Coevolving the "Ideal" Trainer: Application to the Discovery of Cellular Automata Rules

Hugues Juillé
Computer Science Department
Brandeis University
Waltham, Massachusetts 02254-9110, USA
hugues@cs.brandeis.edu

Jordan B. Pollack
Computer Science Department
Brandeis University
Waltham, Massachusetts 02254-9110, USA
pollack@cs.brandeis.edu

ABSTRACT

Coevolution provides a framework to implement search heuristics that are more elaborate than those driving the exploration of the state space in canonical evolutionary systems. However, some drawbacks have also to be overcome in order to ensure continuous progress on the long term. This paper presents the concept of coevolutionary learning and introduces a search procedure which successfully addresses the underlying impediments in coevolutionary search. The application of this algorithm to the discovery of cellular automata rules for a classification task is described. This work resulted in a significant improvement over previously known best rules for this task.

1 Introduction

Some problems are difficult because solutions have to be evaluated against a very large number of test cases in order to determine their score accurately. The discovery of game strategies and learning control procedures for autonomous agents are a few examples of such problems. To make learning tractable, solutions can be evaluated only with respect to a training environment composed of a subset of all the test cases. For such problems, a common approach in machine learning consists in designing a fixed training environment. Usually, a significant amount of knowledge about the problem is explicitly introduced by the designer in that stage. Then, the learning algorithm follows the gradient implemented in the training environment. However, the performance of solutions relies heavily on this training environment and on the search procedure to explore the state space. If little knowledge is available or if it is is difficult to introduce in the training environment, the performance of the system is very limited.

Coevolution is an alternative to get around that problem in the sense that the training environment for learners doesn't have to be designed explicitly. In coevolution, the performance of individuals is evaluated with respect to other members of the population, resulting in a dynamic fitness landscape. The counter part is to ensure that the changing environment does provide useful information to drive the search towards good solutions. In this paper, we define *coevolutionary learning* as a framework in which a population of learners coevolve with a population of problems such that *continuous progress* results from this interaction. By continuous progress, we mean that learners are able to solve an increasing range of problems (and eventually the entire set of problems that compose the task). However, multiple reasons can prevent such a system from continuously improving itself. In the literature, at least two such reasons have been recognized. The first one, named the Red Queen effect, comes from the fact that individuals are evaluated in a changing environment. As a result, they tend to specialize with respect to the current training environment and they tend to forget some of the traits they learnt some generations earlier because the relevant test cases for those traits are no longer present in the current training environment. Later, if those test cases reappear, the agents have to learn again the appropriate traits. The second reason is that the coevolving agents can enter a mediocre stable or meta-stable state in which a number of average performance species coexist in the population in a stable manner. In this situation, there is no driving force for agents to evolve. Any slight alteration of an individual in one species results in no improvement or a smaller performance.

In the research literature, Hillis' work marked an important step by showing that coevolution can be used to improve search performance (Hillis 1992). In his work, a population of sorters (the hosts) coevolve with input vec-

tors (the parasites). The goal of sorters is to construct sequences of comparator-swaps that sort the input vectors that are proposed by the parasites while parasites search for input vectors that are difficult to sort. In a sense, this can be seen as an implementation of a coverage-based heuristic: a construction is sought that sorts correctly every possible input vector, thus resulting in a sorting network. This heuristic adaptively focuses the search for solving problem instances (i.e., input vectors) that are the most difficult for the population of networks. This work has been followed by others using both competitive and cooperative models of coevolution. For instance, Husbands implemented a model similar to Hillis' to address a generalized version of the job-shop scheduling problem (Husbands 1994). Paredis (Paredis 1996) used competition between a population of solutions and a population of problems as a search strategy for applications in inductive learning (Paredis 1994*b*) and constraint satisfaction problems (Paredis 1994*a*). Pursuer/evader games have also been used as a test problem for research in coevolution. In particular, Cliff and Miller (Cliff and Miller 1995, Cliff and Miller 1996) developed several tools to track progress and detect loss of traits resulting from the Red Queen effect. Sims' block-creatures (Sims 1994) and Reynolds' experiments with the game of tag (Reynolds 1994) are also two successful applications of competitive evolution. Rosin's work on coevolutionary learning (Rosin 1997) addresses the different issues related to competitive evolution in the context of adversarial problems (e.g., game strategies). The goal of this work is to define a framework for coevolutionary search that results in continuous progress on the long term. In a theoretical analysis (Rosin and Belew 1996), Rosin and Belew described a coevolutionary environment and proved it allows the discovery of perfect game strategies.

Cooperative models of coevolution have also been implemented. Such models have been used for function optimization (Potter and De Jong 1994) and for the design of control systems (Potter *et al.* 1995). Another application is the search of a space of problem decompositions to construct modular solutions (Potter 1997, Moriarty 1997). Following a different track, Paredis (Paredis 1995) designed a model exploiting a symbiotic relationship to coevolve solutions and their representation.

In this paper, we introduce a framework which addresses the impediments related to coevolution just discussed and which does result in continuous improvement. Coevolutionary learning is based on the controlled evolution of the training environment in response to the improvement of the learners. The goal is to discover the "best" training environment, given the population of learners, hence the name of our approach: "coevolving the "ideal" trainer". The application of this system to a difficult problem, namely the discovery of cellular automata rules to implement the majority classification task, is described. Significantly better results are achieved by the coevolutionary approach compared to previously known solutions to that problem.

This paper is organized as follows. First, our new coevolutionary search procedure is described and the underlying heuristics exploited by this algorithm are identified. Then, section 3 presents cellular automata and the problem of evolving rules to implement a particular classification task. Section 4 describes the application of our coevolutionary approach to this problem, followed by experimental results in section 5.

2 Coevolving the "Ideal" Trainer

As it was discussed before, the main issues we want to address are how to escape from mediocre stable or metastable states and how to avoid the Red Queen effect.

2.1 Description

Since the space of problems to which learners can be exposed is huge, learners can be evaluated only against a subset of the problems. Therefore, the goal is to discover learners that are able to generalize to unseen problem instances after they have been exposed to this sample.

The central idea of the coevolutionary learning approach presented in this paper consists in exposing learners to problems that are just beyond those they know how to solve. By maintaining this constant pressure towards slightly more difficult problems, a arms race among learners is induced such that learners that adapt better have an evolutionary advantage. The underlying heuristic implemented by this arms race is that *adaptability* is the driving force for improvement. In order to achieve this result, our search algorithm tries to satisfy the following two goals:

- to provide an "optimum" gradient for search. This means that the training environment defined by the population of problems can determine reliably which learners are the most promising at each stage of the search process. This means that problems must be informative. If problems are too difficult, nobody can solve them. On the contrary, if they are too easy, everybody can solve them. In both of those cases, learners get little feedback and there is no gradient to determine in which direction the search should focus.

- to allow continuous progress. The goal is to avoid the Red Queen effect by providing a training environment which continues to test learners about problems they solved in the past. In a sense, the training environment must also play the role of memory.

The difficulty resides in the accurate implementation of those concepts in a search algorithm. So far, our method-

ology to implement such a system consists in the construction of an explicit topology over the space of problems by defining a partial order between problems. This partial order is defined with respect to the relative difficulty of problems among each other. In our current work, the concept of "relative difficulty" has been defined by exploiting some *a priori* knowledge about the global task and is task-specific. The definition of this topology over the space of problems makes possible the implementation of the two goals to achieve coevolutionary learning:

- since learners can be evaluated against a known range of difficulty for problems, it is possible to measure their progress and to expose them to problems that are just a little more difficult. However, this last operation also requires the definition of a distance measure in order to formalize the concept of "a little more difficult". Indeed, the topology over the space of problems is independent of the topology over the space of learners. The distance between two problems with respect to their relative difficulty is not necessarily a direct mapping with the probability for a learner that solves only one of them to adapt in order to solve both problems. In practice, several definitions might be tested and some tuning is required to get the appropriate result.

- the progress of learners can be monitored indirectly by tracing the evolution of problems towards increased difficulty. Conversely, the evolution in problems difficulty is controlled by the evolution of the performance of learners. Thus, the evolution of the training environment can be controlled such that the Red Queen effect is prevented (or at least such that its effect is limited).

In the future, our goal is to eliminate some of those explicit components by introducing some heuristics that automatically identify problems that are appropriate for the current set of learners while preventing the Red Queen effect. The work of Rosin (Rosin 1997) already describes some methods to address this issue.

2.2 Discussion

As stated previously, the coevolutionary learning framework introduces a pressure towards adaptability. The central assumption is that individuals that adapt faster than others in order to solve the new challenges they are exposed to are also more likely to solve even more difficult problems. The main difficulty is to setup a coevolutionary framework that implements this heuristic accurately and efficiently.

The idea of introducing a pressure towards adaptability as the central heuristic for search is not new. Schmidhuber (Schmidhuber 1995) proposed the Incremental Self-Improvement system in which adaptability is the measure that is optimized. In this system, the search is performed in a stochastic depth-first way and a Reinforcement Acceleration Criterion (RAC) is computed regularly in order to determine if there has been continuous acceleration of the amount of reinforcement information received from the environment since the birth of the individual. If RAC is not satisfied, a backtracking operation is performed. That is, the individual undoes its last modifications until RAC is satisfied again and new modifications are tried.

3 Discovery of CA Rules for a Classification Task

3.1 One-Dimensional Cellular Automata

A one-dimensional cellular automaton (CA) is a linear wrap-around array composed of N cells in which each cell can take one out of k possible states. A rule is defined for each cell in order to update its state. This rule determines the next state of a cell given its current state and the state of cells in a predefined neighborhood. For the model discussed in this paper, this neighborhood is composed of cells whose distance is at most r from the central cell. This operation is performed synchronously for all the cells in the CA. From now on, we will consider that the state of cells is binary ($k = 2$), $N = 149$ and $r = 3$. This means that the size of the rule space is $2^{2^{2*r+1}} = 2^{128}$.

Cellular automata have been studied widely as they represent one of the simplest systems in which complex emergent behaviors can be observed. This model is very attractive as a means to study complex systems in nature. Indeed, the evolution of such systems is ruled by simple, locally-interacting components which result in the emergence of global, coordinated activity.

3.2 The Majority Function

The task consists in discovering a rule for the one-dimensional CA which implements the majority function as accurately as possible. This is a density classification task, for which one wants the state of the cells of the CA to relax to all 0's or 1's depending on the density of the initial configuration (IC) of the CA, within a maximum of M time steps. Following (Mitchell *et al.* 1994), ρ_c denotes the threshold for the classification task (here, $\rho_c = 1/2$), ρ denotes the density of 1's in a configuration and ρ_o denotes the density of 1's in the initial configuration. Figure 1 presents two examples of the space-time evolution of a CA for $N = 149$ with $\rho_0 < \rho_c$ on the left and $\rho_0 > \rho_c$ on the right. The initial configuration is at the top of the diagram and the evolution in time of the different configurations is represented downward.

The task $\rho_c = 1/2$ is known to be difficult. In particular, it has been proven that no rule exists that will result in the CA relaxing to the correct state for all possible ICs (Land and Belew 1995). Indeed, the density is

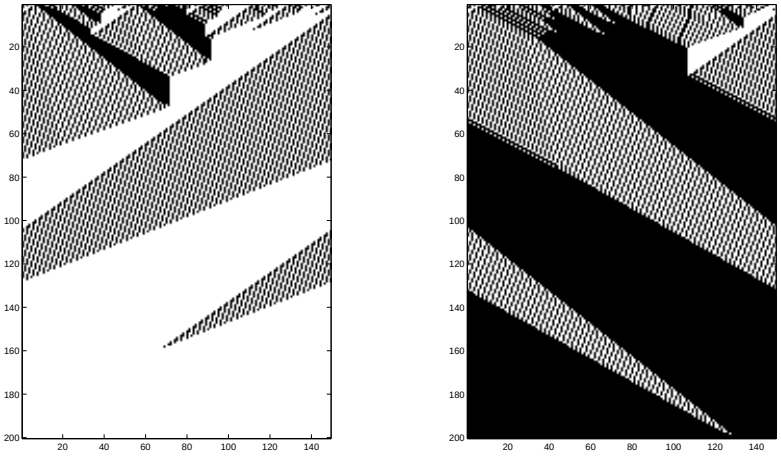

Figure 1 Two space-time diagrams describing the evolution of CA states for the new rule Coevolution (2) which scores 86.0% for $N = 149$. White squares represent cells in state 0 while black squares correspond to cells in state 1. The initial state of the CA is at the top and the evolution for the first 200 time steps is represented by moving downward.

a global property of the initial configuration while individual cells of the CA have access to local information only. Discovering a rule that will display the appropriate computation by the CA with the highest accuracy is a challenge, and the upper limit for this accuracy is still unknown. Table 1 describes the performance for that task for different published rules and different values of N. The Gacs-Kurdyumov-Levin (GKL) rule was designed in 1978 for a different goal than the $\rho_c = 1/2$ task (Mitchell *et al.* 1994). However, for a while it provided the best known performance. (Mitchell *et al.* 1994) and (Das *et al.* 1994) used Genetic Algorithms (GAs) to explore the space of rules. This work resulted in an analysis of some of the complex behaviors exhibited by CAs using "particles". The GKL and Das rules are human-written while the Andre-Bennett-Koza (ABK) rule has been discovered using the Genetic Programming paradigm (Andre *et al.* 1996). For the $\rho_c = 1/2$ task, it is believed that the rules that perform reasonably well have a density close to 0.5 and, indeed, the GKL rule has density 0.5 exactly. An intuitive argument to support this hypothesis is presented in (Mitchell *et al.* 1993). It is also believed that the most difficult ICs are those with density close to 0.5 (since only a little modification can make them switch from $\rho_0 < 1/2$ to $\rho_0 > 1/2$, and vice versa). This information is useful for the understanding of the experimental analysis presented in the following sections.

In the research literature, initial work performed by (Das *et al.* 1994, Mitchell *et al.* 1994) has been followed by (Andre *et al.* 1996) whose rule improved the case $N = 149$ but doesn't generalize as well as the GKL or the Das rule. (Sipper 1994) evolved rules for non-homogeneous CA for which each cell has its own independent version of a rule. (Paredis 1997) describes a

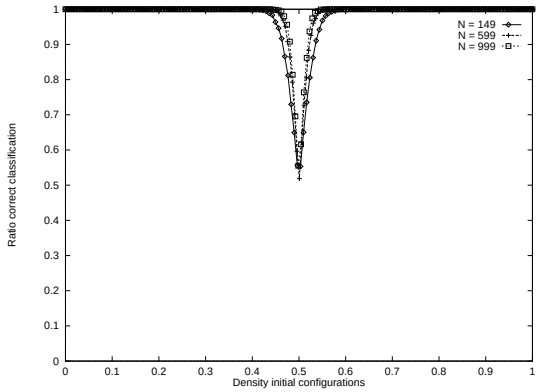

Figure 2 Distribution of performance for the GKL rule for $\rho_0 \in [0.0, 1.0]$.

coevolutionary approach to search the space of rules and shows the difficulty of coevolving consistently two populations towards continuous improvement. (Capcarrere *et al.* 1996) also reports that by changing the specification of the convergence pattern of the CA from all $0's$ or all $1's$ to a pattern in which a block of at least two consecutive $1's$ exists if and only if $\rho_0 > 1/2$ and a block of at least two consecutive $0's$ exists if and only if $\rho_0 < 1/2$, then a two-state, $r = 1$ cellular automaton exists that can perfectly solve the density problem.

4 Implementation of the Concept

The coevolutionary approach described in section 2 is applied to the $\rho_c = 1/2$ task. It is believed, that ICs become more and more difficult to classify correctly as their density gets closer to the ρ_c threshold. This hypothesis is

Table 1 Performance of different published CA rules and two new rules discovered using the coevolutionary learning paradigm for the $\rho_c = 1/2$ task.

N	149	599	999
Coevolution (1)	0.851 +/- 0.001	0.810 +/- 0.001	0.795 +/- 0.001
Coevolution (2)	0.860 +/- 0.001	0.802 +/- 0.001	0.785 +/- 0.001
Das rule	0.823 +/- 0.001	0.778 +/- 0.001	0.764 +/- 0.001
ABK rule	0.824 +/- 0.001	0.764 +/- 0.001	0.730 +/- 0.001
GKL rule	0.815 +/- 0.001	0.773 +/- 0.001	0.759 +/- 0.001

supported by the distribution of the performance for the GKL rule for $\rho_0 \in [0.0, 1.0]$ presented in figure 2. Therefore, our idea is to construct a framework that adapts the distribution of the density for the population of ICs as CA-rules are getting better to solve the task. The following definition for the fitness of rules and ICs has been used to achieve this goal.

$$f(R_i) = \sum_{j=1}^{n_{IC}} W\_IC_j \times covered(R_i, IC_j)$$

where:

$$W\_IC_j = \frac{1}{\sum_{k=1}^{n_R} covered(R_k, IC_j)}$$

and

$$f(IC_j) = \sum_{i=1}^{n_R} W\_R'_i \times E(R_i, \rho(IC_j)) \times \overline{covered(R_i, IC_j)}$$

where:

$$W\_R'_i = \frac{1}{\sum_{k=1}^{n_{IC}} E(R_i, \rho(IC_k)) \times \overline{covered(R_i, IC_k)}}$$

where $covered(R_i, IC_j)$ returns 1 if a CA using rule R_i and starting from initial configuration IC_j relaxes to the correct state. Otherwise, it returns 0. $\overline{covered(R_i, IC_j)}$ returns the complement of $covered(R_i, IC_j)$.

This definition implements a competitive relationship between rules and ICs. Rules get a higher payoff by covering more ICs accurately while ICs get a higher payoff by defeating more rules. This definition also implements a form of niching called resource sharing. Resource sharing implements a coverage-based heuristic by giving a higher payoff to problems that few individuals can solve. In our case, it is introduced in the definition of the fitness of rules and ICs by weighting the outcome of each interaction rule–IC. The weight of an IC corresponds to the payoff it returns if a rule covers it. The underlying mechanism is that if few rules cover an IC, this weight will be much larger than if a lot of rules cover that same IC. The definition for the weight of rules is similar. The benefit of resource sharing in the context of search has already been discussed in (Juillé and Pollack 1996).

The definition of the ICs' fitness has been extended with a new component, namely $E(R_i, \rho(IC_j))$. The purpose of this new component is to penalize ICs with density $\rho(IC_j)$ if little information is collected with respect to the rule R_i. Indeed, we consider that if a rule R_i has a 50% classification accuracy over ICs with density $\rho(IC_j)$ then this is equivalent to random guessing and no payoff should be returned to IC_j. On the contrary, if the performance of R_i is significantly better or worse than the 50% threshold for a given density of ICs this means that R_i captured some relevant properties to deal with those ICs. Once again, the idea is that the training environment for rules should be composed of ICs that provide useful information to identify good rules from poor ones. ICs for which the performance of rules is close to 50% are useless to satisfy this goal and they shouldn't be explored further. The training environment should also allow continuous progress by preventing the Red Queen effect. In our implementation, there is no explicit mechanism to satisfy this purpose. Instead, an intrinsic property of the $\rho_c = 1/2$ task is exploited in order to satisfy this goal indirectly. Indeed, it seems that CA-rules that cover ICs with density $\rho_0 < 1/2$ (respectively $\rho_0 > 1/2$) with high performance will also be very successful over ICs with density $\rho'_0 < \rho_0$ (respectively $\rho'_0 > \rho_0$). Therefore, as ICs become more difficult, their density is approaching $\rho_0 = 1/2$ but rules don't have to be tested against easier ICs.

Following this idea, we defined $E()$ as the complement of the entropy of the outcome between a rule and ICs with a given density:

$$E(R_i, \rho(IC_j)) = \log(2) + p\log(p) + q\log(q)$$

where: p is the probability that an IC with density $\rho(IC_j)$ defeats the rule R_i and $q = 1 - p$. Because of this credit assignment strategy, a balance is maintained between the search for more difficult ICs (competitive mode of interaction) and ICs that can be solved by rules (cooperation between rules and ICs). In practice, entropy is evaluated by performing some statistics over the population of ICs.

5 Experimental Results

The best two CA-rules discovered so far are presented in table 1. Those two new rules exhibit a very significant improvement over the previously known best rules with respect to the case $N = 149$ as well as the generalization ability (tested with $N = 599$ and $N = 999$). The description of those CA rules is presented in table 2. along with the description for the Das, ABK and GKL rules. The lookup tables described in table 2 are using the trivial coding: the leftmost bit corresponds to the output of the rule with input 0000000, the second bit corresponds to input 0000001, ... and the rightmost bit corresponds to input 1111111.

In a first set of experiments composed of about 20 runs, the population size for rules and ICs was 400. The implementation to search the space of rules is similar to the one described in (Mitchell *et al.* 1994). Each rule is coded on a binary string of length $2^{2*r+1} = 128$. One-point crossover is used with a 2% bit mutation probability. The population of rules is initialized according to a uniform distribution over $[0.0, 1.0]$ for the density. Each individual in the population of ICs represents a density $\rho_0 \in [0.0, 1.0]$. This population is also initialized according to a uniform distribution over $\rho_0 \in [0.0, 1.0]$. At each generation, each member generates a new instance for an initial configuration with respect to the density it represents. All rules are evaluated against this new set of ICs. The generation gap is 5% for the population of ICs (i.e., the top 95% ICs reproduce to the next generation). There is no crossover nor mutation. The new 5% ICs are the result of a random sampling over $\rho_0 \in [0.0, 1.0]$ according to a uniform probability distribution. The generation gap is 80% for the population of rules. New rules are created by crossover and mutation. Parents are randomly selected from the top 20%. This choice for the value of the generation gap is a compromise between speed of search and performance. As a result, some of the runs return a poor result because of unfavorable sampling (in those runs, evolved rules don't score more than 76%). The rule "Coevolution (1)" presented in table 1 resulted from one of the runs in this experiments.

In a second of experiments composed of 8 runs, using a population size of 400 for rules and ICs and a generation gap of 10% for the population of rules, some rules were evolved that consistently scored above 80%. In that case, good rules are less likely to disappear from the population but progress is very slow. It might however be possible to improve the time performance by avoiding redundant computation for the evaluation of rules that have been in the population for several generations.

In another set of experiments composed of 6 runs, using a population size of 1000 for both rules and ICs and the same value for the generation gaps as for the first set of experiments, all runs consistently evolved some rules that score above 82%. The rule "Coevolution (2)" presented in table 1 resulted from one of those runs. The goal of those experiments was to test how the performance of evolved rules would scale when increasing the population size. It might be possible to evolve even better rules with a larger population size. However, with our current implementation, a run takes about one week on a workstation for 5000 generations and a population size of 1000.

As a comparison, (Andre *et al.* 1996) used a population of size $51,200$. In their work, the training environment was composed of a fixed training set constructed from a uniform sampling from the space of all ICs (thus, the distribution for the density of ICs in the training set is binomial, centered on $1/2$). In experiments described in (Das *et al.* 1994, Mitchell *et al.* 1994), the learning environment is composed of a set of ICs sampled at each generation according to a uniform distribution over $\rho_0 \in [0.0, 1.0]$. Those authors acknowledged that this distribution for the sampling of the space of ICs, while helpful to bootstrap the search, might no longer provide useful information once some average performance rules have been discovered.

Figures 3 and 4 describe the evolution of the density of rules and ICs for two runs. As rules improve, their density gets closer to $1/2$ and the density of ICs is distributed on two peaks on each side of $\rho = 1/2$. In the case of figure 4, it is only after $1,300$ generations that a significant improvement is observed for rules. It is only at that time that the population of ICs adapts dramatically in order to propose more challenging initial configurations. This shows that our strategy to coevolve the training environment and the learners has been successfully implemented in the definition of the fitness functions. However, it should be noted that the two-peak distribution is a side-effect of the method implemented to measure entropy. Indeed, because of the small population size with respect to the range of values for the density (i.e., N values), ICs were grouped in bins of size two. That is, there are $N/2$ bins instead of N to cover the range of densities. The two-peak distribution means that, in our experiments, evolved rules have an accuracy very close to 50% with respect to the set of ICs in the bin composed of ICs with density $1/\lceil \frac{N}{2} \rceil$ and $1/\lfloor \frac{N}{2} \rfloor$. In experiments that use N bins, the final distribution is composed of a single peak centered on $\rho_0 = 1/2$. However, empirical evidence seems to show that this two-peak distribution results in better performance for the final CA-rules. This is supported by figure 5 which compares the distribution of the performance of the new rules to the GKL rule. This distribution is represented only in the neighborhood of $\rho_0 = 1/2$. Outside the range represented in those figures, the ratio of correct classification is (or is very close to) 100% for the three rules. As discussed before, those figures confirm that for rules with good classification per-

Table 2 Description of the current best rules and previously published rules for the $\rho_c = 1/2$ task.

Coevolution (1)	00000001	00010100	00110000	11010111	00010001	00001111	00111001	01010111
	00000101	10110100	11111111	00010111	11110001	00111101	11111001	01010111
Coevolution (2)	00010100	01010001	00110000	01011100	00000000	01010000	11001110	01011111
	00010111	00010001	11111111	01011111	00001111	01010011	11001111	01011111
Das rule	00000111	00000000	00000111	11111111	00001111	00000000	00001111	11111111
	00001111	00000000	00000111	11111111	00001111	00110001	00001111	11111111
ABK rule	00000101	00000000	01010101	00000101	00000101	00000000	01010101	00000101
	01010101	11111111	01010101	11111111	01010101	11111111	01010101	11111111
GKL rule	00000000	01011111	00000000	01011111	00000000	01011111	00000000	01011111
	00000000	01011111	11111111	01011111	00000000	01011111	11111111	01011111

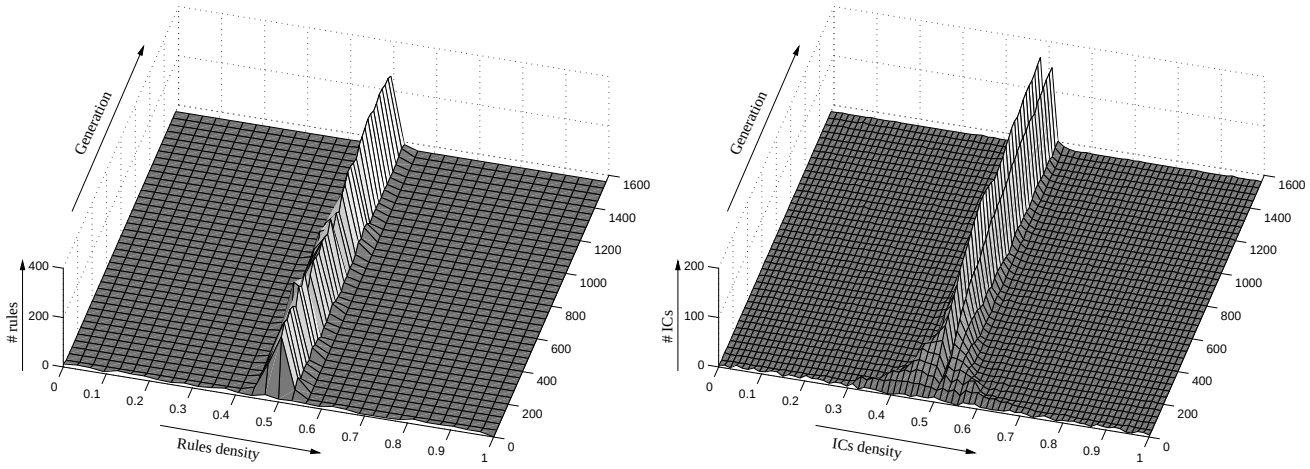

Figure 3 Evolution in time of the distribution of CA rules density (left) and ICs density (right) describing the coevolution of rules and ICs.

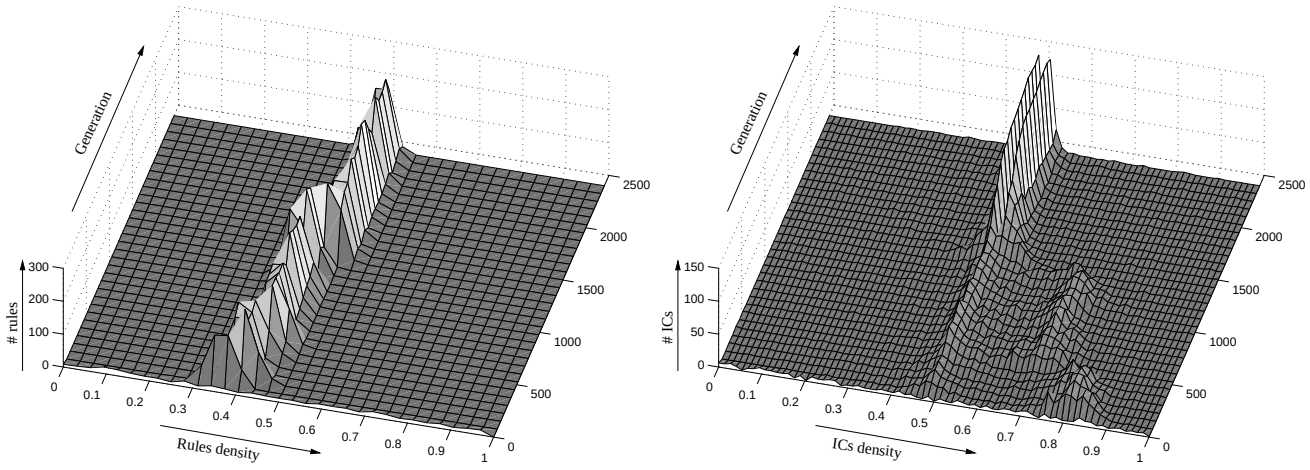

Figure 4 Evolution in time of the distribution of CA rules density (left) and ICs density (right). In that case, the sharp transition around generation 1,300 corresponds to an improvement of rules and results in an adaptation of the distribution of ICs to present more challenging problems.

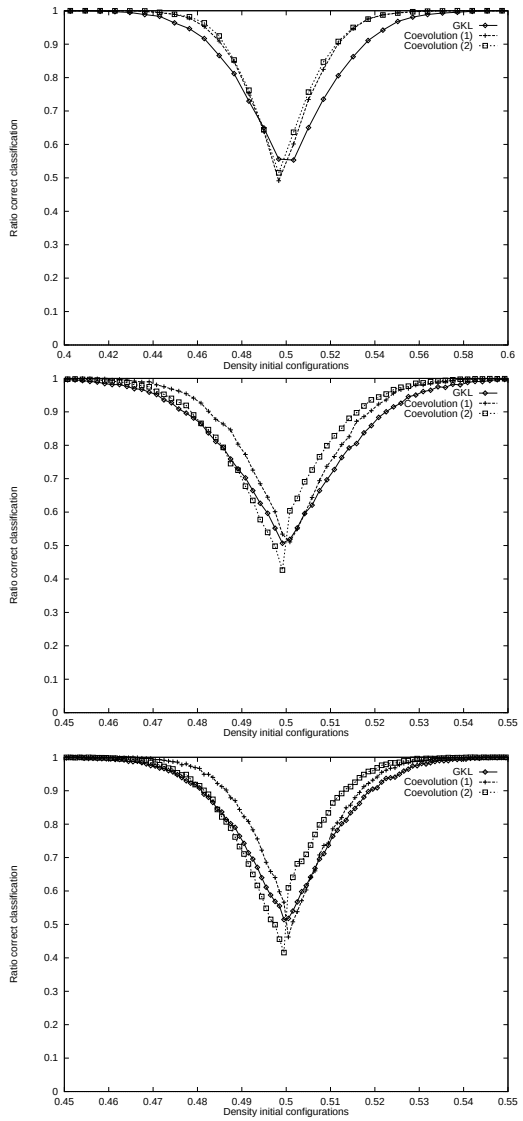

Figure 5 Comparison of the distribution of performance for the two new rules and the GKL rule for N = 149 (top), 599 and 999 (bottom).

formance, the most difficult ICs are the ones with density close to $\rho_0 = 1/2$. Therefore, if the density of ICs in the population of initial conditions converge to a single peak centered on $\rho_0 = 1/2$, the training environment would provide only little feedback (since the ratio of correct classification is very close to 50%). It would be difficult to distinguish rules with high performance from others because even average performance rules have a similar behavior in this area of the space of ICs.

6 Conclusion

This paper presents the concept of *coevolutionary learning*, a new framework for learning based on the coevolution between learners and problems. The exploration of the space of learners and problems is performed such that:

- "optimum" gradient information is provided to learners, and
- continuous progress is maintained.

The work presented in this paper addresses those issues by defining a topology over the space of problems. Then, a procedure is implemented such that the training environment automatically adapts in response to the progress of learners by proposing more challenging problems. This approach allows a more reliable estimate of the absolute performance of learners while providing an efficient gradient for search. We apply this framework to the problem of evolving CA rules for a classification task. Our experiments resulted in new rules whose performance improves very significantly over previously known rules for that particular task.

Another goal of this paper is to provide some insights on the use of coevolutionary approaches for search algorithms. By providing a methodology in which individuals are evaluated in a changing environment, more elaborate heuristics can be implemented for search. In the system presented in this paper, the principal underlying heuristic introduces a pressure towards adaptability. In previous work (Juillé and Pollack 1996), coevolution was used as a niching technique to implement a coverage-based heuristic.

Acknowledgment

The authors gratefully thank Melanie Mitchell for her help and useful discussions. David Andre made helpful comments on a preliminary version of this paper.

References

Andre, David, Forrest H. Bennett III and John R. Koza (1996). Evolution of intricate long-distance communication signals in cellular automata using genetic programming. In: *Proceedings of the Fifth Artificial Life Conference*. pp. 16–18.

Capcarrere, M. S., M. Sipper and M. Tomassini (1996). Two-state, r=1 cellular automaton that classifies density. *Physical Review Letters* **77**(24), 4969–4971.

Cliff, Dave and Geoffrey F. Miller (1995). Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In: *The Third European Conference on Artificial Life*. Springer-Verlag. pp. 200–218. LNCS 929.

Cliff, Dave and Geoffrey F. Miller (1996). Co-evolution of pursuit and evasion ii: Simulation methods and results. In: *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*

(Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack and Stewart W. Wilson, Eds.). MIT Press. Cambridge, Massachusetts. pp. 506–515.

Das, Rajarshi, Melanie Mitchell and James P. Crutchfield (1994). A genetic algorithm discovers particle-based computation in cellular automata. In: *Parallel Problem Solving from Nature III, LNCS 866.* Springer-Verlag. pp. 344–353.

Hillis, W. Daniel (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In: *Artificial Life II* (Chris Langton et al., Eds.). Addison Wesley. pp. 313–324.

Husbands, Phil (1994). Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In: *Proceedings of Evolutionary computing, AISB Workshop Selected Papers* (T. Fogarty, Ed.). Springer-Verlag. pp. 150–165. LNCS 865.

Juillé, Hugues and Jordan B. Pollack (1996). Co-evolving intertwined spirals. In: *Proceedings of the Fifth Annual Conference on Evolutionary Programming.* MIT Press. pp. 461–468.

Land, Mark and Richard K. Belew (1995). No perfect two-state cellular automata for density classification exists. *Physical Review Letters* **74**(25), 1548–1550.

Mitchell, Melanie, James P. Crutchfield and Peter T. Hraber (1994). Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D* **75**, 361–391.

Mitchell, Melanie, Peter T. Hraber and James P. Crutchfield (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems* **7**, 89–130.

Moriarty, David Eric (1997). Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. PhD thesis. University of Texas at Austin, USA.

Paredis, Jan (1994*a*). Co-evolutionary constraint satisfaction. In: *Parallel Problem Solving from Nature - PPSN III, LNCS 866.* Springer-Verlag. pp. 46–55.

Paredis, Jan (1994*b*). Steps towards co-evolutionary classification neural networks. In: *Artificial Life IV* (Brooks and Maes, Eds.). MIT Press. pp. 102–108.

Paredis, Jan (1995). The symbiotic evolution of solutions and their representations. In: *Proceedings of the Sixth International Conference on Genetic Algorithms* (Larry J. Eshelman, Ed.). Morgan Kaufmann. pp. 359–365.

Paredis, Jan (1996). Coevolutionary computation. *Artificial Life.* To appear.

Paredis, Jan (1997). Coevolving cellular automata: Be aware of the red queen!. In: *Proceedings of the Seventh International Conference on Genetic Algorithms* (Thomas Bäck, Ed.). Morgan Kaufmann. pp. 393–400.

Potter, Mitchell A. (1997). The Design and Analysis of a Computational Model of Cooperative Coevolution. PhD thesis. George Mason University, Fairfax, Virginia.

Potter, Mitchell A. and Kenneth A. De Jong (1994). A cooperative coevolutionary approach to function optimization. In: *Parallel Problem Solving from Nature - PPSN III, LNCS 866.* Springer-Verlag. pp. 249–257.

Potter, Mitchell A., Kenneth A. De Jong and John J. Grefenstette (1995). A coevolutionary approach to learning sequential decision rules. In: *Proceedings of the Sixth International Conference on Genetic Algorithms* (Larry J. Eshelman, Ed.). Morgan Kauffmann. San Mateo, California. pp. 366–372.

Reynolds, Craig W. (1994). Competition, coevolution, and the game of tag. In: *Artificial Life IV* (Brooks and Maes, Eds.). MIT Press.

Rosin, Christopher D. and Richard K. Belew (1996). A competitive approach to game learning. In: *Proceedings of the Ninth Annual ACM Conference on Computational Learning Theory.*

Rosin, Christopher Darrell (1997). Coevolutionary Search Among Adversaries. PhD thesis. University of California, San Diego.

Schmidhuber, Jürgen (1995). Discovering solutions with low kolmogorov complexity and high generalization capability. In: *Machine Learning: Proceedings of the twelfth International Conference* (A. Prieditis amd S. Russell, Ed.). Morgan Kaufmann. pp. 188–196.

Sims, Karl (1994). Evolving 3d morphology and behavior by competition. In: *Artificial Life IV* (Brooks and Maes, Eds.). MIT Press. pp. 28–39.

Sipper, Moshe (1994). Coevolving non-uniform cellular automata to perform computations. *Physica D* **92**, 193–208.