# Evolution of Goal-Directed Behavior from Limited Information in a Complex Environment

**Matthew R. Glickman**
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
glickman@cs.cmu.edu

**Katia Sycara**
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
katia@cs.cmu.edu

## Abstract

In this paper, we apply an evolutionary algorithm to learning behavior on a novel, interesting task to explore the general issue of learning effective behaviors in a complex environment that provides only limited perception and goal-feedback. Our specific approach evolves behavior in the form Artificial Neural Networks with recurrent connections. We apply our approach to learn effective behavior for a non-standard maze-navigation problem that is characterized by aspects of problems that are difficult to approach via other methods. Difficult aspects of the specified problem include the inability to sense all task-relevant state at any given time (the problem of "hidden state"), and limited feedback with respect to success or failure. We observe evolved networks to perform very well on the target problem. Further findings include adaptation to noise in action selection, performance proportional to memory capacity, and improved performance when network weights are transferred from training on one maze to another.

## 1 Introduction

### 1.1 Evolving Behavior with Evolutionary Algorithms

As we design agents to operate in increasingly challenging environments, it becomes increasingly important to be able to learn specific behaviors rather than directly specifying them. Given a complex environment and a desired task to be achieved within it, we would like agents to be able to learn to perform the task without requiring significant pre-existing knowledge of the structure of the environment or the mechanics of the task. Moreover, we cannot necessarily depend on the agent having instantaneous access to all task-relevant environmental information, nor can we depend on the environment to provide regular, particularly helpful cues along the path to the goal.

One appealing feature of Evolutionary Algorithms (EA's) for use in such problems is how little feedback–just a rough estimate of the relative desirability of solutions–is necessary to engage the process. Evolutionary algorithms can thus be applied to problems with a search space into which we have relatively limited insight, *i.e.* problems for which any pre-existing domain knowledge offers few hints as to how a solution may be constructed.

Moreover, it is interesting to note that finding solutions to just such tasks is exactly the problem at which biological evolution has excelled. In nature, evolution has yielded agents (organisms) that pursue staggeringly complex strategies (an organism's behavior over a lifetime) to maximize a very noisy and often extremely delayed-in-time form of feedback (individuals' differing reproductive success).

We present here the results of several experiments applying an Evolutionary Algorithm (EA) to evolve effective behavior in the form of Artificial Neural Networks (ANN's) with Recurrent connections (RANN's) for a particular task in an environment such as has been described, i.e. one which does not provide the ability to sense all task-relevant state at any given time and which provides only limited feedback with respect to success or failure.

Contributing to the body of work in which RANN's and/or behaviors for such environments have been evolved, these experiments are distinguished by:

1. The use of a particularly simple method for RANN evolution, thus providing fewer algorithmic features to which the observed performance might possibly be ascribed.
2. A particularly interesting learning task (described in section 2.1) exhibiting many of the features that characterize difficult problems, and which may be seen as the composition of a large number of

sub-tasks which appear to be learned in a semi-hierarchical manner.

3. Some intriguing results that include adaptation to the level of noise in action selection and evidence of inter-task transfer of search effort.

### 1.2 Value-Function Search vs. Direct Policy-Space Search

On a general level, these experiments bear upon the issues investigated in Reinforcement Learning (RL). The RL framework addresses problems where there is continual interaction between an agent and its environment whereby the environment presents information as to its current state, and the agent makes a choice of an action to perform. After each action, the agent may receive some magnitude of reward. The goal in such tasks is to find the *action policy* which an agent can follow to maximize some function of all the reward received.

The prevailing approach of RL algorithms is to search first in the space of value functions–which map states and actions to an estimated reward–and then use the resulting function to derive the best possible policy. For a large set of RL problems there exists a suite of very effective such algorithms along with well-developed theory that provides for some specific theoretical guarantees ([Kaelbling *et al.*, 1996] provides a review). However, the situation is considerably less clear where the state of the world relevant to the task at hand is not always directly observable (problems with "hidden state"). Some innovative algorithms have been presented that have proven successful for particular problems (*e.g.* [McCallum, 1996], [Littman *et al.*, 1995], [Lin and Mitchell, 1992]), but their range of effectiveness remains to be explored.

In contrast to research usually classified as RL, evolutionary approaches typically take a more general, direct approach. Instead of first finding a value function, EA's are generally used to search directly in the space of action policies. While this approach may be seen as failing to take advantage of the specific structure of RL problems, it is not clear that learning a value function first is always better than learning an action policy directly. The relative merits of searching with a value function versus searching directly in policy space is currently an interesting and important line of inquiry the field of RL [Moriarty *et al.*, 1997].

### 1.3 Evolution and the Representation of Behavior

There exist many approaches to learning behaviors using Evolutionary Algorithms. A principal difference between these approaches is the specific representation of behaviors. Wilson's work on "Animats" (a recent example is [Wilson, 1998]) employs Classifier Systems [Holland, 1975] to represent behavior. To deal specifically with problems of hidden state, Teller [Teller, 1994] extended Genetic Programming [Koza, 1992] with indexed memory and applied this technique to solving a novel problem involving moving boxes in a 2D world. Studies–such as the one reported here–that evolve RANN's include [Angeline *et al.*, 1994] and [Jefferson *et al.*, 1992].

ANN's are an interesting representation for behaviors for a number of reasons, including their inherent parallelism as well as their relative novelty as a substrate for computation. Another interesting property of ANN's in general is that their performance is often found to "gracefully degrade" instead of failing catastrophically as operational conditions stray from optimal. This property is of particular interest for EA's as it potentially enhances the smoothness of the adaptive landscape.

Recurrent connections provide the ability to retain state over time (necessary to disambiguate hidden state). At the same time, a cost of recurrency is that it greatly complexifies the relationship between the set of connection weights in the network and the expressed behavior. Moreover, because the feedback provided by the environment in problems such as the one explored here doesn't provide explicit target outputs, it's difficult to employ network training methods which are based upon the backward propagation of error. Both the increase in complexity due to recurrent connections and the absence of explicit output targets contribute to the attraction of using evolutionary methods.

Given the variety of possible representations and search operators for evolving RANN's as well as their potential importance the specific dynamics of search, implementation choices are difficult to make. In order to minimize this difficulty, we've chosen to employ a particularly simple approach (as described in sections 2.2 and 2.3).

In the next section, we describe the task, the layout and evaluation of the RANN's, and the specific evolutionary algorithm. Section 3 then presents the results of several different experiments. We conclude in section 4 by reviewing implications of our observations.

## 2 Methodology

### 2.1 The Start-Anywhere Maze Task

The task explored here involves navigation in a given maze (see figure 1). In many maze problems studied both in computer science and in animal behavior, the agent starts at the designated start point with the goal of finding a path (or the most efficient path) to the designated goal point. In the task studied here, an agent is rewarded according to how quickly it can find its way to the designated goal when started from *any*

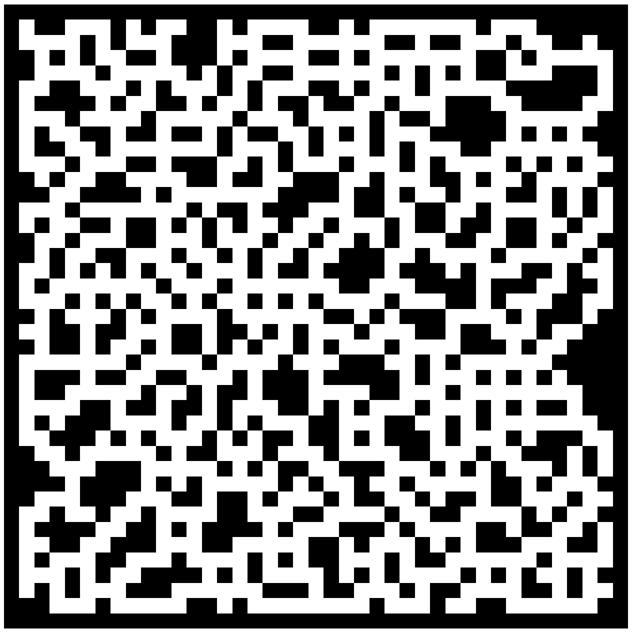
Figure 1: A typical 40x40 maze (the "goal" is in the upper, left-hand corner)

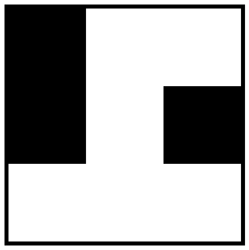
Figure 2: Field-of-view of an agent in the maze (note - orientation is significant)

random point in the maze. Agents are thus trained to solve not just one, but a composite of multiple, related single-start-point tasks.

Significantly compounding the difficulty of this task is the fact that at any given time-step, the agent only "sees" the eight squares immediately surrounding its position (see figure 2). When an agent is started from a given point, it has no *a priori* knowledge of its location. Because this immediate sensory information alone is very unlikely to identify exactly where the agent is in the maze, the agent must somehow acquire and retain information over time in order to effectively choose its actions. Given the clear lack of information at the start, the agent must begin by basically exploring, and progressively integrate observations to narrow in on a strategy to quickly and reliably reach the goal.

An agent in the maze not only has a location, but an orientation (*i.e.* at any given point in time, it is facing up, down, left, or right), and chooses from one of three possible actions: advance one step forward, or turn either left or right and then advance one step. With different orientations, the same location can appear up to four different ways in the sensory field, a factor which can further decrease the utility of immediate sensory information alone for position determination.

Another difficult aspect is that the agent only receives a reward if and when it reaches the goal; if, during a single trial, one agent manages to come within one step of the goal without actually reaching it while another never even gets close, the two receive equally poor scores. Such *delayed reward* thus significantly limits the information available to the selection process for differentiating between alternate behavioral policies.

## 2.2 The RANN's

Other studies evolving RANN's (*e.g.* [Angeline *et al.*, 1994]) have explored the intriguing potential of EA's to search the space of network topologies concurrently with that of connection weights. How to encode and modify network topologies is an open question with a great number of possible and proposed schemes. For reasons of simplicity, we chose a different approach: Simply choose a single, very powerful topology, and explore only in connection weight-space. This arrangement yields a fixed, finite-dimensional search space.

Individuals in the population (of size 100) are encoded as vectors of real-values each corresponding to a connection weight in a network of sigmoid units with a fixed, recurrent topology (an Elman [Elman, 1990] network), consisting of an input layer (9 inputs, 1 for each of the pixel in the 3x3 visual field), fully connected to a single hidden layer, which is fully connected to both the single output unit *and to itself* (see figure 3). The recurrent connections in the hidden layer are effectively time-delayed by one time-step: All of the hidden units' outputs are determined in parallel, and thus

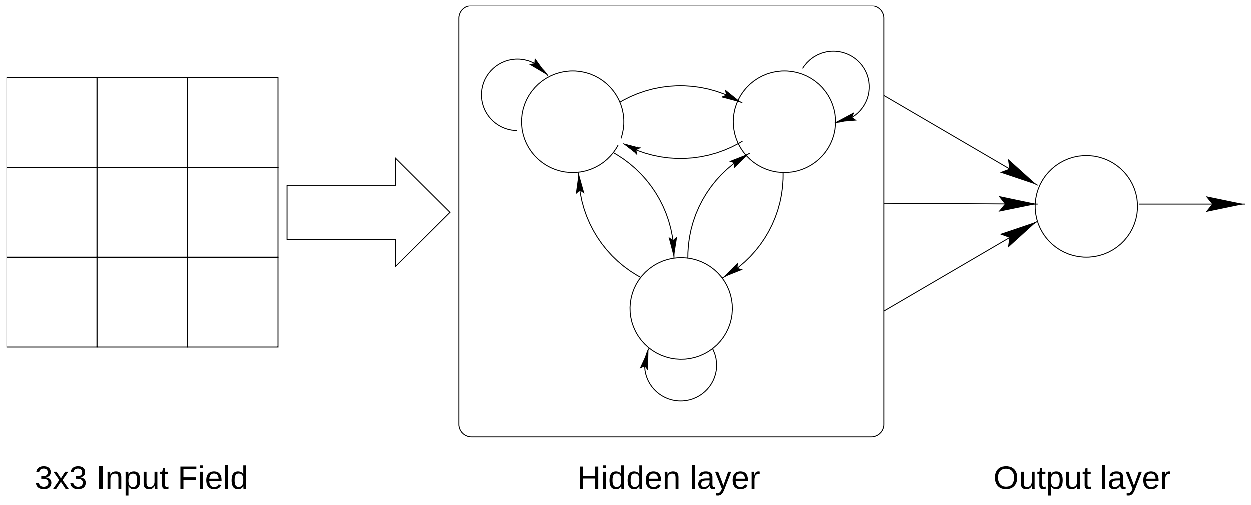

Figure 3: An Elman Network, consisting of a single hidden layer that is completely connected to its inputs (the 3x3 visual field), the output layer (a single unit), and itself

they each see each others' output values from the previous time-step when calculating their current output value. By granting each hidden unit access to the activation levels of all the hidden units in the previous time-step, recurrent connections provide the capacity to retain state over time. What information is retained and how it is encoded is left to be determined via evolution.

If the value produced by the single output unit is below a particular threshold, the agent turns left and tries to advance one grid-unit, if above a different threshold, to the right and forward one unit, and if between these two thresholds, straight ahead one unit. Unless otherwise noted, results presented here are for agents with a total of 12 hidden units, which yields a complete weight vector of 277 weights in length (including bias unit connections).

## 2.3 Evolution

An EA consists of two primary processes, selection and variation, that are alternately performed on a population of candidate solutions. To select the more fit solutions, their quality must be somehow measured. We accomplish this by dropping each member of the population into the maze at each of a set of starting points and is provided a fixed amount of time in which to reach the goal. Each such "opportunity to reach the goal" is known as a *trial*. For each generation, each individual's fitness is calculated by summing up, for each trial, the square of the number of time-steps spent before reaching the goal. The set of starting points is randomly resampled from a uniform distribution over all starting points possible in the given maze. For $25\times25$ mazes, we used trials of length 400, and length 1000 for 40x40's. 10 starting points were selected at random for each generation, and hence each individual was scored on the basis of 10 trials.

The higher its score, the less efficiently the given agent has managed to arrive at the goal. We often thus refer to fitness scores as *error*, which emphasizes that it is a quality we seek to minimize.

The specific form of selection we've used is *tournament selection*. Given a population of size $n$ and user-defined parameter, $k$ ($n = 100$ and $k = 10$ for our purposes here), $n$ "tournaments" are conducted whereby $k$ individuals are randomly chosen from the current population and the one with the best fitness score, the winner, is copied into the next population. In the manner often favored by EA's such as Evolution Strategies (ES [Schwefel, 1981]), and Evolutionary Programming (EP [Fogel *et al.*, 1966]) the variation process then simply mutates each new population member via the addition of Gaussian noise (sampled from a distribution of a fixed width defined by the user–0.2 in this case) to each weight. Again for simplicity, the EA used here does not employ any form of crossover operator.

## 2.4 Experiments

Many properties–such as the mean shortest-path-length to the goal over all potential starting points, the total number of potential starting points, the distribution of visible states along possible paths, *etc.*–vary from maze to maze of the same size. Consequently, what constitutes a "good" score for an agent may vary between mazes as well. To evade results biased for a particular maze, we've run experiments over multiple, differing mazes and summarized the results. However, instead of conducting each and every run on a different maze, we selected a fixed set of randomly-chosen mazes (at least five) and then conducted multiple runs

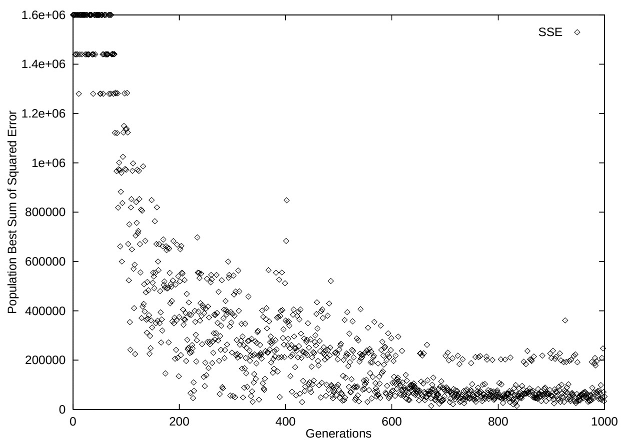

Figure 4: The sum of squared error of the population's best network over time for a typical run for a 25**x**25 maze

on each maze in order to promote faster convergence of performance measures.

Within each graph presented in this paper, the same maze-sets were always used for each group of runs presented so as to provide a valid comparison. However, although new maze-sets were not generated for every new experiment, the size and composition of the maze-sets may vary between graphs, so caution should be employed in comparing performance between graphs.

## 3 Results

### 3.1 Performance

The fitness, or sum-of-squared error, of the best network in the population over time for a typical run on a 25**x**25 maze is shown in figure 4. Note that scores tend to be clustered around particular fitness "levels". These levels correspond to the number of trials during which agents completely failed to find the goal, and thus incurred a score of 400 (the maximum amount of time per trial) squared. For ten trials, then, the maximal (worst) score for a given network is thus $400^2 \times 10 = 1.6 \times 10^6$.

The variance in best performance from generation to generation is not only due both to the nature of stochastic search as well as to non-elitist selection (*i.e.* that there is no mechanism for explicitly retaining in the population the best network yet identified). Variation in performance is also expected because different networks are tested from a different set of starting points each generation; although a given network may have scored well on the 10 starting points from which it was tested in one generation, the same network may possibly be found to perform quite poorly starting from the next 10 points randomly chosen for the following generation.

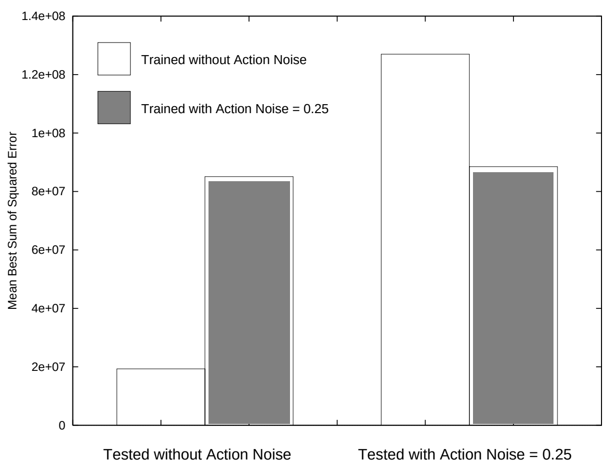

Figure 6: Mean best error for networks trained and tested with and without probabilistic action noise

### 3.2 Stages of Maze Learning

The initial networks (with randomly-selected, near zero weights) are unlikely to arrive at the goal no matter which point they start from (other than the goal itself). Occasional networks may simply move forward as long as they can, and thus be capable of reaching the goal when started from one of the few points directly adjacent.

Not surprisingly then, in general networks first master starting points near to the goal, and then proceed to master points that are increasingly farther away, as depicted in figure 5.

Note however that (1) starting points may be mastered by the best network found at one point in time and then prove difficult for successive "best" networks, and that (2) just because a point lies on the path from some starting point to the goal for a given network does not necessarily imply that it too is a starting point from which the network can reach the goal. With respect to the second point, consider for example the lower-right starting point in the third stage of figure 5. Tracing any possible path from this point to the goal, we can see that one or more points which don't serve as successful starting points must be passed through along the successful path.

### 3.3 Adaptation to the level of action noise

To test the robustness of learning when actions don't always have the intended effect, networks were trained in an environment where the action performed has a non-zero probability of being an action different from the one the agent selected. Networks trained both with and without action noise were then tested in environments both with and without action noise (see figure 6).

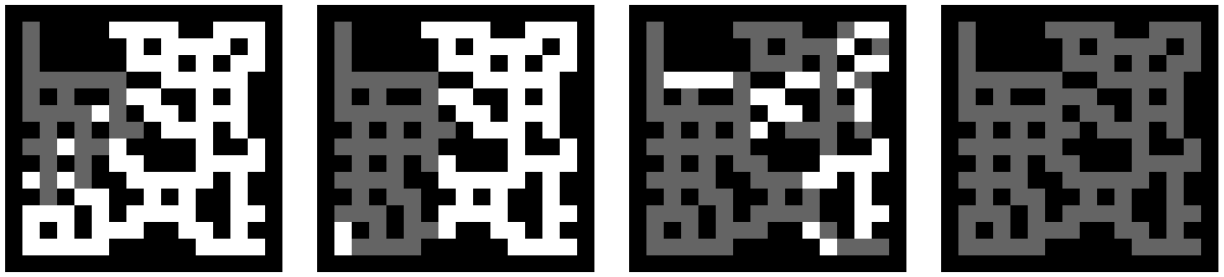

Figure 5: Progressive stages of "maze coverage" over time (running left to right) during training on a simple, 15x15 maze. Gray squares indicate points from which the best network yet found can find its way to the goal (the upper-left corner) in the allotted time.

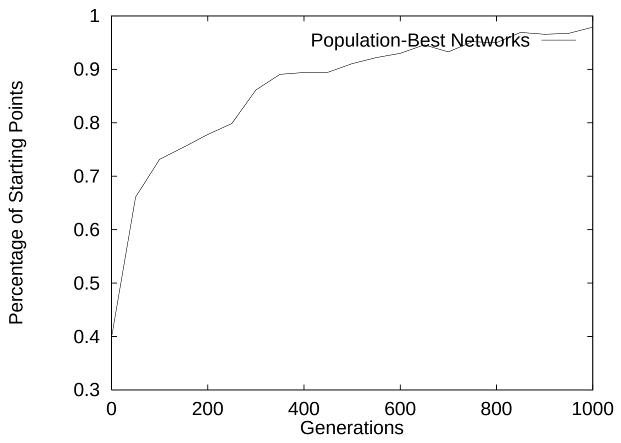

Figure 7: Mean percentage of starting points over time from which the population best networks managed reach the goal trained and tested with action noise probability = 0.25 (trial length 800)

Somewhat predictably, we found that networks trained without any action noise performed rather poorly in noisy environments. On the other hand, networks trained in the presence of action noise seem to adopt strategies which are less efficient overall, but significantly more robust to noise. We thought perhaps that this had to do with networks "giving up" on particularly hard starting points to arrive at simpler but less-optimal stategies, but further experimentation suggests that this is not in fact the case (see figure 7).

## 3.4 Use of Memory

To determine how well the capacity of the networks' recurrent links was being exploited, we conducted experiments with two different forms of "impaired" networks. The first form simply had no recurrent links, providing it with no capacity to retain state over time (and hence in all likelihood unable to represent any policy that succeeds at reaching the goal from at or near most of the possible starting points). The second

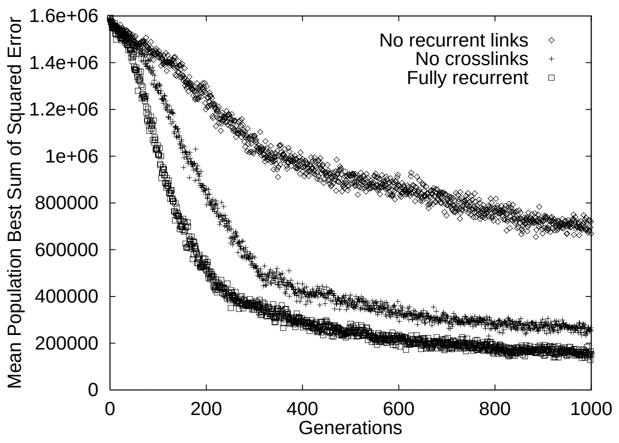

Figure 8: Mean population-best sum-of-squared-error for networks with differing numbers of recurrent links

form has no cross-links between *different* hidden units, the only remaining recurrent links being self-links from hidden-units *to themselves*.

General performance over training of the two impaired forms on 25x25 mazes in comparison to fully recurrent networks is depicted in figure 8. In general, expected performance was found to decrease with the the number of recurrent links removed. This result indicates that the evolved networks are quite naturally exploiting the richer behavioral capacity afforded them by recurrency.

## 3.5 Inter-Task Transfer

An important goal of our research is to find methods whereby an EA solving a series of related problems can benefit from the cumulative computational effort expended to find solutions more quickly and/or of higher quality. Such a system can be said to "transfer" acquired knowledge of the problem domain accumulated during search from one or more tasks to the next.

Results from a quite direct form of inter-task transfer

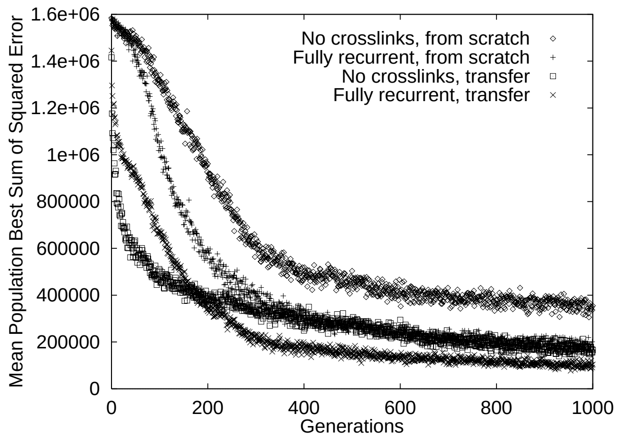

Figure 9: Training performance over time for networks with and without recurrent cross-links, started both from scratch and from a previously trained network

are shown in figure 9. The best score in the population is plotted over time for evolving RANN's to navigate 25**x**25 mazes averaged over several different runs, where networks are either fully recurrent, or without recurrent cross-links (as described in section 3.4). The initial runs were started with weight vectors initialized to near-zero random values (referred to as "from scratch"). In the second runs, the initial weight vectors were instead all initialized by copying the weight vector of *one* of the best networks resulting from the initial runs. Networks trained to navigate one maze in the initial runs were thus trained a second time on a different maze.

By using the best individual from the first runs, the second runs start out more fit and then adapt more quickly, *and* end up at a better fitness level than the first runs. We can see that when the impaired "no cross-links" networks are trained by transferring a weight vector from one of the initial runs, their performance improves significantly faster than and actually achieves the same quality of fitness as the fully recurrent networks when trained from scratch. The fully recurrent networks, too, benefit from this weight-transfer, and end up with higher fitness scores than any of the others.

We also have evidence of still more performance benefits, particularly with respect to speed, resulting from a "third stage" transfer, *i.e.* seeding a set of third runs with weight vectors from one of the best vectors derived from the second set of runs (the first transfer runs). Unfortunately, the network performance is good enough on this task that it's difficult to tell whether this process actually could continue. We are currently exploring both (1) tasks in which there is clear potential for further performance improvements via repeated use of this transfer mechanism, as well as (2) what it is about the previously trained weight-vectors that provides the observed performance benefits.

## 4 Summary

The principal finding of this paper is that the approach to learning goal-directed behavior described above produces viable solutions to the given problem, which is characterized by a number of features that mark particularly difficult RL tasks, including significant relevant hidden state and delayed reward.

Success of this approach attests to a number of hypotheses. First is to the general viability of Evolutionary Algorithms when applied to difficult RL tasks. On the more specific topic of EA search operators, note that in contrast to many other EA/RL applications, the EA employed here relies solely on a Gaussian-based mutation operator.

It is also interesting to observe that given the structure of this particular task, learning seems to proceed in a roughly "hierarchical" manner, in that smaller sub-tasks (reaching the goal from relatively close starting points) are *generally* learned as prerequisites to learning more complex sub-tasks (reaching the goal from starting points that lie further out).

Another hypothesis–that evolution is an effective search procedure for the weight space of RANN's–is supported by virtue of the quality of solutions found on this task. More specifically, evidence for the use of memory capacity further demonstrates that ability of an EA to effectively exploit the benefits of recurrent links.

A related, supported conjecture is that RANN's provide an effective representation for action policies. Moreover, RANN's can represent policies that are effective in the presence of action noise. At a more general level, these experiments display a challenging domain in which direct search in the space of action policies produces effective solutions.

Finally, the performance benefits observed from the inter-task transfer of previously trained weight-vectors point toward a quite direct method to derive benefit from previously expended computational effort for solving related problems.

Our continuing work includes exploring the efficacy of evolving RANN's on both different and larger-scale problems, as well as comparison with other learning methods that have been developed for dealing with hidden state. A particular interest is the reason behind and the potential for improved performance from the inter-task transfer of connection weights: Is there some kind of general maze-navigation ability that's being transferred, or are the results due to some kind of lower-level phenomenon in weight space? We are actively investigating these issues.

**Acknowledgements**

The authors would like to thank James Thomas for recurrent fruitful discussions, Karthik Balakrishnan for helpful comments and suggestions, and four anonymous reviewers for their thoughtful feedback. This research was partially supported by the Office of Naval Research, Contract N00014-95-1-0591. Matthew Glickman was partially supported by an NSF Graduate Fellowship.

## References

[Angeline *et al.*, 1994] Peter J. Angeline, Gregory M. Saunders, and Jordan B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, 1994. Special Issue on Evolutionary Computation.

[Elman, 1990] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[Fogel *et al.*, 1966] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley & Sons, New York, 1966.

[Holland, 1975] John H. Holland. *Evolution in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[Jefferson *et al.*, 1992] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. Evolution as a theme in artificial life: The genesys / tracker system. In [Langton *et al.*, 1991], pages 549–578, 1992.

[Kaelbling *et al.*, 1996] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Koza, 1992] John R. Koza. *Genetic Programming: On the Programming of Computers by the Means of Natural Selection.* MIT Press, 1992.

[Langton *et al.*, 1991] C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors. *Artificial Life II.* Addison Wesley, 1991.

[Lin and Mitchell, 1992] Long-Ji Lin and Tom M. Mitchell. Memory approaches to reinforcement learning in non-markovian domains. Technical Report CMU-CS-92-138, Computer Science Department, Carnegie Mellon University, 1992.

[Littman *et al.*, 1995] M. L. Littman, A. R. Cassandra, and L. P. Kaebling. Learning policies for partially observable environments: Scaling up. In *The Proceedings of the Twelfth International Machine Learning Conference.* Morgan Kauffman Publishers, Inc., 1995.

[McCallum, 1996] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State.* PhD thesis, University of Rochester, 1996.

[Moriarty *et al.*, 1997] David Moriarty, Alan Schultz, and John Grefenstette. Reinforcement learning through evolutionary computation. Technical Report AIC-97-015, NCRAI, 1997.

[Schwefel, 1981] H. P. Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[Teller, 1994] Astro Teller. The evolution of mental models. In K. Kinnear, editor, *Advances in Genetic Programming.* MIT Press, 1994.

[Wilson, 1998] S.W. Wilson. Generalization in the xcs classifier system. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, San Francisco, CA, 1998. Morgan Kaufmann.