Real-Time Evolution of Neural Networks in the NERO Video Game

Kenneth O. Stanley[†], **Bobby D. Bryant**[‡], **Igor Karpov**[‡] and **Risto Miikkulainen**[‡]
[†] School of Electrical Engineering and Computer Science, The University of Central Florida, Orlando, FL 32816 USA
[†] Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712-0233 USA
kstanley@cs.ucf.edu, {bdbryant, ikarpov, risto}@cs.utexas.edu

Abstract

A major goal for AI is to allow users to interact with agents that learn in real time, making new kinds of interactive simulations, training applications, and digital entertainment possible. This paper describes such a learning technology, called real-time NeuroEvolution of Augmenting Topologies (rtNEAT), and describes how rtNEAT was used to build the NeuroEvolving Robotic Operatives (NERO) video game. This game represents a new genre of *machine learning games* where the player trains agents in real time to perform challenging tasks in a virtual environment. Providing laymen the capability to effectively train agents in real time with no prior knowledge of AI or machine learning has broad implications, both in promoting the field of AI and making its achievements accessible to the public at large.

Introduction

One of the main challenges for AI is to create intelligent agents that adapt, i.e. change their behavior based on interactions with the environment, becoming more proficient in their tasks over time, and adapting to new situations as they occur. Such ability is crucial for deploying robots in human environments, as well as for various software agents that live in the internet or serve as human assistants or collaborators.

While general such systems are still beyond current technology, they are already possible in special cases. In particular, video games provide complex artificial environments that can be controlled, and carry perhaps the least risk to human life of any real-world application (Laird & van Lent, 2000). On the other hand, such games are an important part of human activity, with millions of people spending countless hours on them. Machine learning can potentially make video games more interesting and decrease their production costs (Fogel, Hays, & Johnson, 2004). In the long run, it might also make it possible to train humans realistically in simulated adaptive environments. Video gaming is therefore an important application of AI on its own right, and an excellent platform for research in intelligent adaptive agents.

There is very little adaptation in current video games. The behavior of the *non-player-characters* (NPCs), i.e. the autonomous computer-controlled agents in the game, is often repetitive and predictable. They are usually controlled through scripts that cannot learn or adapt: The agents will always make the same moves in the same situations, and the game quickly becomes boring. Machine learning can in principle be used to adapt the agents' behavior in such cases. However, a major problem with current learning techniques is that the game content becomes unpredictable. Because correct behaviors are not known, learning must be done through exploration. As a result, agents sometimes learn idiosyncratic behaviors and sometimes do not learn at all, making the gaming experience unsatisfying.

While AI techniques have been used in video games before, they are usually based on scripts, rules, and planning, and do not involve genuine adaptation (Agre & Chapman, 1987; Maudlin *et al.*, 1984). When machine learning methods are employed, they serve to train agents effectively *before* a game is released (Cole, Louis, & Miles, 2004; Geisler, 2002; Hong & Cho, 2004). However, if agents are to adapt and change *during* gameplay, the learning method needs to be powerful and reliable enough to run interactively in real time. This paper describes such a method, rtNEAT (real-time NEAT), developed as a real-time enhancement of the NeuroEvolution of Augmenting Topologies learning technique (NEAT; Stanley & Miikkulainen, 2002, 2004). The main idea behind NEAT is *complexification*, i.e. starting with simple networks and gradually adding nodes and connections as their behavior becomes more complex. The main innovation in rtNEAT is to complexify neural networks *as the game is played*, making it possible for agents to evolve increasingly sophisticated behaviors in real time.

In order to demonstrate the potential of rtNEAT, the Digital Media Collaboratory (DMC) at the University of Texas at Austin initiated the NeuroEvolving Robotic Operatives (NERO; Stanley, Bryant, & Miikkulainen, 2005a) project in October of 2003 (`http://nerogame.org`). Based on a proposal by Kenneth O. Stanley, the idea was to create a game genre in which learning is indispensable, i.e. a *machine learning game*. In NERO, the player takes the role of a trainer, teaching skills to a set of intelligent agents controlled by rtNEAT. Thus, NERO demonstrates how machine learning in general can open up new possibilities in gaming, and how rtNEAT in particular establishes powerful real-time adaptation. In addition to being a research platform, NERO is also appealing as a game, showing how AI can be effectively introduced to the public at large. NERO has been downloaded over 50,000 times, bringing machine learning into the consciousness of video game enthusiasts

around the world. Players who previously had no exposure to machine learning are now training agents to perform sophisticated tasks using rtNEAT.

The next section introduces NEAT and explains how it was enhanced to create rtNEAT. The following section describes NERO and summarizes the current status and performance of the game. The last section discusses broader implications to the field of AI.

Real-time Neuroevolution of Augmenting Topologies (rtNEAT)

The NeuroEvolution of Augmenting Topologies (NEAT) method was originally developed to solve difficult control and sequential decision tasks. The neural networks control agents that select actions based on their sensory inputs. While previous methods that evolved neural networks, i.e. *neuroevolution* methods, evolved either fixed topology networks (Gomez & Miikkulainen, 1999; Saravanan & Fogel, 1995; Wieland, 1991), or arbitrary random-topology networks (Gruau, Whitley, & Pyeatt, 1996; Opitz & Shavlik, 1997; Yao, 1999; Zhang & Muhlenbein, 1993), NEAT is the first to begin evolution with a population of small, simple networks and *complexify* the network topology over generations, leading to increasingly sophisticated behavior. Compared to traditional reinforcement learning techniques, which predict the long-term reward for taking actions in different states (Sutton & Barto, 1998), the recurrent networks that evolve in NEAT are robust in continuous domains and in domains that require memory, making many applications possible.

Before NERO, NEAT had been successfully applied to pole balancing, competing simulated robots, automobile warning systems, and the game of Go (Stanley & Miikkulainen, 2002, 2004; Stanley, Kohl, & Miikkulainen, 2005). Therefore, evolving agents for video game environments appeared promising. However, a major remaining challenge was to take an evolutionary algorithm, which is usually applied in an offline setting, and make it work in a real-time interactive game.

Before describing the real-time extension, let us review the three key ideas on which the basic NEAT method is based. First, in order to allow neural network structures to increase in complexity over generations, a method is needed to keep track of which gene is which. Otherwise, it is not clear in later generations which individual is compatible with which, or how their genes should be combined to produce offspring. NEAT solves this problem by assigning a unique *historical marking* to every new piece of network structure that appears through a structural mutation. The historical marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited during crossover unchanged, and allow NEAT to perform crossover without the need for expensive topological analysis. That way, genomes of different organizations and sizes stay compatible throughout evolution, solving the previously open problem of matching different topologies (Radcliffe, 1993) in an evolving population.

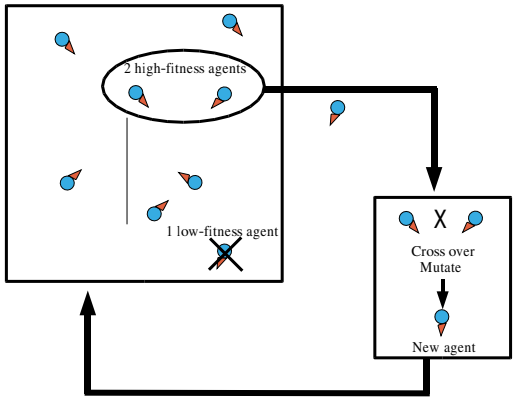

Figure 1: **The main replacement cycle in rtNEAT.** Robot game agents (represented as small circles) are depicted playing a game in the large box. Every few ticks, two high-fitness robots are selected to produce an offspring that replaces one of the robots with low fitness. This cycle of replacement operates continually throughout the game, creating a constant turnover of new behaviors.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before competing with other niches in the population. NEAT uses the historical markings on genes to determine to which species different individuals belong.

Third, unlike other systems that evolve network topologies and weights (Gruau, Whitley, & Pyeatt, 1996; Yao, 1999) NEAT begins with a uniform population of simple networks with no hidden nodes. New structure is introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. This way, NEAT searches through a minimal number of weight dimensions and finds the appropriate complexity level for the problem. This process of complexification has important implications for search: While it may not be practical to find a solution in a high-dimensional space by searching in that space directly, it may be possible to find it by first searching in lower dimensional spaces and complexifying the best solutions into the high-dimensional space.

As is usual in evolutionary algorithms, the entire population is replaced at each generation in NEAT. However, in a real time game or a simulation, such a step would look incongruous since every agent's behavior would change at once. In addition, behaviors would remain static during the large gaps between generations. Instead, in rtNEAT, a single individual is replaced every few game ticks. One of the worst individuals is removed and replaced with a child of parents chosen from among the best. This cycle of removal and replacement happens continually throughout the game and is largely invisible to the player (figure 1).

This replacement cycle presented a challenge to NEAT because its usual dynamics, i.e. protection of innovation through speciation and complexification, are based on equations that assume generational replacement. In rtNEAT, these equations are changed into probabilistic expressions that apply to a single reproduction event (Stanley, Bryant, & Miikkulainen, 2005a,b). The result is an algorithm that can

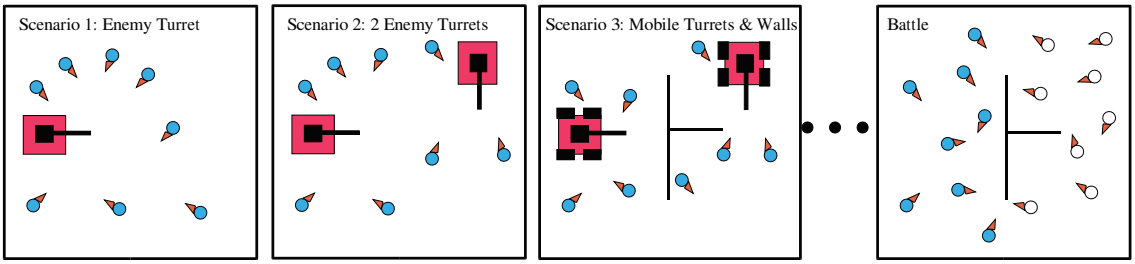

Figure 2: **A sample training sequence in NERO.** The figure depicts a sequence of increasingly difficult training exercises in which the agents attempt to attack turrets without getting hit. In the first exercise there is only a single turret but more turrets are added by the player as the team improves. Eventually walls are added and the turrets are given wheels so they can move. Finally, after the team has mastered the hardest exercises, it is deployed in a battle against another team.

evolve increasingly complex neural networks fast enough for a user to interact with evolution as it happens in real time.

The next section describes the new genre of games, made possible by rtNEAT, where the player trains agents in real time.

NeuroEvolving Robotic Operatives (NERO)

NERO was created over a period of about two years by a team of over 30 student volunteers (programmers and artists). The first phase of the project integrated rtNEAT with with the Torque video game engine, (`http://www.garagegames.com/`). A sequence of subsequent experiments gradually expanded the agents' senses and abilities, adding depth and breadth to the game. A collection of training levels and battle scenarios were created to complete the first version of the game, released in June of 2005. The game is currently under its third year of active development; version 2.0, with more interactive gameplay, is scheduled to be released in the summer of 2006.

The learning agents in NERO are simulated robots, and the goal is to train a team of these agents for combat. The agents begin the game with no skills and only the ability to learn. In order to prepare for combat, the player must design a sequence of training exercises and goals. The design involves placing the agents in specific environments, and rewarding them for specific behaviors (such as approaching an enemy or avoiding getting hit). The exercises need to be increasingly difficult so that the team can begin by learning the basic skills and gradually build on them (figure 2). When the player is satisfied that the team is well prepared, the team is deployed in a battle against another team (trained by another player), making for a captivating and exciting culmination of training. The challenge is to anticipate the kinds of skills that might be necessary in battle and build training exercises to hone those skills. Notably, although the agents in NERO genuinely learn to perform novel tasks, the player need not have any prior understanding of machine learning or AI.

Behavior can be evolved very quickly in NERO, enough so that the player can be watching and interacting with the system in real time. Agents can learn diverse skills that are useful for battle while interacting with the player. Basic skills that the players usually teach the agents include attacking an enemy, dodging enemy fire (figure 3), and navigating a complex maze (figure 4). Further effective behaviors can then be identified by observing actual battles. For example, one of the teams was strong enough in its attack so that the opponent team had to back off, eventually reaching the field's bounding walls. Such opponents presented a surprisingly serious challenge since it was not possible to go around them (as they were trained to do; figure 3). However, when the attacking team was further trained against a turret that had its back against the wall, its agents developed an effective strategy: The team learned to hover near the enemy and fire when it turned away, but back off quickly when it turned towards them. In this way, rtNEAT can discover sophisticated tactics that dominate over simpler ones; the challenge for the player is to figure out how to set up the training curriculum so that they will emerge.

Discussion

The excitement of seeing one's training pay off in battle is one reason for the success of the game since its release. Furthermore, rtNEAT is sufficiently flexible and efficient to allow players to develop intuitions about the underlying mechanics of machine learning without extensive technical background. The NERO discussion forum currently has 1,400 articles and AI novices have posted strategic discussions with a surprising resemblance to publications in machine learning (see for example `http://www.geocities.com/danekjovax/nero-midrange/article-20050706.html`). In other words, NERO is successfully promoting AI to people who would otherwise never experience it. In the long term, such interactive gaming platforms may be a way to attract the next generation of researchers to careers in AI. More generally, NERO shows how video games, because of their mass appeal, can be an excellent vehicle for demystifying AI and demonstrating its value to the public.

Working real-time learning technology also opens up new opportunities for interactive machine learning in entertainment, education, and simulation. Because people in the real world also change and adapt as they interact, rtNEAT can bring a new level of realism to simulated interaction in a variety of domains. Potential future applications include disaster training (e.g. crowd control), massive multiplayer gaming, and battle simulation.

Conclusion

The NERO video game demonstrates that evolutionary computation techniques such as rtNEAT are now sufficiently

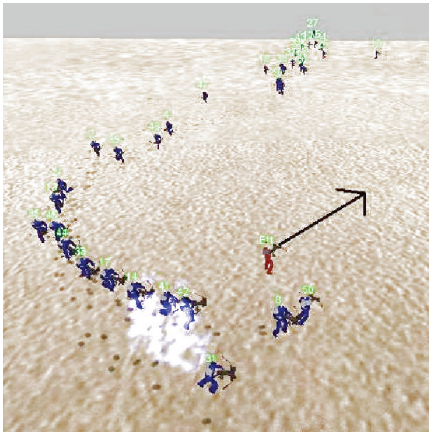

Figure 3: **Avoiding enemy fire.** The black arrow points in the current direction the enemy is firing (the arrow is not part of the actual NERO display). Agents learn to run safely around the enemy fire and attack it from behind. When the enemy moves, the agents change their attack trajectory accordingly. This behavior shows how evolution can discover behaviors that combine multiple goals.

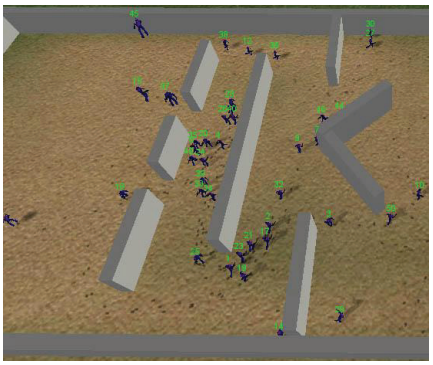

Figure 4: **Successfully navigating a maze .** The agents spawn from the left side of the maze and proceed to an enemy at the right. They learn to navigate mazes through gradual training on increasingly difficult wall configurations and do not need to employ an explicit path-planning algorithm.

flexible and robust to support real-time interactive learning in challenging sequential decision tasks. While game playing is a significant application on its own, these techniques can also be seen as a significant step towards building intelligent adaptive agents for human environments, such as training environments, robot controllers, and intelligent assistants.

Acknowledgments

Special thanks to NERO producer Aliza Gold and the members of the NERO team. This research was supported in part by the Digital Media Collaboratory (DMC) of the IC^2 Institute, by NSF (IIS-0083776), and THECB (ARP-003658-476-2001). The Torque engine is licensed from GarageGames (`http://www.garagegames.com/`). The rtNEAT software is available freely for research purposes at `http://www.nn.cs.utexas.edu/keyword?rtNEAT` and the NERO video game at `http://nerogame.org`.

References

Agre, P. E., and Chapman, D. 1987. Pengi: An implementation of a theory of activity. In *Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence*, volume 1, 268–272. Los Altos, CA: Morgan Kaufmann.

Cole, N.; Louis, S.; and Miles, C. 2004. Using a genetic algorithm to tune first-person shooter bots. In *Evolutionary Computation, 2004. CEC2004. Congress on Evolutionary Computation*, volume 1, 139–145. Piscataway, NJ: IEEE.

Fogel, D. B.; Hays, T. J.; and Johnson, D. R. 2004. A platform for evolving characters in competitive games. In *Proceedings of 2004 Congress on Evolutionary Computation*, 1420–1426. Piscataway, NJ: IEEE Press.

Geisler, B. 2002. An empirical study of machine learning algorithms applied to modeling player behavior in a 'first person shooter' video game. Master's thesis, Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI.

Gomez, F., and Miikkulainen, R. 1999. Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1356–1361. San Francisco: Kaufmann.

Gruau, F.; Whitley, D.; and Pyeatt, L. 1996. A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R.; Goldberg, D. E.; Fogel, D. B.; and Riolo, R. L., eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

Hong, J.-H., and Cho, S.-B. 2004. Evolution of emergent behaviors for shooting game characters in robocode. In *Evolutionary Computation, 2004. CEC2004. Congress on Evolutionary Computation*, volume 1, 634–638. Piscataway, NJ: IEEE.

Laird, J. E., and van Lent, M. 2000. Human-level AI's killer application: Interactive computer games. In *Proceedings of the 17th National Conference on Artificial Intelligence and the 12th Annual Conference on Innovative Applications of Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Maudlin, M. L.; Jacobson, G.; Appel, A.; and Hamey, L. 1984. ROG-O-MATIC: A belligerent expert system. In *Proceedings of the Fifth National Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI-84)*.

Opitz, D. W., and Shavlik, J. W. 1997. Connectionist theory refinement: Genetically searching the space of network topologies. *Journal of Artificial Intelligence Research* 6:177–209.

Radcliffe, N. J. 1993. Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications* 1(1):67–90.

Saravanan, N., and Fogel, D. B. 1995. Evolving neural control systems. *IEEE Expert* 23–27.

Stanley, K. O., and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10:99–127.

Stanley, K. O., and Miikkulainen, R. 2004. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research* 21:63–100.

Stanley, K. O.; Bryant, B.; and Miikkulainen, R. 2005a. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation* 9:653–668.

Stanley, K. O.; Bryant, B. D.; and Miikkulainen, R. 2005b. Evolving neural network agents in the NERO video game. *IEEE Symposium on Computational Intelligence and Games*.

Stanley, K. O.; Kohl, N.; and Miikkulainen, R. 2005. Neuroevolution of an automobile crash warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Wieland, A. 1991. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), 667–673. Piscataway, NJ: IEEE.

Yao, X. 1999. Evolving artificial neural networks. *Proceedings of the IEEE* 87(9):1423–1447.

Zhang, B.-T., and Muhlenbein, H. 1993. Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems* 7:199–220.