Real-Time Neuroevolution in the NERO Video Game

Presentation by Brittany A. Duncan

Paper by Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen

Take away points…

- Neural networks can be used to train game agents in much less time than it would take to program the actions
- Neural networks also allow these actions to be reversed or adjusted as needed
- It is important to have speciation in order to foster innovation and population limitation in order to maintain speciation

Overview

- People
- Introduction
- Related Work
- Neuroevolution of Augmenting Topologies (NEAT)
- Real-Time NEAT (rtNEAT)
- Neuroevolving Robotic Operatives
- Playing NERO
- Discussion
- Conclusion
- What next?

People

- Kenneth Stanley:
  - Assistant Professor at University of Central Florida
  - Director of the Evolutionary Complexity Research Group
- Bobby Bryant:
  - Assistant Professor at the University of Nevada, Reno
  - Director of the Neuroevolution and Behavior Laboratory
  - Director of the CCRC Agent Modeling Laboratory
- Risto Miikkulainen:
  - Professor at University of Texas at Austin
  - Director of the UTCS Neural Networks Research Group

Introduction

- Why video games?
  - Decrease production cost, increase longevity
  - Millions of players
  - Drawback: unpredictable game play
- What is NERO?
  - Neuroevolving Robotic Operatives game
  - Proposed by Kenneth Stanley in October 2003
  - The idea is “without learning NERO could not exist as a game.”

Introduction

Related Work

- Machine Learning in games
  - Blondie24 playing checkers
  - Out-game-learning vs. In-game-learning
    - Learning complete when game ships
    - Learning from player interactions
  - Machine Learning Games
    - Player explicitly trains agents

Related Work

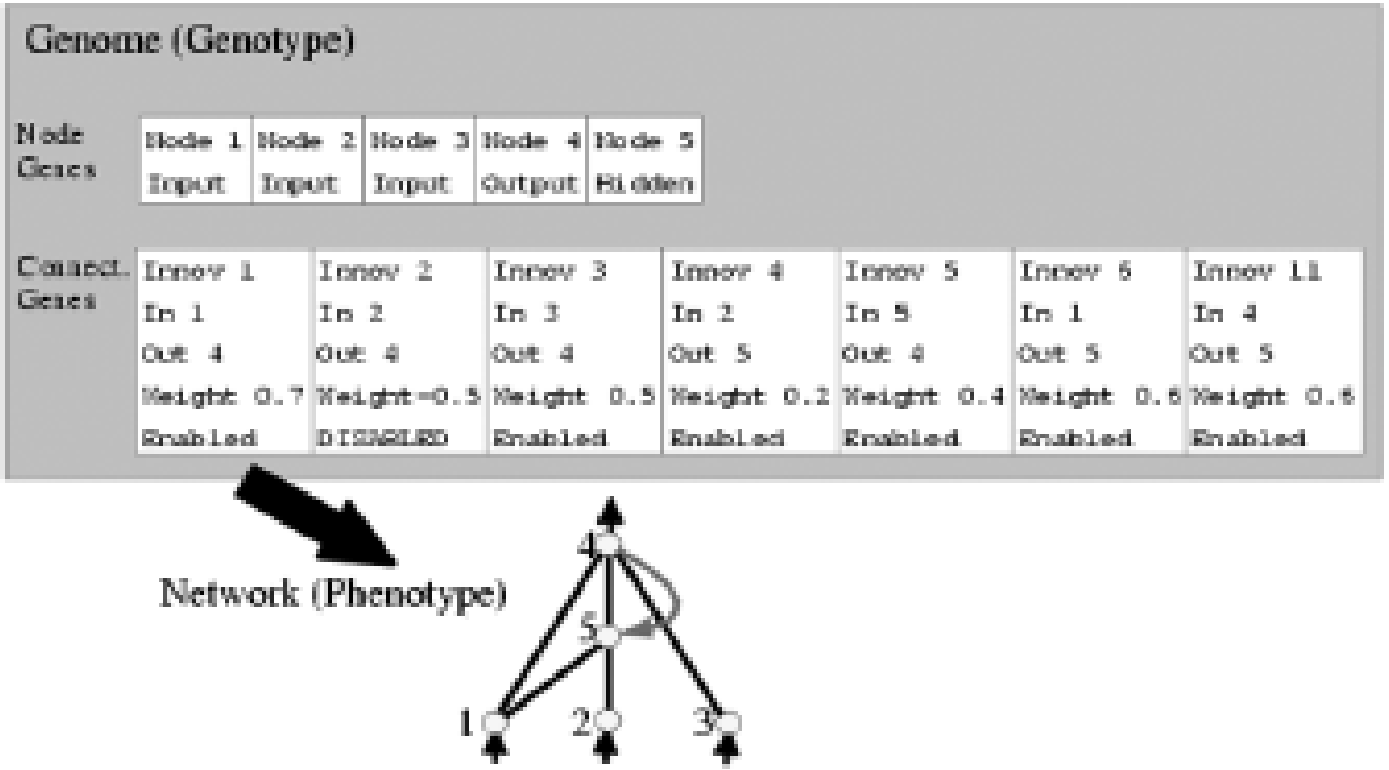
	Machine Learning	Neuroevolution
Large state/ action space	Large cost to game engine	Agents only output one action per tick
Diverse behaviors	Convergence is guaranteed	Speciation
Consistent individual behaviors	Random action required	Does not change, chooses from same network
Fast adaptation and sophisticated behaviors	Simple behaviors can be learned quickly, sophisticated are slower to develop	Representation is evolved, which allows them to be complexified
Memory of past states	Unrealistic to scale	Implements and utilizes effective memory structures

Related Work

- Topology and weight evolving artificial neural networks (TWEANNs) are different from other neural networks because they adapt based on the evolution
- Have we read about any others?
- NEAT is unique because it starts with minimal networks and adds nodes and connections over generations
  - This allows simple problems to help solve complex problems

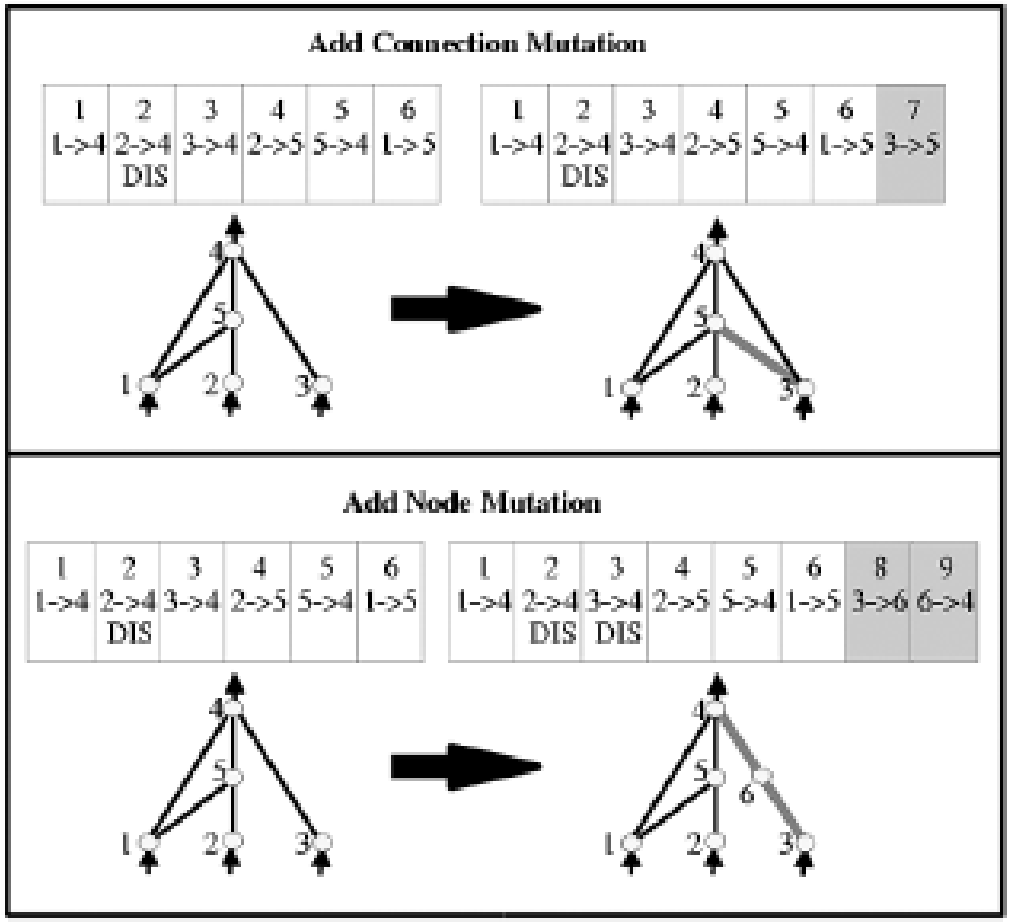
Neuroevolution of Augmenting Topologies (NEAT)

- Genetic Encoding



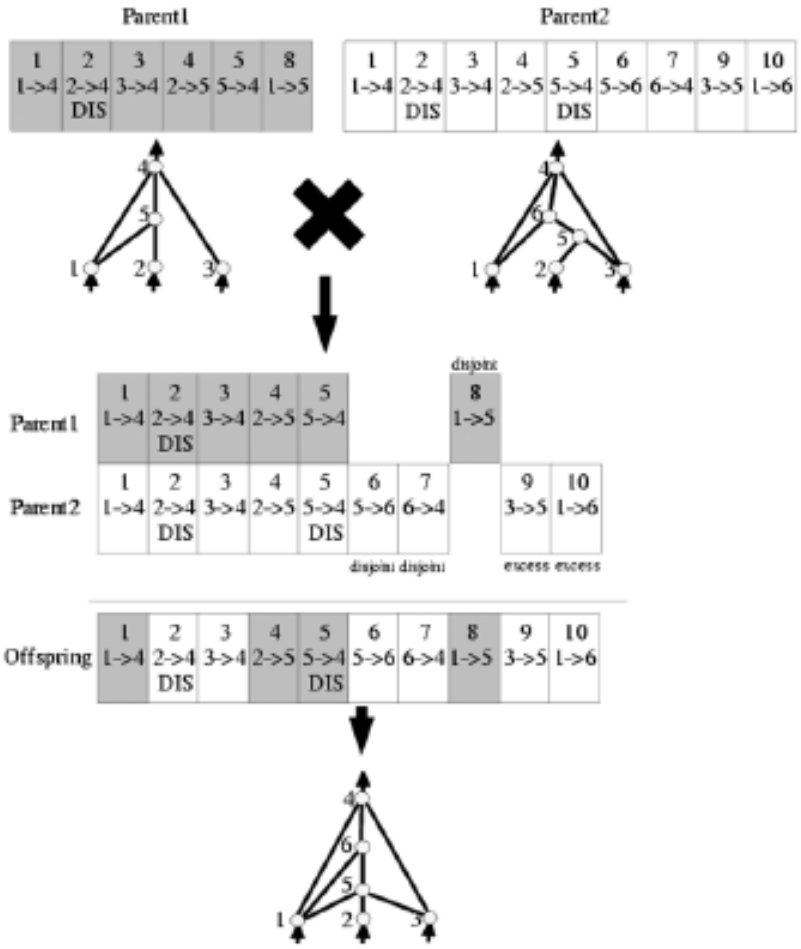
Neuroevolution of Augmenting Topologies (NEAT)

- Genetic Encoding
  - Add node makes the new weight between 3 and 6 equal to 1, and the weight between 4 and 6 equal to .5
  - This creates the nonlinearity



Neuroevolution of Augmenting Topologies (NEAT)

- Tracking Genes through Historical Markings
  - Global innovation numbers track changes
  - Tracking generations prevents the same change being given two numbers
  - Uniform crossover randomly chooses matching genes
  - Blended crossover averages the connection weights of matching genes



Neuroevolution of Augmenting Topologies (NEAT)

- Speciation
  - Allows individuals to compete with similar individuals rather than the population at large
  - Helps prevent bloating of genomes by allowing smaller genomes to survive as long as they are competitive

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \bar{W}. \qquad (1)$$

  - In order to divide the species, Formula 1 is used.
  - δ can be made dynamic in order to create the right number of species

Neuroevolution of Augmenting Topologies (NEAT)

- Speciation
  - Species reproduce by eliminating the lowest performing members and then using the remaining members' offspring to replace them
- Minimizing Dimensionality through Complexification
  - Begins with simple networks with no hidden nodes, but different initial weights
  - New structures are introduced through mutations and survival is based on fitness
  - This is similar to complexification in biology

Neuroevolution of Augmenting Topologies (NEAT)

- NEAT Performance
  - Necessity of components proven in previous work through testing interactions
  - Also proven to outperform other neuroevolution methods in previous works

Real-Time NEAT

- Motivation
  - Need to update NEAT to run online, on the entire population
  - Cannot replace the entire population at the same time
  - As in some evolutionary strategy algorithms, one individual is replaced every few game ticks

Real-Time NEAT

- The rtNEAT Algorithm
  1. Calculating Adjusted Fitness
     - Fitness of the individual/ number of individuals in the species
  2. Removing the Worst Agent
     - Agent with the worst adjusted fitness to protect small species
  3. Reestimating Average Fitness
     - Can change significantly between steps
  4. Creating Offspring
     - Chooses based on the average fitness compared to other species, preserving speciation

Real-Time NEAT

- The rtNEAT Algorithm
  5. Reassigning Agents to Species
     - The interval for this varies by game
     - Every 5 replacements for NERO because the game progresses quickly
  6. Replacing the Old Agent with the New One
     - Depends on the game and whether the agent's "head" can be replaced without the agent's body being destroyed

Real-Time NEAT

- Running the Algorithm
  - Law of eligibility describes the percentage of the population which will be ineligible for replacement when the evolution has reached a steady state
    - $I = m/Pn$
    - m is minimum time for evaluation, P is population size, and n is the number of ticks between replacements
  - The user should choose I and allow rtNEAT to compute n

Neuroevolving Robotic Operatives

- Training Mode
  - The exercises should start simply and then build upon the basic skills
  - The players can place objects on the field and determine goals using sliders
    - These sliders are the coefficients for the fitness components
  - Training is standardized by having all agents appear in the same area
    - Agents are only replaced when they are in this area
    - Agents are only evaluated when they are in this area

Neuroevolving Robotic Operatives

- Training Mode
  - A true average of the agent's fitness is maintained for the first few trials
  - A continuous leaky average is maintained after that
    - This allows agents to have an up-to-date average, but still one that takes all of their experiences into account
  - Teams can be saved when they reach a satisfactory level

Neuroevolving Robotic Operatives

- Training Mode

Neuroevolving Robotic Operatives

- Battle Mode
  - Team is 20 agents from as many different trained teams as desired
  - Designed to run over a server for competitions
  - The agents are destroyed after being shot several times
  - Action can cease before complete defeat if agents are continuously avoiding each other
  - Field can be made complex or simple

Playing NERO

- Training Basic Battle Skills
  - Seeking behavior is most simple
    - Can be trained in 99.7 seconds for 90% of agents
  - Different from most applications of EAs because the entire population is judged
  - Flexible enough to devolve a population from seeking behavior to avoidance behavior
    - Through this they learned to run backwards so they could still see to shoot

Playing NERO

- Training More Complex Behaviors
  - Could train to run around walls, through this and the addition of more walls they could navigate a maze
    - Generalized enough to navigate varied mazes
- Battling Other Teams
  - An aggressive seeking team won twice as many battles as an avoidant team (6 to 3), though this was noted as only a slight advantage
  - More interesting is that the challenge is to conceive clearly dominant strategies
  - A team based on avoiding a turret placed against a wall won the tournament and all games against the aggressive team

Discussion

- Neuroevolution can be deployed in a real game
- rtNEAT could be used in MMOGs to continually adapt them
  - Do you feel this would be beneficial?
- Open issue: how to assess results?
- These results could allow better training games

Conclusion

- Players are able to evaluate the population as it evolves asynchronously
- rtNEAT allowed a new type of game to be created

What next?

- The researchers at UT Austin are currently developing OpenNERO, funded by Google.
- The researchers at UCF are working on how to teach animated characters to dance using interactive evolution.
- At UCF, they are also working on Picbreeder, which breeds pictures by combining two pictures to create a “child” picture.
- The researchers at UNR are looking at Neuroevolution in the Quake II video game.

Take away points…

- Neural networks can be used to train game agents in much less time than it would take to program the actions
- Neural networks also allow these actions to be reversed or adjusted as needed
- It is important to have speciation in order to foster innovation and population limitation in order to maintain speciation

- Thank you for your attention. Any questions, comments, or concerns are welcome.