Coevolutionary Learning: a Case Study

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

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

Abstract

Coevolutionary learning, which involves the embedding of adaptive learning agents in a fitness environment that dynamically responds to their progress, is a potential solution for many technological chicken and egg problems. However, several impediments have to be overcome in order for coevolutionary learning to achieve continuous progress in the long term. This paper presents some of those problems and proposes a framework to address them. This presentation is illustrated with a case study: the evolution of CA rules. Our application of coevolutionary learning resulted in a very significant improvement for that problem compared to the best known results.

1 Introduction

A recurrent issue in the field of machine learning is that the performance of a learning system relies heavily on the amount of knowledge that has been introduced by the designer. This knowledge can be expressed in the form of an appropriate representation, specific search operators, a training set which provides a good gradient or a special utility function. The success of most learning systems actually results from all this engineering effort.

However, the goal of machine learning is a system that can improve itself by continuously capturing and exploiting new knowledge. The framework which is presented in this paper to achieve such a goal is based on a coevolutionary approach. An important factor in the performance of learning systems is the design of a training environment. Usually, this training environment is fixed and constructed by the human designer. However, when little knowledge is available about the problem or if this knowledge is difficult to introduce in the training environment, learning can become intractable. The approach proposed in this paper to get round that problem consists of coevolving the training environment with a population of learners. Starting with simple problems, the training environment gets more challenging as learners are improving themselves. Hopefully, such a setup leads to continuous progress. For the rest of the paper, we define *coevolutionary learning* as a search procedure involving a population of learners coevolving with a population of problems such that *continuous progress* results from this interaction.

In practice, the picture is not that simple. We will discuss the different issues that are involved to achieve coevolutionary learning by considering a particular problem: the discovery of cellular automata rules to implement a classification task. This problem presents some interesting properties that provide us with a simple framework to monitor the dynamics of the search resulting from different setups. Section 2 describes this problem. In section 3, an experimental analysis presents the different impediments to coevolutionary learning and a solution to address them is proposed in section 4. Experimental results for the classification problem are presented in section 5.

2 Description of the Problem

2.1 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

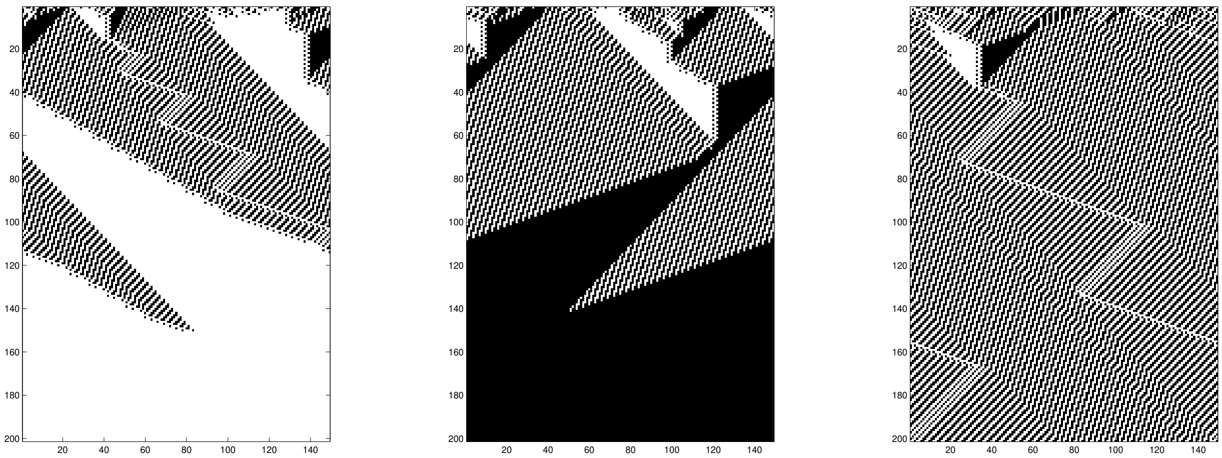

Figure 1: Three space-time diagrams describing the evolution of CA states: in the first two, the CA relaxes to the correct uniform pattern while in the third one it doesn't converge at all to a fixed point.

Table 1: Performance of different published CA rules and a new best rule for the $\rho_c = 1/2$ task.

N	149	599	999
Coevolution	0.863 +/- 0.001	0.822 +/- 0.001	0.804 +/- 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

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.

2.2 The Majority Function

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 all 1's depending on the density of the initial configuration (IC) (whether it has more 0's or more 1's), 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 three examples of the space-time evolution of a CA. One with $\rho_0 < \rho_c$ on the left and another with $\rho_0 > \rho_c$ in the middle for which the CA relaxes to the correct configuration. The third one shows an instance for which the CA doesn't relax to any the two desired convergence patterns. For each diagram, the initial configuration is at the top and the evolution in time of the state of the CA 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 results in the CA relaxing to the correct state for all possible ICs [Land & Belew, 1995]. Indeed, the density is 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, along with the performance of the new best rule that resulted from the work presented in this paper. The Gacs-Kurdyumov-Levin (GKL) rule was designed in 1978 for a different goal than solving 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. The main purpose of this work was to develop a particle-based methodology for the analysis of the complex behaviors exhibited by CAs. 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]. More recently, [Paredis, 1997] describes a 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, a two-state, $r = 1$ CA exists that can perfectly solve the density problem in $\lceil N/2 \rceil$ time steps.

For the $\rho_c = 1/2$ task, it is believed that the best rules are in the domain of the rule space with density close to 0.5. 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.

3 Models for Coevolutionary Search

The idea of using coevolution in search was introduced by [Hillis, 1992]. In coevolution, individuals are evaluated with respect to other individuals instead of a fixed environment (or landscape). As a result, agents adapt in response to other agents' behavior. The particular model of coevolution considered in this paper is based on two populations for which the fitness of individuals in each population is defined with respect to the members of the other population. Two cases can be considered in such a framework, depending on whether the two populations benefit from each other or whether they have different interests. Those two modes of interaction are called *cooperative* and *competitive* respectively. In the following sections, those modes of interaction are described experimentally using the $\rho_c = 1/2$ task in order to stress the different issues related to coevolutionary learning.

For the experiments presented in this section, we used an implementation of Genetic Algorithms 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 size is $n_R = 200$ for rules and $n_{IC} = 200$ for ICs. The population of ICs is composed of binary strings of length $N = 149$. The population of rules and ICs are initialized according to a uniform distribution over $[0.0, 1.0]$ for the density. For all the experiments in this paper, the value of M (the maximum number of time steps) is set to 320 and is kept unchanged. At each generation, the top 95% of each population reproduces to the next generation and the remaining 5% is the result of crossover between parents from the top 95% selected using a fitness proportionate rule. This small generation gap (the percentage of new individuals) has been used because of the dynamic fitness landscape. Indeed, a large generation gap can result in a dramatic change in the composition of the population. As a consequence, because of the relative definition of the fitness, a lot of variation in individuals' fitness can occur from one generation to the other, making the identification of the most promising individuals very unreliable.

3.1 Cooperation between Populations

In this mode of interaction, improvement on one side results in positive feedback on the other side. As a result, there is a reinforcement of the relationship between the two populations. From a search point of view, this can be seen as an *exploitative* strategy. Agents are not encouraged to explore new areas of the search space but only to perform local search in order to further improve the strength of the relationship. In the cooperative model, a natural definition for the fitness of rules (resp. ICs) is the number of ICs (resp. rules) for which the CA relaxes to the correct state:

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

$$f(IC_j) = \sum_{i=1}^{n_R} covered(R_i, IC_j)$$

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.

Figure 2 presents the evolution of the density of rules and ICs for one run using this cooperative model. Without any surprise, the population of rules and ICs quickly converge to a domain of the search space where ICs are easy for rules and rules consistently solve ICs. As a result, there is little exploration of the search space. The convergence configuration depends on the initial populations, some other runs ended up with low

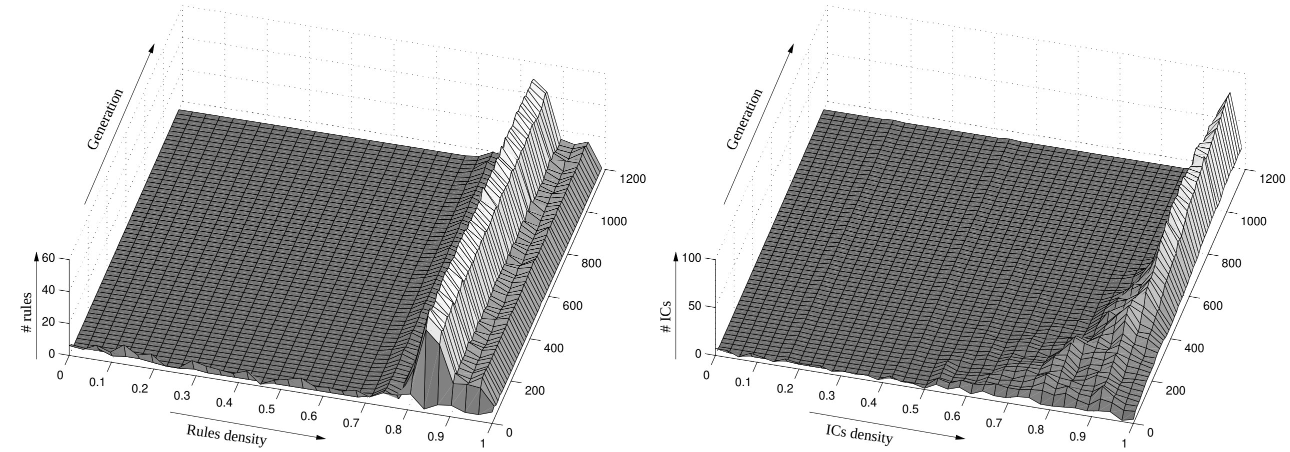

Figure 2: Coevolution of CA rules (left) and ICs (right) in a cooperative relationship.

density rules and ICs. This experiment confirms that ICs with low or high density are the easiest to classify since a larger number of rules classify them correctly.

3.2 Competition between Populations

In this mode of interaction, the two populations are in conflict. Improvement on one side results in negative feedback for the other population. The fitness of rules and ICs defined in the cooperative case can be modified as follows to implement the competitive model:

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

$$f(IC_j) = \sum_{i=1}^{n_R} \overline{covered(R_i, IC_j)}$$

where $\overline{covered(R_i, IC_j)}$ returns the inverse of the original function. Here, the goal of rules is to defeat (i.e. cover) ICs, while the goal of ICs is to defeat rules by discovering initial configurations that are difficult to classify. Figure 3 describes a run using this definition of the fitness. Two kind of behaviors can be observed in this experiment. In a first stage, the two populations exhibit a cyclic behavior. It is a consequence of the *Red Queen* effect [Cliff & Miller, 1995]: fitness landscapes are changing as a result of agents of each population adapting in response to the evolution of members of the other population. The evaluation of individuals' performance in this changing environment makes continuous progress difficult. A typical consequence is that agents have to learn again what they already knew in the past. In the context of evolutionary search, this means that domains of the state space that have already been explored in the past are searched again. Then, a stable state is reached: in this case, the population of rules adapts faster than the population of ICs, resulting in a population focusing only on rules with high density and eliminating all instances of low density rules (a finite population is considered). Then, low density ICs exploit those rules and overcome the entire population. A similar experiment is described in [Paredis, 1997].

3.3 Resource Sharing and Mediocre Stable States

Several techniques have been designed to improve evolutionary search. Usually they maintain diversity in the population in order to avoid premature convergence. [Mahfoud, 1995] presents different niching techniques that achieve this goal. Resource sharing, first introduced in [Rosin & Belew, 1995], is a technique that we successfully used in the past [Juillé & Pollack, 1996]. Resource sharing implements a coverage-based heuristic by giving a higher payoff to problems that few individuals can solve. Resource sharing can be introduced in the competitive model of coevolution as follows:

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

where:

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

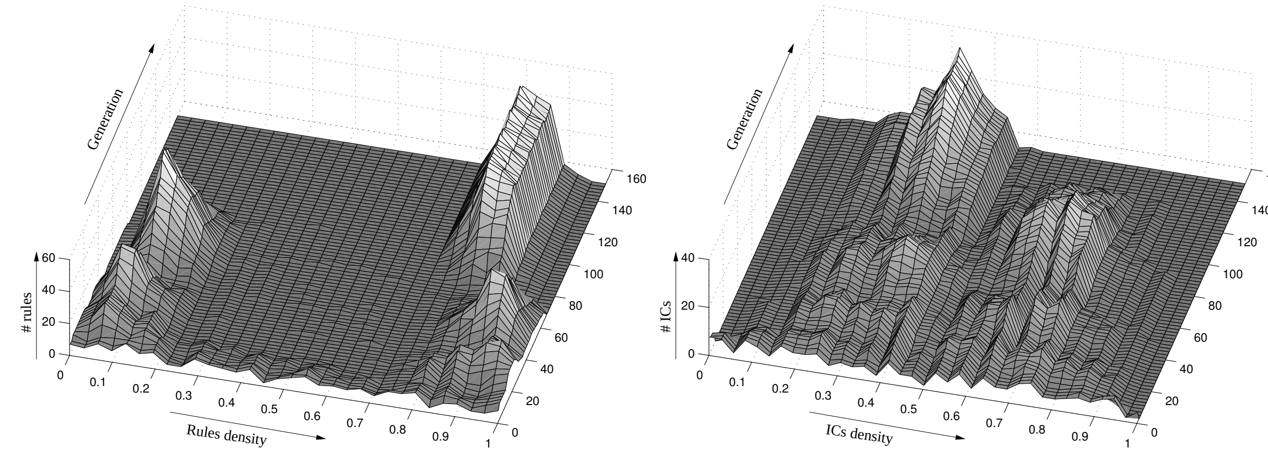

Figure 3: Coevolution of CA rules (left) and ICs (right) in a competitive relationship.

and

$$f(IC_j) = \sum_{i=1}^{n_R} weight\_R_i \times \overline{covered(R_i, IC_j)}$$

where:

$$weight\_R_i = \frac{1}{\sum_{k=1}^{n_{IC}} \overline{covered(R_i, IC_k)}}$$

In this definition, the weight of an IC corresponds to the payoff it returns if a rule covers it. 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 has the same purpose. This framework allows the presence of multiple niches (or species) in populations. Figure 4 describes one run for this definition of the fitness. The cyclic behavior which was observed in the previous section doesn't occur anymore. Instead, two species coexist in the population of rules: a species for low density rules and another one for high density rules. Those two species drive the evolution of ICs towards the domain of initial configurations that are most difficult to classify (i.e., $\rho_0 = 1/2$). However, the two populations have entered a *mediocre stable state*. This means that multiple average performance niches coexist in both populations in a stable manner. Put in another way, this can be seen as an equilibrium configuration in which a number of suboptimal species have found a way to collude by sharing the total credit between themselves. Usually, this is a consequence of some singularities inherent in the problem definition and/or the search procedure. In our example, ICs are concentrated around the $\rho_0 = 1/2$ threshold and they can be divided into two groups: those with density $\rho_0 < 1/2$ and those with density $\rho_0 > 1/2$. This distribution means that ICs can be exploited consistently by rules with low and high density that both occur in the second population (because a CA implementing a low (resp. high) density rule usually relaxes to all 0's (resp. all 1's) for most ICs). However, this is a mediocre stable state in the sense that evolved rules have poor performance with respect to the $\rho_c = 1/2$ task and there is no pressure towards improvement. The concept of mediocre stable states is also discussed in [Pollack et al., 1996].

3.4 Discussion

We have described different models for the coevolution of two populations. Some of the fundamental impediments to coevolutionary learning have been identified along with some of the reasons why continuous progress is difficult to achieve. It is now clear that none of these approaches can address successfully the problem of coevolutionary learning alone. All the rules discovered in those experiments perform poorly since they never approach the 50% density. The following section proposes a framework to get around those problems.

Each of the canonical models discussed so far implements a single specific strategy. In the literature, there has been some successful applications for both the cooperative and the competitive approaches. However, those works usually introduce some mechanisms to address the problems specific to each model. For instance, a noisy evaluation of the fitness can force exploration in a cooperative model, and an evaluation of individuals with respect to a set of opponents extracted from previous generations can limit the cyclic behavior observed in competitive models (e.g., see the life-time fitness evaluation technique described in

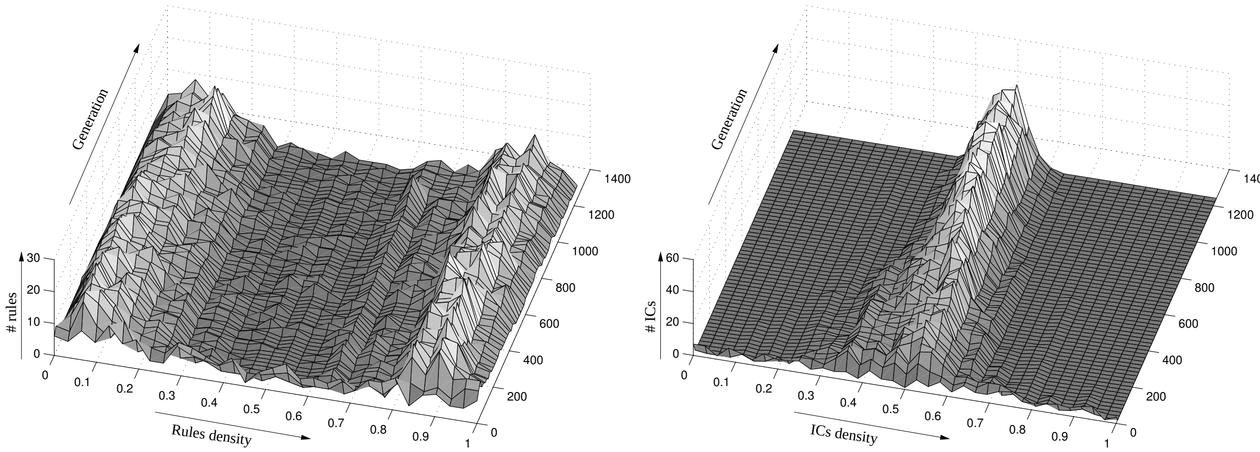

Figure 4: Coevolution of CA rules (left) and ICs (right) in a competitive relationship with resource sharing.

[Paredis, 1996] or the "hall of fame" method presented in [Rosin, 1997]). However, those mechanisms usually fail to address entirely the fundamental issues discussed previously.

4 Coevolving the "Ideal" Trainer

4.1 Presentation of our Approach

From the analysis of the experiments presented in section 3 at least two reasons seem to prevent continuous progress in coevolutionary search. The first one is that the training environment provided by the population of ICs returns little information to the population of evolving rules because a stable configuration is reached in which the credit is distributed according to a fixed pattern (e.g., all the ICs are covered by rules). The second reason is that the dynamics of the search performed by the two coevolving populations doesn't drive individuals to the domain of the state space that contains most promising solutions because there is no "high-level" strategy to play that role. This is a consequence of the Red Queen effect.

Our approach proposes a coevolutionary framework in which those two issues are addressed as follows:

- the training environment provides at any time a gradient for search by proposing a variety of problems covering a range of difficulty. Indeed, if problems are too difficult, nobody can solve them. On the contrary, if they are too easy, everybody can solve them. In both cases, those problems are useless for learning since they provide little feedback.
- a "high-level" strategy allows continuous progress by preventing the negative effects associated with the Red Queen.

The central idea of this coevolutionary learning approach 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. The difficulty resides in the accurate implementation of the concepts presented above in a search algorithm. So far, our methodology to implement such a system consists in the construction of an explicit topology over the space of problems by defining a partial order 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 task. The definition of this topology over the space of problems makes possible the implementation of the two goals required in our coevolutionary learning approach. Indeed, since learners are evaluated against a known range of difficulty for problems, it is possible to monitor their progress and to expose them to problems that are just "a little more difficult". In our work, this last concept has been formalized by defining empirically a distance measure. In this framework, learners are always exposed to a gradient for search and it is possible to control the evolution of the training environment towards more difficult problems in order to ensure continuous progress.

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. The work of Rosin [Rosin, 1997] already describes some methods to address this issue.

4.2 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 new contribution of this work is the idea of maintaining a gradient for search as one of the underlying heuristics. In the literature, different approaches have been proposed to address the issues associated with the Red Queen effect [Paredis, 1996, Rosin, 1997]. However, to our knowledge, explicit methods to force progress and to prevent mediocre stable states in the context of evolutionary search have never been tried.

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. The concept of an ideal trainer is also discussed in [Epstein, 1994] in the context of game learning. However, this work addresses the issue of designing the "ideal" training procedure which would result in high quality players rather than coevolving the training environment in response to the progress of learners.

5 Application to the Discovery of CA Rules

5.1 Experimental Setup

The approach described in the previous section 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. 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.

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

where:

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

and

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

where:

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

This definition implements the competitive relationship with resource sharing. However, a new component, namely $E(R_i, \rho(IC_j))$, has been added in the definition of the ICs' fitness. 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 should be composed of ICs that provide useful information to identify good rules from poor ones. In order to allow continuous progress, our implementation exploits an intrinsic property of the $\rho_c = 1/2$ task. Indeed, it seems that CA-rules that cover ICs with density $\rho_0 < 1/2$ (resp. $\rho_0 > 1/2$) with high performance will also be very successful over ICs with density $\rho'_0 < \rho_0$ (resp. $\rho'_0 > \rho_0$). 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 $q = 1 - p$. $E()$ implements the distance measure discussed in section 4.1. Its purpose is to maintain the balance between the search for more difficult ICs and ICs that can be solved by rules. In practice, the entropy is evaluated by performing some statistics over the population of ICs.

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

Coevolution	00010100	01011111	01000000	00000000	00010111	11111100	00000010	00010111
	00010100	01011111	00000011	00001111	00010111	11111111	11111111	11010111
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

5.2 Experimental Results

Experiments were performed with different sizes for the population of rules and ICs. The best rule whose performance is reported in table 1 resulted from the experiments that used the largest population size. In those experiments, 6 runs were performed for $5,000$ generations, using a size of $1,000$ for the two populations. 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%. All runs consistently evolved some rules that score above 82%. Table 2 describes lookup tables for the current best CA rule and other rules discussed in the literature. The leftmost bit corresponds to the result of the rule on input 0000000, the second bit corresponds to input 0000001, ... and the rightmost bit corresponds to input 1111111.

Figure 5 describes the evolution of the density of rules and ICs for one run. As rules improve, their density gets closer to $1/2$ and the density of ICs is distributed on two peaks on each side of $\rho_c = 1/2$. In that particular run, it is only after $1,300$ generations that a significant improvement is observed for rules and that, in response, 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.

6 Conclusion

This paper presents a new framework based on the concept of *coevolutionary learning*. This approach coevolves the training environment with respect to a population of learners such that learners are always exposed to a gradient for search, and evolution of problems towards increasing difficulty 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. We applied this framework to the problem of evolving CA rules for a classification task. Our experiments resulted in a new rule whose performance improves very significantly over previously known rules for that particular task.

Acknowledgment

I would like to thank Melanie Mitchell for her help and useful discussions.

References

[Andre et al., 1996] 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 of the Fifth Artificial Life Conference*, pp. 16–18.

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

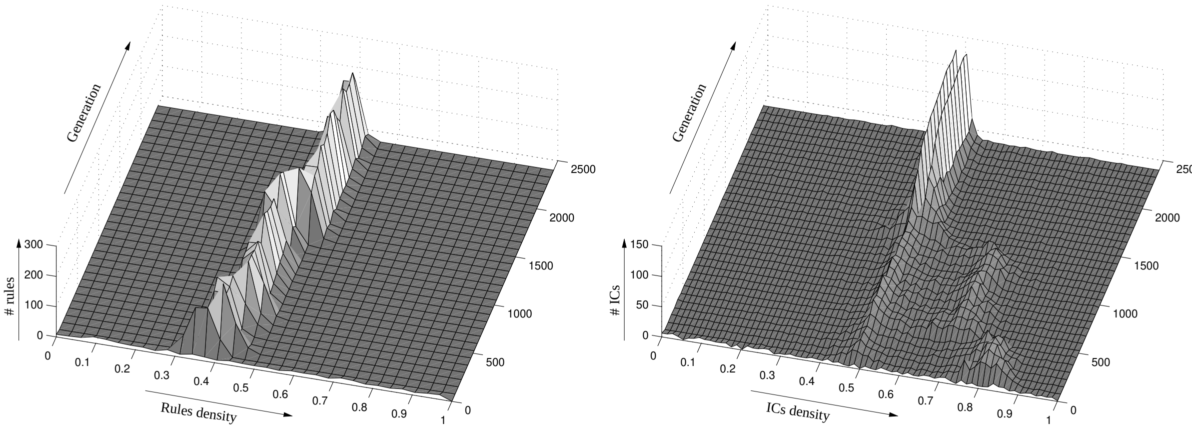

Figure 5: Coevolutionary learning between CA rules (left) and ICs (right).

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

[Das et al., 1994] Das, R., Mitchell, M., & Crutchfield, J. P. (1994). A genetic algorithm discovers particle-based computation in cellular automata. In *Parallel Problem Solving from Nature III, LNCS 866*, pp. 344–353. Springer-Verlag.

[Epstein, 1994] Epstein, S. L. (1994). Toward an ideal trainer. *Machine Learning*, 15:251–277.

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

[Juillé & Pollack, 1996] Juillé, H. & Pollack, J. B. (1996). Co-evolving intertwined spirals. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 461–468. MIT Press.

[Land & Belew, 1995] Land, M. & Belew, R. K. (1995). No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):1548–1550.

[Mahfoud, 1995] Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign. IlliGAL Report No. 95001.

[Mitchell et al., 1994] Mitchell, M., Crutchfield, J. P., & Hraber, P. T. (1994). Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391.

[Mitchell et al., 1993] Mitchell, M., Hraber, P. T., & Crutchfield, J. P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130.

[Paredis, 1996] Paredis, J. (1996). Coevolutionary computation. *Artificial Life*, 2(4).

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

[Pollack et al., 1996] Pollack, J. B., Blair, A., & Land, M. (1996). Coevolution of a backgammon player. In Langton, C. (Ed.), *Proceedings of Artificial Life V.* MIT Press.

[Rosin, 1997] Rosin, C. D. (1997). *Coevolutionary Search Among Adversaries.* PhD thesis, University of California, San Diego.

[Rosin & Belew, 1995] Rosin, C. D. & Belew, R. K. (1995). Methods for competitive co-evolution: Finding opponents worth beating. In Eshelman, L. J. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Mateo, California. Morgan Kauffmann.

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