Neural Processing Letters **11:** 29–37, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Online Interactive Neuro-evolution

A. AGOGINO, K. STANLEY and R. MIIKKULAINEN
Dept. of Computer Science, The University of Texas at Austin, TAY 2.124, C0500, Austin, TX 78712–1188, U.S.A., E-mail: kstanley@cs.utexas.edu

Abstract. In standard neuro-evolution, a population of networks is evolved in a task, and the network that best solves the task is found. This network is then fixed and used to solve future instances of the problem. Networks evolved in this way do not handle real-time interaction very well. It is hard to evolve a solution ahead of time that can cope effectively with all the possible environments that might arise in the future and with all the possible ways someone may interact with it. This paper proposes evolving feedforward neural networks online to create agents that improve their performance through real-time interaction. This approach is demonstrated in a game world where neural-network-controlled individuals play against humans. Through evolution, these individuals learn to react to varying opponents while appropriately taking into account conflicting goals. After initial evaluation offline, the population is allowed to evolve online, and its performance improves considerably. The population not only adapts to novel situations brought about by changing strategies in the opponent and the game layout, but it also improves its performance in situations that it has already seen in offline training. This paper will describe an implementation of online evolution and shows that it is a practical method that exceeds the performance of offline evolution alone.

Key words: evolution, online, game, neural, network, genetic, real-time

1. Introduction

Genetic algorithms with neural networks are a powerful combination that has been successfully employed in many application domains in the past. For example, a variety of neuro-evolution methods have been applied to board games such as Othello, Go, and Backgammon [3, 7, 9, 10]. Also, evolution has been successful in stochastic, dynamic tasks such as foraging, herding, communication, and prey capture [2, 7, 11].

In all of this previous work, populations are evolved off-line. At each generation, individuals play a round of the game first, and are then evaluated. The next generation is created based on those individuals that did well over the course of their round of play. After many rounds and many evaluations, a few proficient individuals should emerge.

Although offline neuro-evolution has proven to be useful in board games and dynamic environments, there has been little work applying it to real-time interactive environments. The main problem is that in these domains, good performance requires adapting to the opponents and the changing environmental conditions online. For example, tactical units that evolve online would make military simulations of rapidly changing, unpredictable environments more realistic. Robotic controls could be evolved online as a robot attempts to adapt to a new environment or when it has to cope with sudden problems such as failing sensors. One excellent example of an online domain is real-time gaming. Here, the opponents are constantly changing their strategies and the environment presents new challenges all the time. The population has to adapt during a round a play, with constant evaluation and change.

Very few systems have been built where evolution takes place as the system is performing. One domain where it has been tried was to help a robot navigate a maze [1]. In this case a population of networks was evolved on a single robot as it tried to navigate through a maze it had not seen before. While the robot could effectively learn to navigate through the maze, it took a very long time to do so. This would make it unacceptable in many domains such as real time gaming where the algorithm has to adapt to a particular opponent in a real time. Other experiments in robot control evolution have taken place offline [6, 8].

In this paper, we will use the paradigm of a real-time game world as a platform for developing methods for online evolution. We will demonstrate that a population can evolve online by keeping a ranked order of its individuals, and periodically replacing the lower ranked individuals with offspring of their higher ranked peers. A fitness evaluation function that applies in real-time is used to keep the ranking up to date. Below we will first describe the gaming scenario and the evolution methods, followed by a detailed comparison of offline evolution with online evolution.

2. The Game

To develop an on-line evolution algorithm for interactive environments, we implemented a small game inspired by the popular PC game WarCraft II by Blizzard Entertainment. Our game contains two characters: the human-controlled enemy and the computer-controlled peon. The game consists of a planet, a base, gold mines, the enemy, and a population of peons (Figure 1A). There are 30 peons, which all start out at the solitary base. The objective of the peons is to find one of the gold mines as fast as possible without getting killed by the enemy, whose location is controlled by the human player. If the enemy comes into contact with the peon, the peon dies. Once the peon finds a mine, it will be immediately transported back to base, ready to start a new journey.

In the PC game, the peons are controlled by simple algorithms that are easy to defeat, making them unexciting. They tend to head towards the closest mine, whether it is safe to do so or not. In our version of the game, neural nets control these peons so that through evolution they become more sophisticated in their ability to find mines and respond to enemies. They can evaluate the risk between going towards a closely guarded mine and an unguarded one farther away and develop strategies to avoid enemies (Figure 1B).

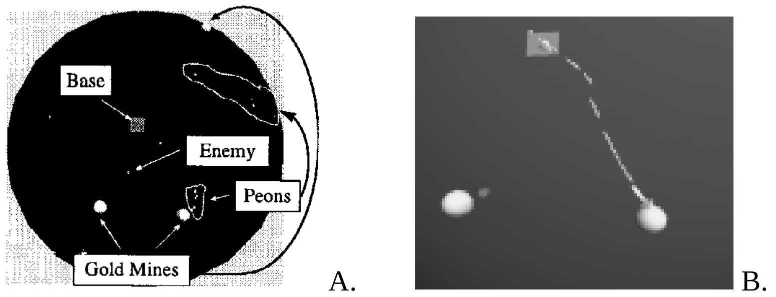


Figure 1. A. Configuration of the game. Peons start from the base and try to find gold mines while avoiding the enemy. B. Example of the beginnings of intelligent behavior: Peons move to the mine the enemy is not guarding even though it is slightly further away. Later on peons will show that they can handle more complex behavior. A demo of this process is available at http://www.cs.utexas.edu/users/nn/pages/research/neuroevolution.html.

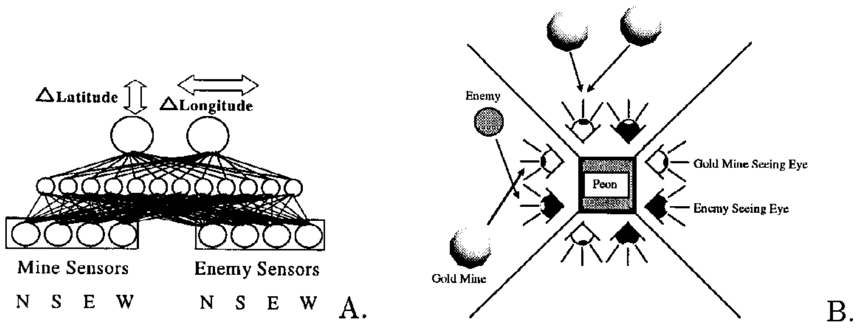


Figure 2. A. Peon's neural net with inputs and outputs. The sensor information is sent to the input layer of the feedforward network. The two output nodes indicate where the peon should go in terms of latitude and longitude distance from the current location. B. Configuration of a peon's eyes. Four of the eyes return the average distances to gold mines in each quadrant and the other four eyes return the average distance of the enemy.

3. Network Architecture

Each peon is endowed with a feedforward neural network that it uses to decide what to do at each time step of the game (Figure 2A). The networks are not recurrent, which means individual peons have no memory of prior actions or senses. The peon receives 8 sensor readings as its input and generates an output that indicates how far it wants to move in latitude and longitude. The sensor data comes from a configuration of 8 eyes that can sense the distance to gold mines and enemies in the entire world (Figure 2B). The sensory inputs are not precise in that the peon is not aware of the actual angles of objects in its sensors. Rather, it can only make decisions based upon which quadrant the object is in.

It is important to note that other approaches to this problem would be possible. For example, reinforcement learning such as Q-learning or Temporal Difference learning could be used. However, a number of studies have shown that evolution is more powerful than reinforcement learning in similar domains, especially those that are continuous and contain hidden state information [4, 5]. Our goal is to extend this power to online systems, and to show that in such domains, online evolution is more effective than offline.

4. Online Evolution Algorithm

In the interactive real-time environment, where numerous peons are continuously being born and killed, evolution is a natural method for learning. When a peon dies, it is replaced with either a mutation of a highly ranked peon or the result of a mating of two highly ranked peons (Figure 3A). The mutation is performed by applying Gaussian noise to the weights, and the mating algorithm is based on single-point crossover. The proportion of peons born through mutation vs. mating has little effect on experimental results (%50/%50 was used in experiments reported here). Also the exact mating algorithm was not found to be critical.

The peons are ranked based on their rate of productivity using the following formula:

$$\text{Fitness} = \frac{\text{Mines Found} \times V - C}{\text{Age}}$$

Each peon is awarded V units of gold for each mine found, but C units are subtracted for its initial cost of being born. This measure tends to reward finding mines quickly, but also awards longevity, because the initial cost of the older peons is amortized over more time. In this paper V is set to 100 and C is set to 1000. With these values, the longevity of a peon is an important criterion, and populations tend to evolve that avoid enemies, while still going after the mines.

5. Offline Evolution Algorithm

In order to evaluate the performance of online evolution, it needs to be compared with offline evolution. To make this comparison, the standard offline neuro-evolution algorithm was adapted for the task of evolving populations of peons. In principle, the offline algorithm must evolve a population that can cope with all possible situations that could arise in the game. Therefore, it needs to be evolved with a good sample of possible scenarios. In order to do this, we developed 16 different game scenarios. Each scenario consists of one of four unique mine placements with one of four different enemies. Any given population may perform well on some scenarios but usually performs poorly on others: the task of offline evolution is to develop one that would be generally proficient, i.e. able to handle any scenario.

The offline algorithm is designed to be as similar as possible to the online one. In the beginning 30 random peons are generated. Each of these peons need to be evaluated on how well it does on all 16 scenarios. Since groups effect the dynamics of the game, a single peon cannot be run through a scenario by itself. To resolve this

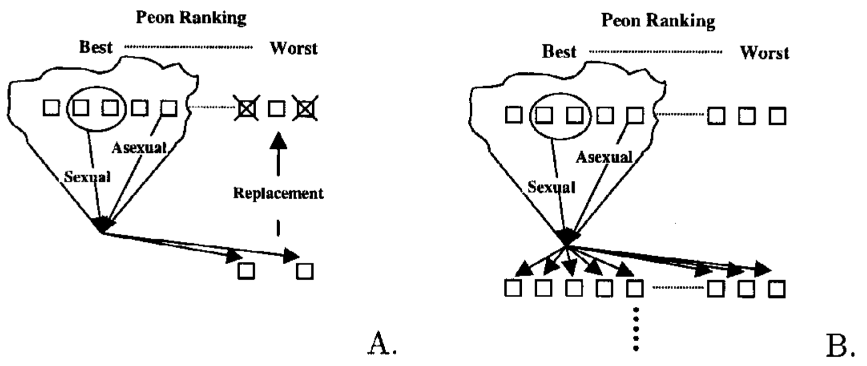


Figure 3. A. Online Evolution. When a peon dies it is replaced by either a copy of good peon or a mating of good peons. The good peons are chosen stochastically among the ones with the highest rank. B. Offline Evolution. Each peon is evaluated over all 16 scenarios. The new population is then created through copying and sexual reproduction in the best peons , which is exactly the same way new peons are chosen to create replacements for deceased members in the online population.

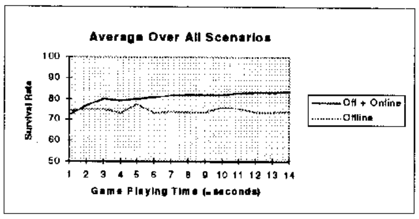
problem, thirty copies of it are made at the beginning of the scenario, and tested all at once. The performance of the original is based on how well these copies did. The copies are then thrown away at the end of each scenario. Once the performance of each peon is evaluated through all the scenarios, a new population is created from the best of the old ones, in the exact same way dead peons are replaced in the online evolutionary strategy (Figure 3B). The peons are evolved until they stop improving.

6. Results

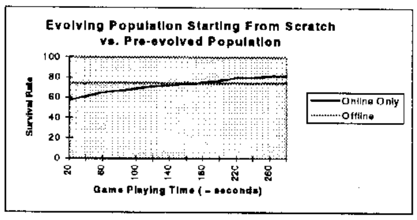
To compare the performance of offline evolution to that of online evolution, first an offline population is evolved. Two copies of it are made. One of them, labeled ‘offline’ in Figures 4 and 5, does not evolve during testing. The other one, labeled ‘off+online’, forms a starting point for online evolution. It is allowed to evolve and adapt in real-time as it plays each scenario. If online evolution is really superior, the evolving population should quickly improve its performance within the scenario.

It is important to emphasize that such improvement is by no means guaranteed. Real-time evolution could easily lead to degrading the population as well. For example, when the population is evolving against one type of enemy it might lose its ability to deal with other enemies, or it could lose most of its intelligence altogether. This problem does not affect an offline population, since it is not being evaluated on short-term performance. This paper will answer the question of whether such problems hamper online evolution to such an extent that it cannot operate.

The two populations were compared using several performance measures, which all led to similar results. The performance measure reported in this paper is average survival rate of the peons. Over time, this quantity gives us a measure of how the population improves or degrades during an individual game.



A.



B.

Figure 4. A. Average performance over all scenarios of a population that is allowed to evolve online (marked ‘Off + Online’) compared to one that is not (‘Offline’). After a few time steps the online population is able to improve its performance, while the static offline population performs approximately the same throughout the game (each time step is equal to 250 passes through the main loop of the game program: the differences are statistically significant starting from the third time step). B. A population started with random weights that evolves online will outperform the population trained offline when given enough time. The differences are statistically significant within the time intervals [0,120] and [220,280].

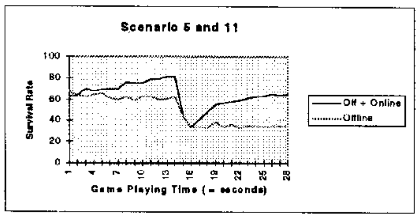
6.1. AVERAGE PERFORMANCE

The average performance of offline and off+online populations is graphed in Figure 4A. The performance was measured every 250 simulation time steps, or approximately every second of real game playing time. These measurements were averaged over 100 games played in each of the 16 scenarios. The whole experiment was repeated 4 times with different offline populations evolved with different random number seeds and similar results were found in each case. The results are clear: Online evolution significantly improves game playing performance. The differences were shown to be statistically significant with 99% certainty by the third measurement point. Since the offline population had seen all these scenarios during evolution, this result shows that online evolution improves performance even in environments for which the offline population was optimized.

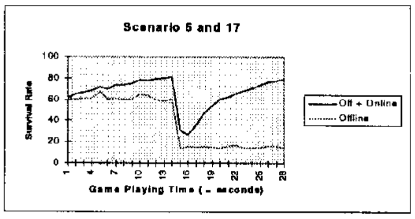
An important question is, is it necessary to have the offline evolution as the starting point? It turns out that even if the online population is started from with random weights, it eventually evolves to outperform the offline population on any given scenario (Figure 4B). This result shows that online evolution can be used in domains where it is hard to train a population offline.

6.2. GENERALIZING TO NEW SCENARIOS AND ENVIRONMENTAL TRANSITIONS

To test the performance of online and offline evolution on novel situations, Scenario 17 was created. This scenario consisted of a unique placement of mines and an enemy that behaved differently from training. After being switched to this new scenario the offline population is devastated: it is no longer evolving, and the methods it has learned through prior evolution are better suited to the scenarios it has already seen. On the other hand, the online population is able to quickly adapt



A.



B.

Figure 5. A. Even after the population has adapted to Scenario 5, it has no trouble adapting to a sudden change to Scenario 11. The graph is an average of 100 runs. The differences are statistically significant within the time intervals [6, 14] and [17, 28]. B. The improvement is even more dramatic when the new scenario is the novel Scenario 17. This graph presents an average of 100 runs, and the differences are statistically significant starting from time 6.

(Figure 5B). This result shows that online evolution is very powerful in adapting to novel environments. This is an important advantage in real world applications, where there will always be surprises. Offline populations will not be able to generalize well enough to deal with them, but online evolution can adapt to them as they occur. This result also shows that the online population can react to sudden environmental transitions. Even though it adapted well to the previous scenario, it can still quickly adapt to the new one, whether this new scenario was part of the offline training (Figure 5A) or not (Figure 5B). Since sudden environmental transitions are common in many domains, this result shows that online evolution is a robust approach to real world tasks.

7. Discussion and Future Work

The results show that online evolution performs significantly better than offline evolution in interactive real-time domains. If environmental factors are predictable, offline evolution is a reasonable strategy, although online evolution can still increase performance even in these cases. It quickly perfects strategies in scenarios where the population is not completely proficient. In real life, however, future scenarios are not predictable and novel scenarios constantly come up. The ability to react to novel situations is an important part of intelligent behavior. Online evolution displays precisely this ability to acclimate to new situations, as the analysis of Scenario 17 shows (Figure 5B). This property of online evolution is a key result of this research. Online evolution is an extremely useful tool for coping with unpredictable situations.

Great potential exists for future systems that take advantage of online evolution. With the increasing power of microprocessors and graphical engines, complex real-time simulations and virtual environments are becoming feasible. Online evolution can be used in these new environments to interact with the people using them. Some of the results presented in this paper could be of immediate use to the gaming industry. The technique could in principle be applied to any live gaming population.

Since many games use algorithms that do not adapt, real-time evolution could make these games more interesting. A great deal of research is possible in other domains as well. Online evolution could be tested immediately in domains such as robot control, traffic management and military applications. For example, robot control systems could be evolved through online evolution. This online evolution could also be applied to novel domains that were not considered suitable for evolution in the past. For example search engines could evolve online by spawning multiple agents and letting them compete. Similarly, computer virus extermination systems could be evolved to deal with new viruses. The technique is very general and could allow many domains to benefit from learning online.

8. Conclusion

This paper presents an approach to online evolution and validates it by strong experimental results on a real-time gaming task. Online evolution exceeds the performance of offline evolution in numerous tests. By adapting to game scenarios as they come up, the population is able to improve performance even on those scenarios that the offline population was evolved to optimize. The performance difference is even more dramatic with novel scenarios, allowing it to deal with the unpredictability of the real world. Online evolution is therefore a promising new approach to real-time adaptation in general and especially in domains such as gaming, robot control and traffic management.

References

1. Floreano, D., and Mondada, F.: Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot, *Simulation of Adaptive Behavior SAB-94* (1994), 421–430.
2. Gomez, F. and Miikkulainen, R.: Incremental evolution of complex general behavior, *Adaptive Behavior* **5** (1997), 317–342.
3. Moriarty, D., and Miikkulainen, R.: Discovering complex Othello strategies through evolutionary neural networks, *Connection Science* **7**(3) (1995), 195–209.
4. Moriarty, D. E.: *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*, PhD thesis, Department of Computer Sciences, The University of Texas at Austin, Technical Report UT-AI97-257, 1997.
5. Moriarty, D. E. and Miikkulainen, R.: Efficient reinforcement learning through symbiotic evolution, In: Kaelbling, L. P. (ed.), *Recent Advances in Reinforcement Learning*, Kluwer, Dordrecht, Boston, 1996.
6. Nolfi, S. and Parisi, D.: Evolving non-trivial behaviors on real robots: An autonomous robot that picks up objects, In: *Proceedings, the Fourth Congress of the Italian Association for Artificial Intelligence*, Florence, Springer Verlag, 1995.
7. Nolfi, S., Elman, J. and Parisi, D.: Learning and evolution in neural networks, *Adaptive Behavior* **2** (1994), 5–28.
8. Nolfi, S., Floreano, D., Miglino, O. and Mondada, F.: Now to evolve autonomous robots: Different approaches in evolutionary robotics, In: *Proceedings, Artificial Life IV*, pp. 190–197, 1994.
9. Pollack, J. B., Blair, A. D. and Land, M.: Coevolution of a backgammon player, In: *Proceedings of the Fifth Artificial Life Conference*, MIT-Press, Cambridge, MA, 1996.

10. Richards, N., Moriarty, D. and Miikkulainen, R.: Evolving neural networks to play Go, *Applied Intelligence* **8** (1997), 85–96.
11. Werner, G. M. and Dyer, M. G.: Evolution of communication in artificial organisms, In: Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S. (eds.), *Artificial Life II*, Addison-Wesley, Reading, MA, pp. 659–687, 1991.