A comparison of evolutionary and coevolutionary search

Ludo Pagie
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM87501, US
ludo@santafe.edu

Melanie Mitchell
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM87501, US
mm@santafe.edu

Abstract

We present a comparative study of an evolutionary and a coevolutionary search model. In the latter, strategies for solving a problem coevolve with training cases. We find that the coevolutionary model has a relatively large efficacy: 41 out of 50 (82%) of the simulations produce high quality strategies. In contrast, the evolutionary model has a very low efficacy: 1 out of 50 runs (2%) produce high quality strategies. We show that the increased efficacy in the coevolutionary model results from the direct exploitation of low-quality strategies by the population of training cases. We also present evidence that the generality of the high-quality strategies can suffer as a result of this same exploitation.

1 INTRODUCTION

In coevolutionary search the fitness of evolving solutions depends on the state of other, coevolving individuals rather than a fixed evaluation function. The reasons typically cited for the expected increased success and efficiency of coevolutionary search algorithms are the gain in efficiency of the evaluation of evolving solutions (Hillis, 1990), the possible automatic adjustment of the selection gradient which is imposed on the evolving solutions (Juillé & Pollack, 2000), and the potential open-ended nature of coevolutionary systems (Rosin & Belew, 1997; Ficici & Pollack, 1998).

Some successful applications of coevolution have used spatially embedded systems (Hillis, 1990; Husbands, 1994; Pagie & Hogeweg, 1997). In such systems the individuals are distributed on a regular grid and evolutionary processes take place locally among neighboring individuals. Hillis (1990) and Pagie & Hogeweg (1997) gave examples of spatially embedded coevolutionary models that were more successful than spatially embedded, but otherwise standard, evolutionary models for the same search problem. Other successful applications of coevolution have used techniques to produce and maintain diverse populations or to retain good solutions in the populations (e.g., Rosin & Belew, 1997; Paredis, 1997).

Coevolutionary search is not always successful. Unsuccessful cases are often characterized by *Red Queen dynamics* (Paredis, 1997; Shapiro, 1998; Pagie & Hogeweg, 2000; Juillé & Pollack, 2000), *speciation* (Pagie, 1999), or *mediocre stable states* (Pollack *et al.*, 1997; Juillé & Pollack, 2000). Pagie & Hogeweg (2000) showed how Red Queen dynamics can occur in a spatially embedded coevolutionary model under continuous mixing*, while high quality solutions evolve in the same model if mixing is omitted so that spatial patterns can form. In the latter case speciation events can result in persisting sub-species, whereas in the former case global competitive exclusion occurs on a much shorter time-scale and tends to converge the population to one species. Heterogeneity in the populations prevents Red Queen dynamics from persisting. Instead, evolution proceeds to produce general, high quality solutions (Pagie & Hogeweg, 2000), or results in stable situations of diverse populations in which the individuals exhibit sub-optimal behavior (Pagie, 1999).

Here we report results from a follow-up comparative study of a coevolutionary and a standard evolutionary search process, both of which are embedded in space. The most significant result of the study is the difference in efficacy of the two models on a given task. Whereas the coevolutionary model finds solutions that use high quality strategies in 41 out of 50 runs (82%) the evolutionary model finds such a solution only once in 50 runs (2%). By comparing the evolutionary dynamics in the two models we try to characterize the processes that lead to this large difference in search efficacy.

*Continuous mixing in a spatial model approximates a non-spatial model in which selection and recombination take place without regard for spatial location of individuals.

2 CA DENSITY CLASSIFICATION TASK

The task given to both models is the density classification task for cellular automata (CAs) (Packard, 1988; Mitchell *et al.*, 1996). A CA consists of a one-dimensional array (lattice) of cells with periodic (circular) boundary conditions, along with the update rule, that specifies for all possible cell neighborhoods of a given size what the update state should be of the center cell in the neighborhood at the next time step. The initial configuration (IC) of the CA is the state of all cells in the lattice at $t = 0$. The density classification task is defined for one-dimensional, binary state CAs with a neighborhood size of 7; that is, at each time step a cell's new state is determined by its current state and the states of its three neighbors on each side.

The task for a CA is to classify bit strings on the basis of density, defined as the fraction of 1s in the string. If the bit string has a majority of 0s it belongs to density class 0; otherwise it belongs to density class 1. Here we restrict our study to bit strings of length 149, which means that the majority is always defined. (Note that the performance of a CA on this task generally degrades with increasing length of the bit string.) A CA's lattice consists of 149 cells and starts with the given bit string (the one to be classified) as the CA's initial configuration (IC). If, after a maximum of 320 iterations (slightly more than twice the lattice size), the CA settles into a fixed-point homogeneous state of all 0s it classifies the bit string as density class 0. If the CA settles into a fixed-point homogeneous state of all 1s it classifies the bit string as density class 1. If the CA does not settle into one of these states by 320 iterations it makes by definition a mis-classification. Land & Belew (1995) showed that no CA, as defined above, exists that can correctly classify all possible bit strings, but did not give an upper bound on possible classification accuracy.

An objective measure of the classification accuracy of a CA is its *performance* value P, defined as the fraction of correct classifications of 10^4 bit strings, each selected from an "unbiased" distribution—a binomial density distribution centered around density 0.5. Another characteristic of a CA is the type of strategy that it uses in order to classify a bit string. The three main strategies discussed by Mitchell *et al.* (1996) are default, block-expanding, and particle. The default strategy classifies all bit strings to one and the same density class. Thus there are two types of default strategies: default-0 and default-1. Default strategies have performance values of approximately 0.5. Block-expanding strategies are refinements of the default strategy: most bit strings are classified to one density class, for instance density class 0. However, if the bit string contains a sufficiently large block of, in this case, 1s, then this block is expanded to cover the whole string; this results in classification to density class 1. The definition of "sufficiently large" block depends on the particular CA, but typically means blocks equal to or larger than the CA neighborhood size of 7 cells. This strategy reflects the fact that the presence of a block of 1s (0s) in a 149-bit string roughly correlates with overall high (low) density, and the string is classified accordingly. Block-expanding strategies typically have performance values between 0.55 and 0.72 on bit strings of length 149. Finally, particle strategies use interactions among meso-scale patterns in order to classify bit strings (Mitchell *et al.*, 1996). They typically have performance values of 0.72 and higher on bit strings of length 149. In evolutionary search runs one typically sees a series of epochs in which default strategies are evolved first, then block-expanding strategies, and finally (on some runs) particle strategies (Mitchell *et al.*, 1996). The highest observed performance of a particle strategy to date is 0.86 on bit strings of length 149 (Juillé & Pollack, 2000). The strategy of a given CA is determined by either manual observation of its space-time behavior or by measuring its performance P, which correlates very well with strategy.

A number of research groups have used evolutionary search to find strategies for the CA density classification task as a paradigm for the evolution of collective computation in locally interacting dynamical systems (e.g., Mitchell *et al.*, 1996; Juillé & Pollack, 2000). In a standard evolutionary setup, the GA evolves a population of CAs, each of which is encoded as a bit string representing its update rule (see Mitchell *et al.*, 1996 for details). The fitness of a CA is the fraction of correct classifications it makes on a small random sample of bit strings interpreted as ICs. In a coevolutionary setup the bit strings, or ICs, make up the second population and the fitness evaluation of the CAs is based on the evolving ICs. Coevolutionary models which focus on this task were previously studied by Paredis (1997), Juillé & Pollack (2000), and Pagie & Hogeweg (2000).

It appears to be difficult to evolve high-performance CAs; many studies have shown that only a small number of runs produce CAs that use particle strategies. If we define the *efficacy* of an evolutionary search model as the percent of runs of that model that produce particle strategies, then the efficacy of standard evolutionary and previous coevolutionary search models has been found to fall somewhere between 0% and 40% (Mitchell *et al.*, 1996; Werfel *et al.*, 2000; Juillé & Pollack, 2000; Oliveira *et al.*, 2000), with the remaining runs producing block-expanding CAs. Two studies report very high efficacy (Andre *et al.*, 1996; Juillé & Pollack, 2000) but in these cases the computational resources used per simulation were orders of magnitude higher than used here or in the other studies.

Coevolutionary models applied to the CA density classification task generally show Red Queen dynamics of the CAs and ICs (Paredis, 1997; Pagie & Hogeweg, 2000; Juillé &

Pollack, 2000). For the density classification task this type of evolutionary dynamics is characterized by oscillations in the types of ICs and CAs that are present in the population. The ICs oscillate between bit strings of density class 0 and bit strings of density class 1. The CAs oscillate between default-0 strategies and default-1 strategies. Paredis (1997) circumvented Red Queen dynamics by replacing the coevolution of the ICs by randomly generated ICs. Juillé & Pollack (2000) changed the coevolutionary setup by introducing a limit on the selection of ICs such that it was constrained by the ability of the CAs. Pagie & Hogeweg (2000) embedded the coevolutionary model in a spatial grid and introduced an extension on the fitness function used to evaluate the ICs that imposed stabilizing selection on the ICs toward strings that can be classified more easily by CAs. Here we use this same IC fitness function, described below.

3 THE MODEL

The populations in both the evolutionary and coevolutionary models are embedded in a two-dimensional regular grid of 30 by 30 sites with periodic boundary conditions. At each location in this grid one CA and one IC are present, giving 900 individuals in each population. In the coevolutionary model both CAs and ICs evolve whereas in the evolutionary model only the CAs evolve while the ICs are generated anew every generation according to a fixed procedure (see below).

The fitness of a CA is the fraction of its correct classifications on the nine ICs in its Moore neighborhood (i.e., the same site plus the eight direct neighbors). The fitness of an IC is based only on the CA in the same site: $\Phi(IC_i) = 0$ if CA_i classifies it correctly; otherwise $\Phi(IC_i) = |density(IC_i) - \frac{1}{2}|$. This asymmetric fitness evaluation was found to improve the evolutionary search process (Pagie & Hogeweg, 1997) and the dependence on density for the IC fitness function imposed stabilizing selection on the ICs toward strings that can be classified more easily by CAs (Pagie & Hogeweg, 2000). In both models the fitness evaluation is extremely sparse. Sparse evaluation is in fact unavoidable since a complete evaluation of a CA would cover all 2^{149} possible ICs—clearly an infeasible task. Pagie & Hogeweg (1997) showed that sparse fitness evaluation can in fact help the evolutionary process rather than hinder it (see also Hillis, 1990).

Selection of CAs in both models is based on tournament selection. Each CA in the population is replaced by the winner of the tournament which includes the CA itself plus its eight neighboring CAs. The winner is selected, probabilistically, based on the rank order of the individuals in the tournament. The probability of an individual to be selected is 0.5^{rank}, for the eight highest ranked individuals. The lowest ranked individual (i.e., *rank*=9) also has a probability 0.5^8 of being selected, in order to make the probabilities add up to 1. In the coevolutionary model the same selection procedure is applied to the ICs.

After selection we apply point mutations to the CAs and, in the coevolutionary model, to the ICs. The mutation probabilities are 0.0016 per bit for the CAs and, in the coevolutionary case, 0.0034 per bit for the ICs[†]. For the sake of simplicity in this study, crossover is not used in either model.

In the evolutionary model the population of ICs is generated anew at every generation, selected at random from a "uniform" density distribution in which each density in [0,1] is equally likely (c.f. Werfel *et al.*, 2000). Thus, the only difference between the two models is the origin of the new generations of ICs: based on the previous generation in the coevolutionary model and based on a fixed procedure in the evolutionary model.

Each CA is encoded as a 128-bit string representing its update rule (see Mitchell *et al.*, 1996 for details). In both models, the CA population is initialized as random bit strings with an unbiased (binomial) density distribution. In the coevolutionary model the population of ICs is initialized as all-0 bit strings. This initialization of the ICs strongly induces a transient phase of Red Queen dynamics which, in a spatially embedded model, is unstable (Pagie & Hogeweg, 2000). Each simulation is run for 5000 generations. Define an *IC evaluation* as the classification by a CA of a single IC. Then in each model, at each generation $900 \times 9 = 8100$ IC evaluations are performed, i.e., 4.05×10^7 IC evaluations per simulation. This compares to a total number of IC evaluations per simulation in other studies as: 2.61×10^9 in Andre *et al.* (1996), 10^6 in Oliveira *et al.* (2000), 4.0×10^7 in Werfel *et al.* (2000), and 5.0×10^9 and 4.0×10^6 in Juillé & Pollack (2000). For each model we ran 50 simulations each starting with a different random seed.

4 RESULTS

4.1 Efficacy

Table 1 lists the efficacy for each model and each classification strategy, i.e., the number of runs out of 50 on which the given strategy was produced. All three models produced both default and block-expanding strategies on every run. Forty-one coevolutionary runs produced particle CAs, whereas only one evolutionary run did.

[†]The mutation probabilities that we used used correspond to an expected number of mutation events of $0.2 \times$ population size in the CA population and $0.5 \times$ population size in the IC population.

model	default	block	particle
coevolution	50	50	41
evolution	50	50	1
modified coevolution	50	50	12

Table 1: Number of runs (out of 50 total runs) in which each of the three strategies is found in the coevolutionary model, the evolutionary model, and in a modified version of the coevolutionary model (see sect. 4.4).

Figure 1 plots, for each of the 50 runs, the maximum performance P obtained by a CA in that run, for both the coevolutionary model (solid line, top) and evolutionary model (dashed line, bottom). The values are ordered according to decreasing performance values. The symbols denote which strategy is used by the CAs: block-expanding ($\bigtriangledown$) or particle ($\bigtriangleup$).

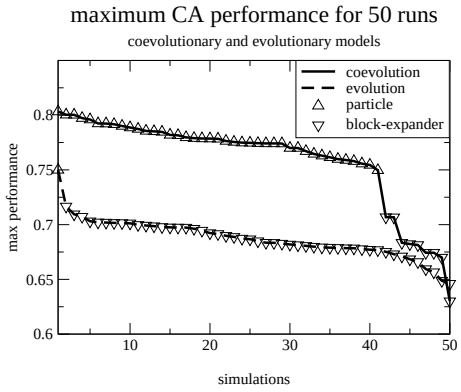

Figure 1: The maximum CA performance P found for each of the 50 runs in the coevolutionary model (solid line) and the evolutionary model (dashed line). The coevolutionary model produces high performance classification strategies (i.e., particle; $\bigtriangleup$) more often than the evolutionary model which produces block-expanding strategies ($\bigtriangledown$) more often.

The difference in efficacy between the coevolutionary and the evolutionary models is very striking. The evolutionary model is unable in almost all cases to make the transition from the block-expanding strategy to the particle strategy, whereas the coevolutionary model is able to make this transition in most runs. We have identified a mechanism that takes place in the coevolutionary model (and not in the evolutionary model) that we believe underlies its increased efficacy: the evolution of bit patterns that target block-expanding CAs. (There are likely to be additional mechanisms that we have not yet identified.) We describe this mechanism and some of its side effects in the next subsections.

4.2 Coevolution of "deceptive" blocks

As was discussed in section 2, the block-expanding strategy operates by expanding sufficiently large blocks of 1s (or 0s) in the IC to produce an all-1 (all-0) state. This strategy can produce fairly high *fitness* when the fitness of a CA is evaluated using random ICs drawn from a uniform distribution, as is done in the evolutionary model, but produces much lower performance P, which is evaluated using random ICs drawn from the unbiased (binomial) distribution.

In the coevolutionary model, block-expanding strategies can easily be exploited by coevolving ICs, and this process of exploitation is very important for producing the high efficacy seen in the coevolutionary model. An IC with a low (high) density can incorporate a block of 1s (0s) which deceptively indicates a high (low) density to a block-expanding CA and thereby force mis-classification. We call such blocks *deceptive blocks*.

For the purpose of this investigation we define the occurrence of a "deceptive block" in an IC to be the occurrence of a block of seven or more adjacent 0s or 1s in the 149-bit-string but only when the probability of the occurrence of such a block in a random string with the same density is less than 0.01. The top panel of fig. 2 illustrates the occurrence and dynamics of deceptive blocks in the IC population in a run of the coevolutionary model. We plot, at every 10 generations, the proportion of ICs in the population that contain deceptive blocks as defined above. We have marked the periods during the simulation when the best CAs use default strategies, block-expanding strategies, and particle strategies. Clearly, in the period when the CAs use block-expanding strategies the number of ICs that contain deceptive blocks is significantly higher than would be expected in random bit strings of the same density. These deceptive blocks—a result of coevolution that directly targets the block-expanding CAs in the population—push the CA population to discover new, more sophisticated strategies that are immune to deceptive blocks.

4.3 Coevolution of bit patterns that target weaknesses in particle CAs

Other types of bit patterns are evolved by ICs in order to target weaknesses in particular particle CAs. The existence of such bit patterns can be demonstrated by looking at the classification performance *P-evolved* of a CA on a set of evolved ICs verses its performance *P-randomized* on the same set of ICs but in which the order of bits has been randomized, effectively destroying any bit patterns that might have evolved. If *P-randomized* $=$ *P-evolved*, then the evolved ICs do not contain any particular bit patterns to which the CA is sensitive. If *P-randomized* $>$ *P-evolved*, that gives evidence that the evolved set of ICs contains bit

patterns that specifically target the CA. If *P-randomized* < *P-evolved*, that gives evidence that the CA has evolved to do well on the specific bit patterns contained in the evolved ICs.

The lower panel in fig. 2 gives results along these lines for the same run of the coevolutionary model used in the top panel. To create the solid black line, every 10 generations we plotted *P-randomized* − *P-evolved* averaged over all unique CAs in the population at that generation, where the set of ICs is the set of all unique ICs in the population at that generation. It can be seen that *P-randomized*−*P-evolved* = 0 during the "default" stage. It becomes positive during the block-expanding stage, reflecting the evolution of deceptive blocks in the IC population, and remains positive during the particle stage, reflecting the evolution of bit patterns that specifically target particle strategies.

To show that the bit patterns targeting particle strategies are not simply deceptive blocks, we also plot *P-randomized* − *P-evolved* averaged over a set of five CAs from the final generation of the run (generation 5000) that use particle strategies and have relatively high performance values. This plot is given as the solid grey line in the lower panel of fig. 2. The ICs used to calculate these performances are the same ones used for the solid black line—all the unique ICs in the population at each generation. This is plotted every 10 generations. It can be seen that during the block-expanding stage *P-randomized*−*P-evolved* is negative, implying that the high-performance particle CAs have evolved specifically to perform well in the presence of deceptive blocks. However, *P-randomized* − *P-evolved* becomes positive during the particle stage, implying that these high-performance particle CAs have weaknesses that are targeted by specific bit patterns in the evolved ICs.

In summary, the data presented in fig. 2 give strong evidence that ICs indeed evolve so as to specifically exploit the weaknesses in the strategies used by CAs. This coevolution of the CAs and the ICs during the block-expanding stage is very important for the evolution of particle CAs, and is likely a major cause of the high efficacy of the coevolutionary model. The data suggest that the particle CAs are also exploited by the ICs during the particle stage. In the next section we will introduce a modification in the coevolutionary models that demonstrates some side-effects of this exploitation of particle CAs by the ICs.

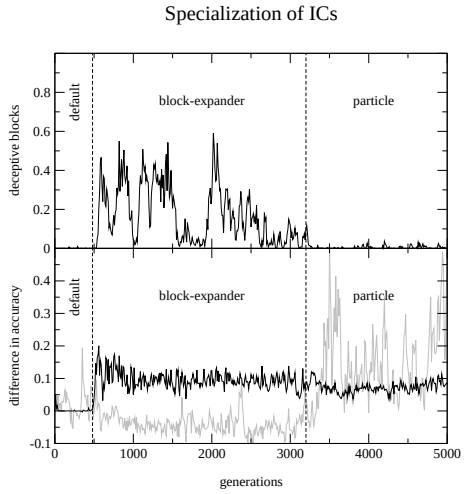

Figure 2: Coevolving ICs exploit CAs. In the top panel the proportion of ICs with deceptive blocks is plotted for a single coevolutionary simulation. When CAs use block-expanding strategies ICs tend to contain significant numbers of deceptive blocks. The plot in the lower panel further demonstrates that ICs are exploiting weaknesses in CA strategies. The solid black line gives, at every 10 generations, the average of *P-randomized* − *P-evolved* over all unique CAs in the population at that generation and the solid grey line gives the same quantity for five high-performance particle CAs from the final generation of the run. (See text for explanation and interpretation).

4.4 Results from a modified coevolutionary model

To further explore the effect of the evolution of particular bit patterns in the ICs, we modified the coevolutionary model so that no such patterns can form. In this modified model, instead of representing ICs as bit strings we represent them as density values. At each generation we create a new bit string for each IC with a corresponding density value (c.f. Werfel *et al.*, 2000, and Juillé & Pollack, 2000). The mutation of the density values is now defined as a unit change (i.e., $\pm\frac{1}{149}$) in the density value. The mutation rate is such that the same number of mutations events occur in the population of ICs as in the original coevolutionary model (i.e., 0.5× IC population size).

The third row in table 1 shows the efficacy of this modified model. In 12 out of 50 simulations (24%) particle CAs were found. The efficacy is thus much lower than in the original coevolutionary model, which is in accordance with the idea that CAs evolve particle strategies as a consequence of being exploited by the ICs when they use block-expanding strategies. In the modified model such bit-patterns can no longer be selected for. In comparison with the evolutionary model, however, this modified coevolutionary model still shows a higher efficacy. Apparently, the exploitation of CA strategies by evolving particular bit patterns is not the only mechanism that causes the high efficacy of the coevolutionary model.

The particle CAs that evolve in the modified model tend to have higher performance values than the particle CAs that evolve in the original coevolutionary model. In the

modified model the average of the highest performance particle CAs that are found during a simulation is 0.797 (stdev=0.023). However, the lowest performance value in this set of CAs is an outlier[‡] ($P = 0.7248$) without which the average is 0.804 (stdev=0.008). In the original coevolutionary model the average is 0.778 (stdev=0.013). The distribution of performance values of particle CAs in the original coevolutionary model is significantly below that of the particle CAs in the modified model ($P<0.001$ on standard t-test). The difference in the performance values of particle CAs that evolve in the two coevolutionary models suggests that the exploitation of the CAs by the ICs does not result in selection for CAs that perform general classification of bit strings. Rather, it seems likely that the CAs specialize on the population of ICs.

5 DISCUSSION

In this paper we compared evolutionary and coevolutionary search using the density classification task for cellular automata as the search problem. We showed that the coevolutionary search model exhibits a very high efficacy compared with the standard evolutionary model (82% versus 2%). The difference in the efficacy between the two models results from the exploitation by the coevolving population (the ICs) of lower-performance strategies, i.e., block-expanding strategies. When the ability of the ICs to exploit the CAs is removed, the efficacy of the coevolutionary model drops dramatically to 24%.

We have shown that the coevolving population explicitly targets weaknesses in the strategies that are used by the CAs. (A similar result was given by Pagie & Hogeweg, 1997.) As a result, the CAs undergo selection to adapt to this exploitation, which leads to the high efficacy in the coevolutionary model. Our results also suggest that the exploitation of CAs that use higher-performance strategies (particles) leads to the evolution of less general CAs than when the exploitation by the ICs is prevented. This aspect of coevolutionary exploitation in evolutionary search clearly needs further study.

Acknowledgments

This work was supported by the Santa Fe Institute and by National Science Foundation grant IRI-9705830.

References

ANDRE, D.; BENNETT III, F. H. & KOZA, J. R. (1996); Evolution of intricate long-distance communication signals in cellular automata using genetic programming. In: *Proceedings Artificial Life V*. MIT Press, Nara, Japan.

FICICI, S. G. & POLLACK, J. B. (1998); Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In: Adami, C.; Belew, R. K.; Kitano, H. & Taylor, C. (eds.), *Proceedings Artificial Life 6*. MIT Press, pp. 238–247.

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

HUSBANDS, P. (1994); Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In: Fogarty, T. (ed.), *Evolutionary Computing, AISB Workshop Selected Papers*. Springer-Verlag, vol. 865 of *LNCS*, pp. 150–165.

JUILLÉ, H. & POLLACK, J. B. (2000); Coevolutionary learning and the design of complex systems. *Advances in Complex Systems* **2**(4):371–394.

LAND, M. & BELEW, R. K. (1995); No perfect two-state cellular automata for density classification exists. *Physical Review Letters* **74**(25):5148–5150.

MITCHELL, M.; CRUTCHFIELD, J. P. & DAS, R. (1996); Evolving cellular automata with genetic algorithms: A review of recent work. In: *Proceedings of the First Int. Conf. on Evolutionary Computation and its Applications (EvCA'96)*.

OLIVEIRA, G. M. B.; DE OLIVEIRA, P. P. B. & OMAR, N. (2000); Evolving solutions of the density classification task in 1d cellular automata, guided by parameters that estimate their dynamic behavior. In: M.A. Bedau, J.S. McCaskill, N. P. & Rasmussen, S. (eds.), *Artificial Life VII*. MIT Press, pp. 428–436.

PACKARD, N. (1988); Adaptation towards the edge of chaos. In: Kelso, J. A. S.; Mandell, A. J. & Shlesinger, M. F. (eds.), *Dynamic Patterns in Complex Systems*, World Scientific. pp. 293–301.

PAGIE, L. (1999); Coevolutionary dynamics: information integration, speciation, and red queen dynamics. In: *Information integration in evolutionary processes*, Bioinformatics group, Utrecht University, www.santafe.edu/˜ludo/articles/thesis.pdf, chap. 5. pp. 67–93.

PAGIE, L. & HOGEWEG, P. (1997); Evolutionary consequences of coevolving targets. *Evolutionary Computation* **5**(4):401–418.

PAGIE, L. & HOGEWEG, P. (2000); Information integration and red queen dynamics in coevolutionary optimization. In: *Proceedings CEC 2000*. pp. 1260–1267.

PAREDIS, J. (1997); Coevolving cellular automata: be aware of the red queen! In: Baeck, T. (ed.), *Proceedings ICGA VII*. pp. 393–400.

POLLACK, J. B.; BLAIR, A. D. & LAND, M. (1997); Coevolution of a backgammon player. In: Langton, C. G. & Shimohara, K. (eds.), *Artificial Life V*. The MIT Press, pp. 92–98.

ROSIN, C. D. & BELEW, R. K. (1997); New methods for competitive coevolution. *Evolutionary Computation* **5**(1):1–29.

SHAPIRO, J. L. (1998); Does data-model co-evolution improve generalization performance of evolving learners? In: Eiben, A.; Bck, T.; Schoenauer, M. & Schwefel, H.-P. (eds.), *PPSN V*. vol. 1498 of *LNCS*, pp. 540 – 549.

WERFEL, J.; MITCHELL, M. & CRUTCHFIELD, J. P. (2000); Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation* **4**(4):388–393.

[‡]The outlier CA was found in a run at t=5000. When this simulation was continued the performance of the best CA increased to .786 within 500 generations.