Real-time Interactive Neuro-evolution

Adrian Agogino
Dept. of Electrical and Computer Eng.
University of Texas at Austin
Austin, TX 78712
agogino@ece.utexas.edu

Kenneth Stanley
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
kstanley@cs.utexas.edu

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

Technical Report AI98-266, May 1998

Abstract

In standard neuro-evolution, a population of networks is evolved in the 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.

1 Introduction

Genetic algorithms with neural networks is 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 (Moriarty & Miikkulainen, 1995; Pollack, Blair & Land, 1996; Richards, Moriarty & Miikkulainen, 1997). Also, evolution in environments related to gaming have been successful, such as in its application to stochastic, dynamic tasks like foraging, herding, communication, and the context of prey capture (Gomez and Miikkulainen, 1997; Nolfi, Elman, Parisi, 1994; Werner & Dyer, 1990, 1993).

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, 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*. To our knowledge, no methods of online evolution have been developed so far, although there are a number of areas where online evolution would be beneficial. 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.

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

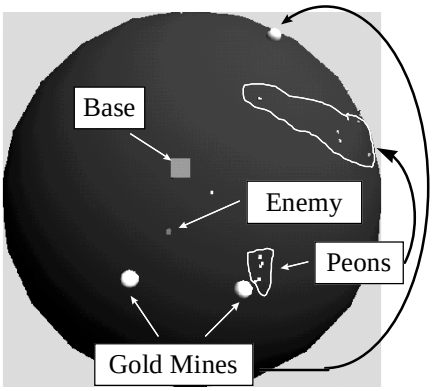


Figure 1: Configuration of the game. Peons start from the base and try to find gold mines while avoiding the enemy.

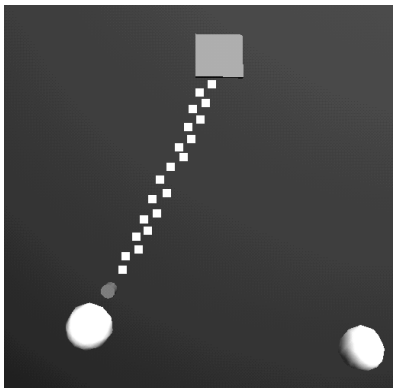


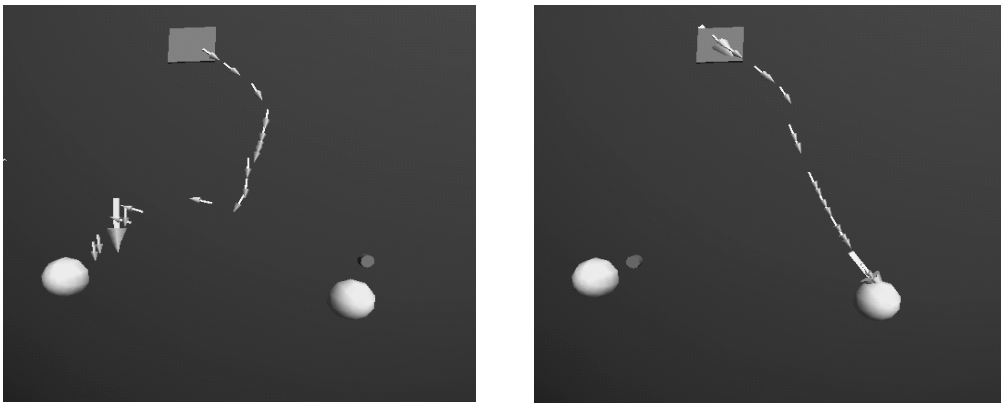
Figure 2: Group of unintelligent peons about to be slaughtered by enemy while blindly going towards mine.

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, a small game inspired by the popular PC game WarCraft II by Blizzard Entertainment was implemented. In the PC game both the computer and the human player control a wide variety of different characters that fight each other to the death. The game implemented in this paper contains only two of these characters: the human controlled enemy and the computer controlled peon. Our game consists of a planet, a base, gold mines, the enemy, and a population of peons (Figure 1). There are 30 peons, which all start out being located at the solitary base. The objective of the peons is to find one of the gold mines as fast as possible without being killed by the enemy, whose location is being 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 original PC game, the algorithms for controlling the peons are limited. One of their major weaknesses is that long strings of peons will find their way to gold mines, but once the enemy shows up at the gold mine, they will all be slaughtered (Figure 2). The peons will do very little to try to get away from the



(a) (b)

Figure 3: Example of intelligent behavior: Peons in parts a and b move to the mine the enemy isn't guarding. Note that mines never disappear or move during a scenario.

enemy and new peons emerging from their base will keep going to the same gold mine. This kind of behavior is not believable, and makes the game less interesting than it could be.

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. In contrast to the unintelligent peons, the neural-network-controlled peons should be able to go around enemies and should be able to evaluate the risk between going towards a closely guarded mine and an unguarded one farther away.

Normally, a human would control the enemy, but to make systematic evolution possible we will have the computer control it in our simulations. Since humans are not consistent, performance measures would tend to correlate with the human's own performance rather than those of the evolving population. In the simulation there are four computer-controlled enemies that are run by fixed algorithms.

In the game, the strategy of the enemy and the placement of the mines challenge the peons. Their task is difficult. They are always at risk for learning something too specific and mechanical. For instance, if the two mines are always in the same place, they may learn to move to the mines in that static location, but when the mines are suddenly moved, the peons will be lost. Similarly, the peons may learn evasion tactics for a particular enemy, but if the enemy suddenly changes its strategy, the peons will start falling into its traps. The peons should be able to adapt to such changes as they happen on-line, yet they should have enough

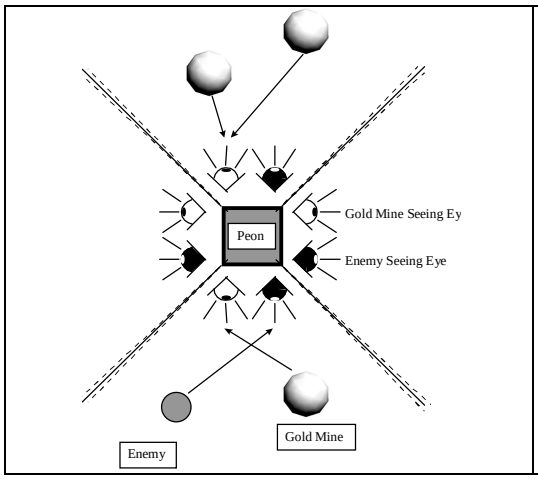


Figure 4: Configuration of a peon's eyes. Four of the eyes return the average distances to gold mines in each quadrant and the other four eyes return the average distance of the enemy.

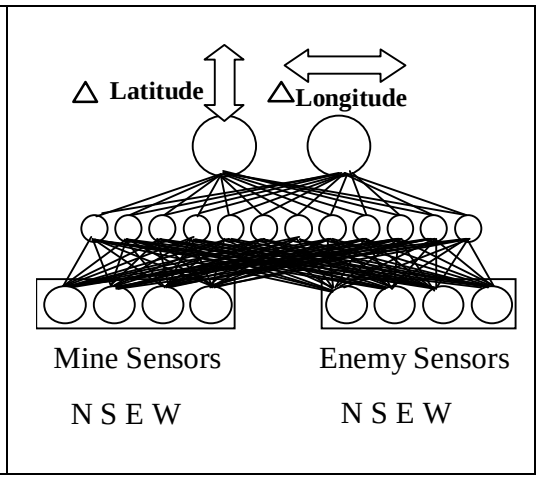


Figure 5: Peon's neural net with inputs and outputs. The sensor information is sent to the input layer of the feedforward network. The two output nodes indicate where the peon should go in terms of latitude and longitude distance from the current location.

general intelligence to handle minor environmental changes without changing their networks.

Figure 3 gives an example of the kind of intelligent behavior we are looking for. When the enemy is guarding one mine (Figure 3a), the peons move towards the opposite one, but if the enemy switches the mine it guards (Figure 3b), the peons will move to the mine the enemy has abandoned. Such behavior is a first step towards general intelligent behavior through online evolution.

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 5). 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 (Figure 4). The eyes can sense the entire world. 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. In the current implementation quadrants are allowed to slightly overlap to

help the peon avoid oscillating sensations when an object is right at the border between two quadrants.

The network uses sigmoidal activation functions, and therefore its output is bounded to [0,1]. The simulation system subtracts 0.5 from the outputs to obtain a peon movement request between –0.5 and 0.5 degrees latitude and longitude. Therefore, the peons can move in all directions, but they are limited to small, incremental movements and cannot jump directly to gold mines and over enemies.

It is important to note that other approaches to the problem presented are possible. For example, reinforcement learning is a possible alternative approach. However, our focus here is on validating online evolution as a method in its own right, and therefore the multitude of possible alternatives are not within the scope of this paper. This point aside, a number of papers have shown that evolution can be more powerful than reinforcement learning in certain domains (Moriarty, 1997, Moriarty & Miikkulainen, 1996).

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

$$\text{Fitness} = \frac{\text{Mines Found} * V - C}{\text{Age}} \qquad (1)$$

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.

In the simulations the new peon is given an 80% chance of being copied from a single highly ranked peon and a 20% chance of being a product of a mating of two highly ranked peons. The highly ranked peons are chosen by the roulette wheel method among all peons, such that those with the highest rank are chosen much more often. The mating is performed by probabilistically swapping the weights of one parent with another to produce a child, using uniform crossover. Once the new peon has been created there is a 20% chance that it will be mutated. This mutation is performed by adding Gaussian noise to 20% to its

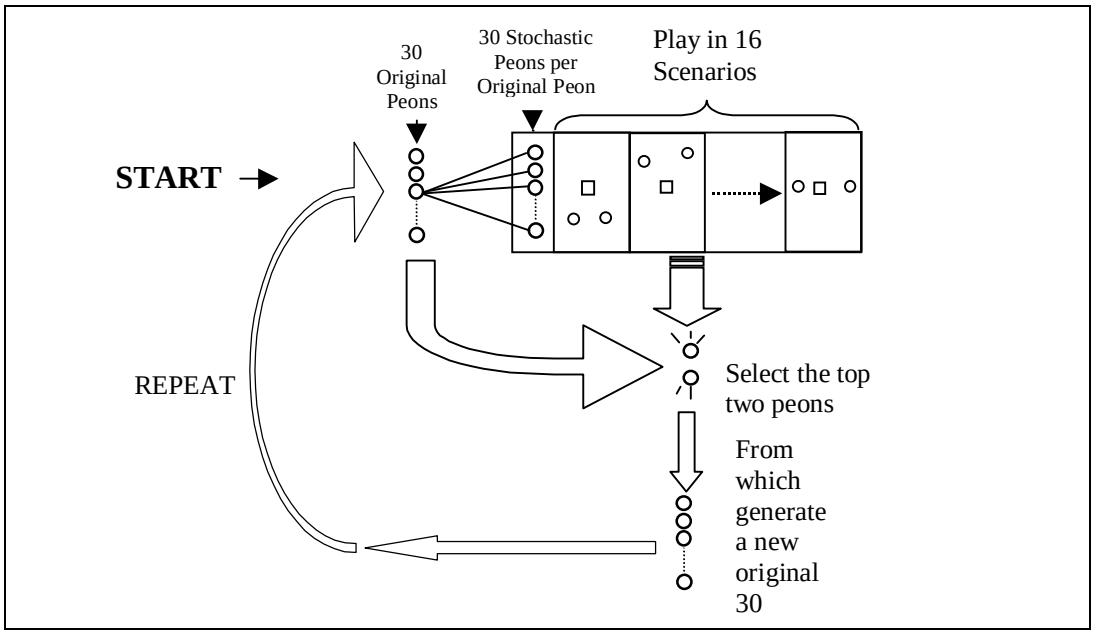


Figure 6: Offline Evolution Algorithm. Thirty Peons are evolved offline to develop an effective peon population for the game. At the beginning of each generation each of the 30 peons are placed in the game world and replicated to form a population. The peon is ranked based on how well its population did over the 16 scenarios. The scenarios themselves each last for an equivalent amount of time, and then switch to the next one. At the end of each round of scenarios the two best peons are used to replace the 28 others with the offspring of those best two. Note that the choosing of top 2 ancestor peons ties this makes this approach analogous to the online approach where the current top 2 peons are the source of offspring. Peons spend an equal amount of time in each scenario.

weights. These percentages were found by trail and error to give good results. Slight variations of these values produce roughly similar results.

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

offline algorithm is geared towards developing a population that is *generally* proficient so that it can handle any given scenario.

In the beginning, 30 random peons are generated. Stochastic mutations are performed on each of the original peons 30 times, yielding 30 populations of 30 peons each. At each generation we must find out which of the 30 populations performs the best. This question is at the heart of offline evolution: each population plays out a round, and is subsequently rated on its performance. The populations do not evolve as they play.

Each of the 30 populations is tested in all 16 scenarios, and the top two populations in terms of average fitness (equation 1) are found. These top two populations originally came from mutations of two of the original 30 peons. Those two peons are now taken and mated 14 times, which yields 14 new peons for the next generation. The top one is then mutated 14 times, adding 14 more peons to make a total of 28. Finally, the two champions themselves are added to these 28 to make a new population of 30. This new population should be better than its predecessor population since it originates from the previous population's champions. The parameter values for the offline algorithm were found to be appropriate through trial and error, but this system is not sensitive to small variations in those parameters.

At each generation of offline evolution 480 scenarios are played out, which is computationally quite expensive. We found that the progress of offline evolution tended to stagnate before 50 generations and due to the enormous computation time (more than 20 hours on an SGI O_2 for each run), the evolution was stopped at the 58^{th} generation. The best population found by that point was used for the comparisons.

Since the offline populations saw all the scenarios an equal number of times, the final populations are highly generalized. However, they are by no means equally proficient at every scenario. There are some scenarios where they still perform poorly. This result simply reflects the difficulty of generalization to numerous diverse environments.

Let us next compare the performance of the online and offline populations in a number of different scenarios.

6 Results

In this section we will address a number of questions about the performance of online evolution. First we will examine how online evolution compares to offline evolution on average. We will then look at specific examples in order to tell under what conditions online evolution performs the best. We will also examine some special cases, like completely novel scenarios and the effect of

unexpected environmental transitions on a population. Finally we will look at the different components of our online evolution method and see how robust it is against variations in the algorithms. Our tests will show conclusively that online evolution is a useful tool that can handle cases where offline evolution is poorly suited. Note that this is by no means a predictable result. Online evolution is constantly at peril for sudden, unpredictable deteriorations, because of it's real-time nature. It is reasonable to question whether such instability might make it less useful than its offline counterpart.

6.1 Comparing Populations

The comparisons need to be done with respect to an appropriate performance measure. One possibility is using the selection criterion of equation 1. Even though this measure is an accurate description of the evolution process, which involves maximizing both survival and amount of gold collected, it is not a good illustration of the performance in an individual game. Since the birth of a peon has an initial cost, the selection criterion tends to show an initial drop that does not reflect the quality of the population. Therefore, a related measure of population performance is used instead: After every x iterations, the number of mines found per peons deployed is recorded. This is essentially the 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. Note that this measure could not be used as a selection criterion in online evolution since each peon is evaluated separately, and total deployment is a property of the entire population.

The offline population is evolved first. Two copies of it are made. One of them, labeled "offline" in figures 8-13, 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 should be noted 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 may lose its ability to deal with other enemies. The population can also become extremely unlucky and for a long period of time, not a single peon may survive. The selection criterion becomes essentially random in this case and the population will degrade greatly. These problems do not affect offline populations, since they do not change during testing, even if many of them are suddenly killed. The question

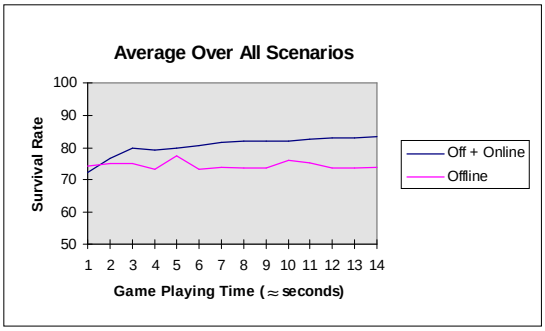


Figure 7: Average performance of a population that is allowed to evolve online (marked "Off + Online") compared to one that is not ("Offline"). After a few time steps the online population is able to improve its performance, while the static offline population performs approximately the same throughout the game (each time step is equal to 250 passes through the main loop of the game program).

is whether such disasters hamper online evolution to such an extent that it cannot operate.

6.2 Average Performance

It turns out that online evolution easily overcomes these challenges and significantly increases performance. To find this out, the performance of offline and off+online populations was measured over all 16 scenarios and averaged. This average is graphed in Figure 7, giving an idea of general performance over time across all scenarios. The performance is measured every 250 simulation time steps, or approximately every second of real game playing time. These measurements were then 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 average 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.

6.3 Performance in Individual Scenarios

Let us now look at how online versus offline evolution performed in individual scenarios. It turns out that when an offline population is already highly efficient in a particular scenario, online evolution does little to improve the

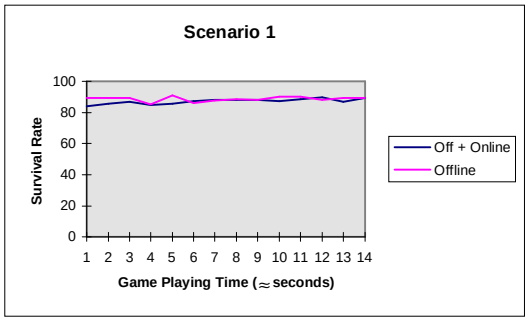


Figure 8: Both populations perform well in Scenario 1, and online evolution is not a major advantage.

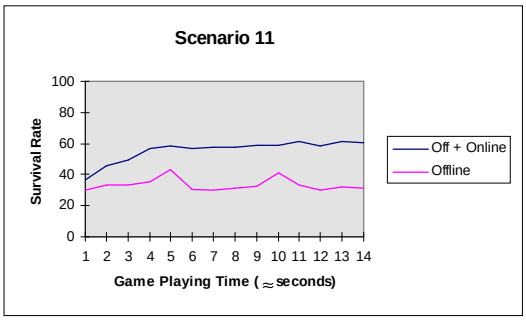


Figure 9: In scenarios that are hard for the offline population, training online can make a major difference, like here in Scenario 11.

performance. However, performance generally does not degrade either, showing that since they rarely happen, devastating events are not a major problem. For example, in Scenario 1(Figure 8), where the enemy's strategy is simplistic and the mines are both south of the home base, the offline population already performs at 90% and the off+online population does not improve it much (the differences are not statistically significant). Scenario 1 is one of the easiest in the game. The more difficult scenarios are more interesting because in many of those, offline evolution could not perfect the peons. The online populations showed statistically significant improvement in 10 of the scenarios, and the most improvement in those that were the hardest for the offline population. Scenario 11 is particularly dramatic (Figure 9). While the offline population struggles at around a 30% survival rate, the online population climbs into the 60% range. This scenario was difficult because the enemy often stayed close to the base and was able to scare most of the peons away from the mines. When allowed to evolve, however, the peons would risk coming closer to the enemy to get to a mine, and generally became better at it. This result shows that if there is room for improvement, online evolution will make the necessary adjustments so that it can make use of it, whereas the offline population is limited to the behavior that works well on average.

6.4 Performance of Online Evolution Alone

In some domains it may not be possible to pre-evolve the population offline, or such initial training may be wiped out by a devastating event. An important question is, is online evolution powerful enough so that it can solve the task even when started from scratch? It turns out that given a little time, even a population started with random weights will eventually surpass a pre-evolved

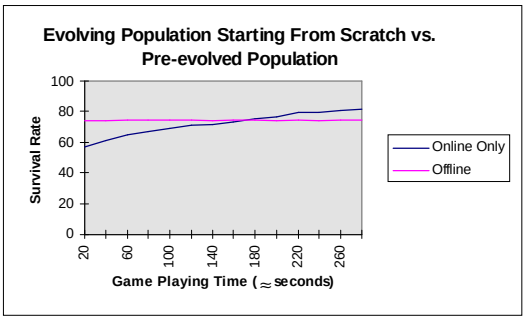


Figure 10: A population started with random weights that evolves online will outperform the population trained offline when given enough time.

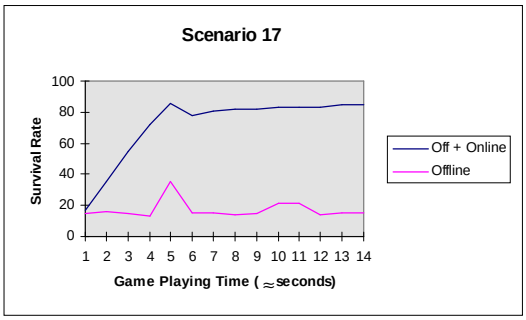


Figure 11: Online evolution quickly adapts to the new scenario (up to a 90% survival rate) whereas the static offline population cannot generalize to it at all (survival rate less than 20%). The results are statistically significant from the second time step.

offline population in performance. Figure 10 shows the average performance over 100 trials over 16 scenarios, with the time scale extended to 70,000 time steps, or about 5 minutes real-time. The online population starts out poorly, but catches up with the offline population after about 180 seconds! The differences are statistically significant within the time intervals [0,120] and [220,280]. This result is important in that it shows that online evolution is a general and robust method. If initial offline evolution is feasible, it will speed up learning, but it is not necessary for the algorithm to work.

6.5 Effects of a New Scenario

Since the offline population was trained to perform well in a variety of scenarios, the question arises as to how well it would perform in a completely new scenario. To answer this question, Scenario 17 was created, which consisted of two mines close together and an enemy that moved back and forth far enough to block both mines from one side. It turns out the novelty of the scenario is devastating to the offline population which has never seen anything like it before. The members of the population either get killed by the enemy or never make it to any mine. However, when allowed to evolve online, the population shows truly dramatic improvement (Figure 11). The members of this population were able to find a distant path around the area that allowed them to avoid the enemy and eventually get to the mines. This result shows that online evolution is very

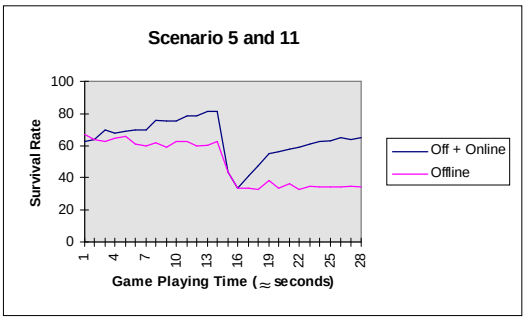


Figure 12: Even after the population has adapted to Scenario 5, it has no trouble adapting to a sudden change to scenario 11. The graph is an average of 100 runs. The differences are statistically significant within the time intervals [6, 14] and [17, 28].

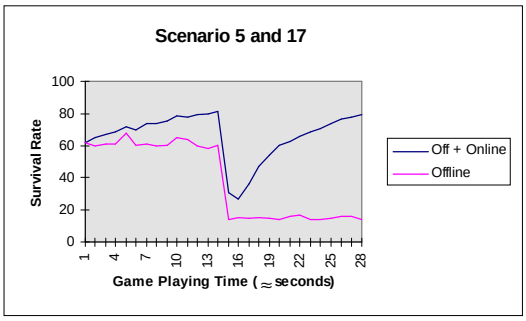


Figure 13: The improvement is even more dramatic when the new scenario is the novel Scenario 17. This graph presents an average of 100 runs, and the differences are statistically significant within the time intervals [6, 28].

powerful in adapting to novel environments. This is perhaps its most important advantage in real world application, when 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.

6.6 Effects of Environmental Transitions

Another interesting special case is when the scenario changes during performance. This is a common occurrence in the real world where the environment is not under the control of the agent. For example, a sensor can fail or a motor or some other part can break in a robot. In gaming, a transition can happen when a population is smoothly tackling a scenario with steady proficiency when suddenly the scenario changes. In principle such a transition could be difficult for the online evolution because it has strongly adapted to the current scenario and may have unlearned the skills required in other scenarios.

To test the effect of environmental transitions, a population was started off in scenario 5 and then switched at time 15 to another scenario (11 in Figure 12 and 17 in Figure 13). Since the offline population is static, it is not effected by such transition, and performs at its normal level in all these scenarios (cf. Figures 9 and 11). Predictably, the online population's performance initially falls, but surprisingly little. It actually never falls below offline performance, suggesting that it has not unlearned the offline behavior; it just has not yet adapted to these new scenarios. After a brief adaptation period, the online performance is back at

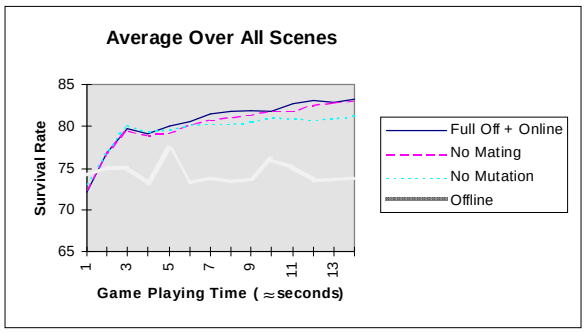


Figure 14: Performance of online evolution using different types of reproduction algorithms. The performance of the populations evolved online increases at a rate largely independent of the type of reproduction used. The graph is an average of 100 runs over the 16 scenarios, and the differences between the online evolutions are statistically insignificant. The offline evolution performance is also included for comparison.

around 80%, much higher than that of the offline population. This result shows that online evolution is able to handle sudden environmental transitions, making it a robust approach to real world tasks.

In sum, online evolution shows superior performance to offline evolution on average. In individual scenarios, online evolution does as well or better than pre-evolved offline populations. In a matter minutes a completely raw online population surpasses an offline population that took hours to train. Online populations can handle new scenarios while offline populations generalize very poorly to them. Environmental transitions do not throw online evolution off course. Instead the population is back at superior performance in a few seconds. Online evolution is therefore superior to offline evolution in almost every case. Lets next look at the robustness of the online evolution method against variations in the algorithm.

6.7 Deconstructing Online Evolution

The online evolution method in this paper used a specific combination of stochastic mutation and mating. It turns out that the exact reproduction mechanisms used are not critical to the success of online evolution. In an additional experiment, stochastic mutation was turned off, leaving mating as the only form of creating a new peon. In another, mating was turned off leaving only stochastic mutation. The average results over all scenarios are shown in figure 14.

The graphs reveal that versions of our online method perform at the same level on average. This same result applies to all 16 different scenarios individually. In other words, whether the peons use stochastic mutation, crossover mating, or both to reproduce, it makes little difference. What makes our method work is not the exact form of reproduction used, but that we can evaluate and update peons in such a way that they can be evolved while the population is in use. This is an encouraging result since it suggests that many different versions of genetic algorithms should work in online evolution. This is in contrast to previous work on offline evolution in difficult tasks where a delicate balance exists between crossover and mutation rates (Whitley, Dominic, Das, and Anderson, 1993).

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, as the analysis of Scenario 11 shows (Figure 9). In real life, however, future scenarios are not predictable. Novel scenarios constantly pop up. The ability to react to novel situations is part of what many people think of as intelligent behavior. Online evolution displays precisely this ability to acclimate to new situations, as the analysis of Scenario 17 shows (Figure 11). This property of online evolution is a key result of this research. Online evolution is an extremely useful tool for coping with unpredictable situations.

Online evolution is a viable method also for its ability to adapt in *real-time*. For example in Figure 10, within a matter of minutes, online evolution from scratch surpassed 20 hours of offline evolution. The difference is that online evolution needs to adapt only to the current scenarios, whereas offline evolution needs to optimize the behavior over all the different scenarios. If this scenario then changes, online adaptation takes only a few seconds, as shown in figures 12 and 13. Such a delay is reasonable in games and in many other applications as well.

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 theoretically 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.

Several possibilities exist for future work. For example, it might be possible to speed up evolution in real-time games using background evolution. In interactive game play, the rate at which evolution can occur is controlled by the pace of the game which is probably much slower than what the processor can handle. It might be possible to evolve the population in the background using artificially generated variations of scenarios that are likely to occur. Such evolution could proceed at the maximum speed of the processor, and the rate of evolution could be greatly increased.

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. One can imagine a robot with a failed sensor. If the robot has a few test actions it should be able to perform, it can evolve models for action which are evaluated on their ability to perform the test actions using online evolution. Eventually, a model would arise that compensates for the lack of the sensor sufficiently to handle the test actions. This ties in with online evolution's aptness at adapting to novel scenarios.

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 very strong performance data 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.

Acknowledgments

This research was supported in part by NSF under grant #IRI-9504317.

References

Gomez, F., and Miikkulainen, R. (1997). Incremental Evolution of Complex General Behavior. *Adaptive Behavior*, 5:317-342.

Moriarty, D. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis. Technical Report, UT-AI97-257.

Moriarty, D., and Miikkulainen, R. (1995). Discovering Complex Othello Strategies through Evolutionary Neural Networks. *Connection Science*, 7:195-209.

Moriarty, D., and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*,

Nolfi, S., Elman, J., and Parisi, D. (1994). Learning and Evolution in Neural Networks. *Adaptive Behavior*, 2:5-28

Pollack, J., Blair A., and Land M. (1996). Coevolution of Backgammon. In *Proceeding of the Fifth Artificial Life Conference*. Cambridge, MA: MIT Press.

Richards, N., Moriarty, D., and Miikkulainen, R. (in press). Evolving Neural Networks to Play Go. *Applied Intelligence.*

Werner, G., and Dyer, M. (1991). Evolution of Communication in Artificial Organisms. *Artificial Life II*, 659-687. Reading, MA: Addison Wesley.

Werner, G., and Dyer, M. (1993). Evolution of Herding Behavior in Artificial Animals. In *From Animals to Animats 2: Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior,* Langton, C., Taylor, C., Farmer, J., and Rasmussen S., editors, Cambridge, MA: MIT Press.

Whitley, D., Dominic S., Das, R., and Anderson, C. (1993). Genetic Reinforcement Learning for Neurocontrol Problems. *Machine Learning*, 13: 259-284