the least-squares fits of Figs. 1(a)–(d).

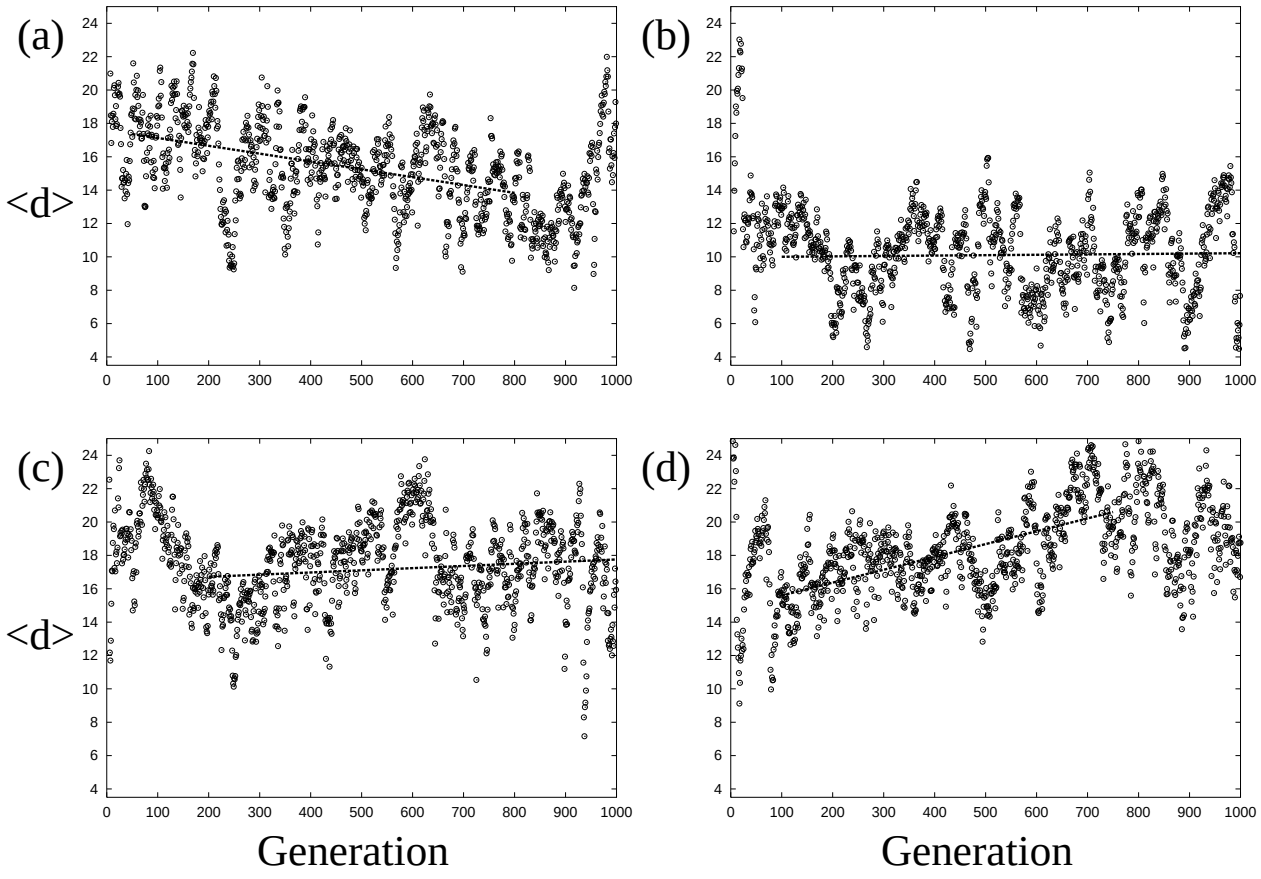

Figure 1: Average pairwise Hamming distance $\langle d \rangle$ over time for single GA runs with (a) neither resource sharing nor coevolution, (b) coevolution alone, (c) resource sharing alone, (d) resource sharing and coevolution. $\langle d \rangle$ is large (≈ 64) during the initial generations, and so these data points do not appear on the scales plotted. The straight lines show the trends in population diversity. They are least-squares fits over stationary fitness epochs in the population dynamics. The estimated slopes $\langle \dot{d} \rangle$ of the lines and the standard deviations $\sigma_{\langle d \rangle}$ of fluctuations about them are quoted in Table 1. The runs shown here are typical of those that evolved particle strategies under each alternative GA.