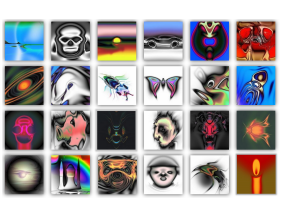
Interactive Neuroevolution

Risto Miikkulainen

October 14, 2024

Overview of Interactive Neuroevolution

- Neuroevolution allows the discovery of behaviors for agents embedded in environments.
- Solutions may be surprising and challenging for human designers.
- Human guidance may be helpful in discovering complex or aesthetic objectives.
- It also leads to a new genre of machine-learning games!

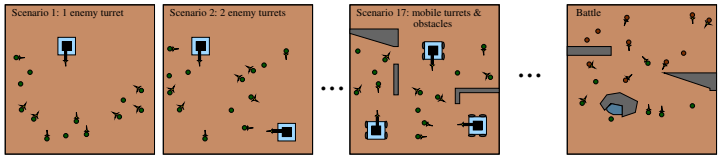

The NERO Machine Learning Game

- NERO is a pioneering machine learning game:
  - Produced by UT's Digital Media Collaboratory 2003-06
  - Professional producer, about 30 volunteer undergrads
- Agents are controlled by neural networks evolved using NEAT.
- Human players act as trainers or coaches, designing learning challenges.

NERO
NEURO EVOLVING ROBOTIC OPERATIVES

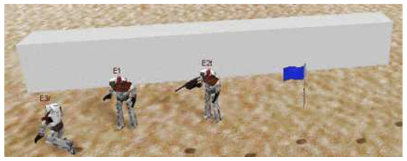

NERO Gameplay

- The task for the player is to train the agents for battle
  - Agents are placed in an enclosed environment with walls and obstacles.
  - Enemy agents can be static, mobile, or firing at the player's agents.
  - Player needs to design a curriculum of exercises
  - Agents evolve in real time
- After training, agents placed in a battle against another team
- New genre: Learning *is* the game
  - Challenging platform for reinforcement learning
  - Real time, open ended, requires discovery



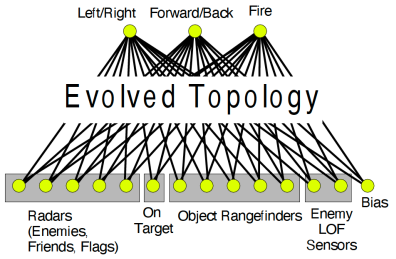
NERO Player Actions

- The player can place items in the field:
  - Walls, flags, static enemies, rovers, turrets.
- Players can adjust sliders to define fitness objectives
  - Approach/avoid enemies, cluster/disperse, hit targets, avoid fire...
- Curriculum can be designed dynamically, starting with simple tasks and increasing complexity.



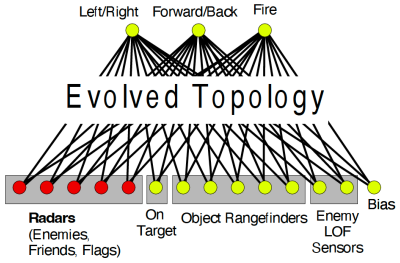
NERO Neural Network Agent

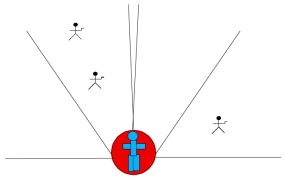
- Each agent is controlled by an evolved neural network
  - Inputs: egocentric sensors
  - Outputs: simple actions



NERO Neural Network Agent

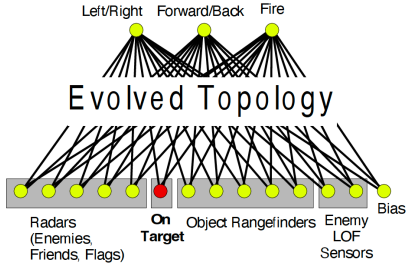
- Each agent is controlled by an evolved neural network
  - Inputs: egocentric sensors
  - Outputs: simple actions

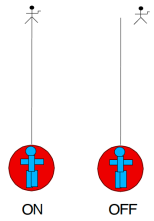




NERO Neural Network Agent

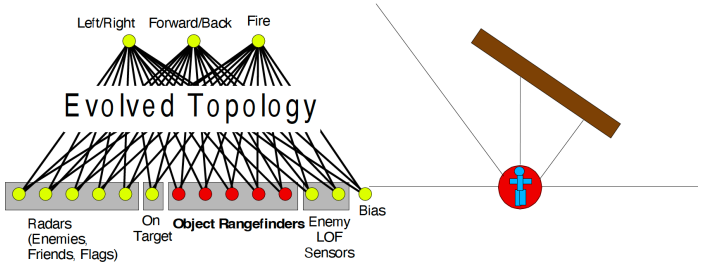
- Each agent is controlled by an evolved neural network
  - Inputs: egocentric sensors
  - Outputs: simple actions





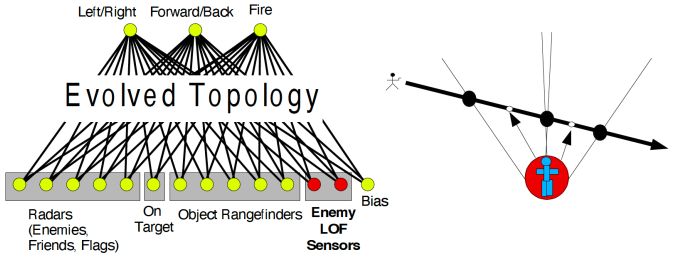
NERO Neural Network Agent

- Each agent is controlled by an evolved neural network
  - Inputs: egocentric sensors
  - Outputs: simple actions



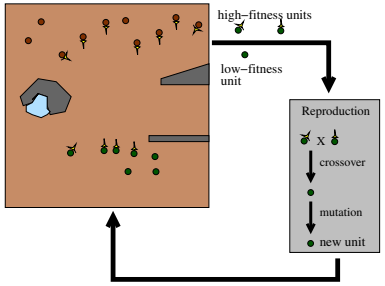
NERO Neural Network Agent

- Each agent is controlled by an evolved neural network
  - Inputs: egocentric sensors
  - Outputs: simple actions



Real-Time NEAT (rtNEAT)

- A parallel, continuous version of NEAT.
  - The entire population is evaluated together.
  - Parents selected probabilistically weighted by fitness.
  - The worst-performing agent is replaced with an offspring every n ticks
- This allows ongoing evolution without pauses between generations, making the process seamless for players.
- Long-term evolution is equivalent to generational NEAT.



NERO Training Demo

Demo

Approach Enemy

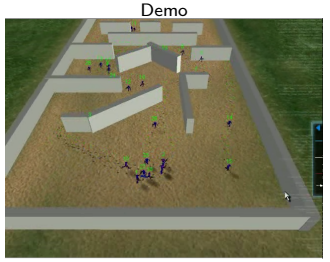
Demo

Switch to Avoid

Demo

Avoid, first-person

Demo



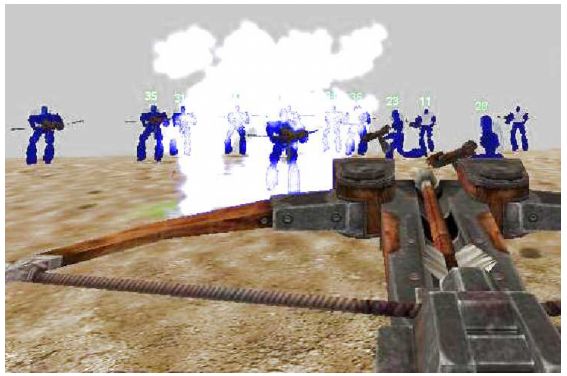
Maze Running

Battle Mode in NERO

- After evolution, teams are evaluated in battle mode.
- Teams compete in head-to-head matches in different environments.
- Agents use their evolved behaviors independently from the human player.
- Teams win when they eliminate the opponent or the clock runs out.

Discovering Effective Behaviors

- Agents can learn to charge enemies, hide, use cover, and attack strategically.
- Dynamic adjustments to fitness sliders allow agents to evolve toward effective strategies.
- Surprising strategies can emerge, such as running backward to avoid fire while keeping the weapon pointed to the enemy.



Unexpected and Effective Team Behaviors

- Evolution can discover unexpected but useful team behaviors as well:
- Avoidant teams retreat to form a strong firing squad.
- Subteams of agents evolve to coordinate attacks.

Demo

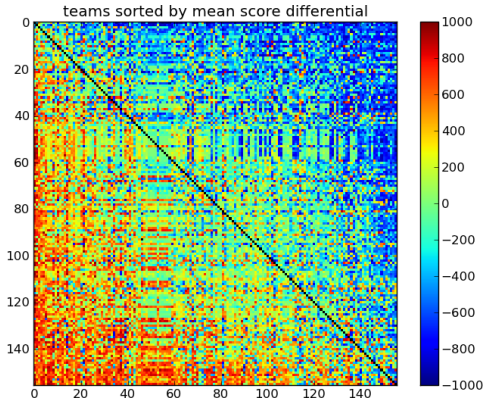
Aggressive vs. Avoidant

Demo

Teams of three

Circularity in Strategies

- Large-scale experiments show that no single strategy is dominant.
  - 159-team OpenNERO tournament in the first ever MOOC (2011).
- No single dominant strategy: many ways to be successful.
- Circularity: Team A beats Team B, B beats C, but C beats A.
- Multiple strategies needed; makes the game interesting!



But is it Fun?

- NERO was a tech/science demo.
- NERO 2.0: Territory Mode was an actual game.
  - Battle to conquer control points: interactive & more fun
  - Player trains various specialists: defenders, chargers, snipers,...
- During battle, player is a high-level commander
  - Dynamically deploys specialists
  - Dynamically specifies targets
  - Real time against other players
- Making a fun game is a different goal altogether.

Demo

Challenge: Utilizing Human Knowledge

- Given a problem, NE discovers a solution by exploring
  - Sometimes highly original solutions
  - Requires lots of exploration
- Game designers may want to have more control
  - Seeding with initial behaviors
- Players may want to interact with learning
  - Specifying desired behaviors during evolution

Incorporating Human Knowledge into NERO

- NERO already utilizes shaping through human insight.
- Human knowledge can also be incorporated through:
  - Rule-based advice.
  - Behavioral examples.
- These approaches can improve neuroevolution beyond trial and error.

Rule-Based Advice

- Humans can observe obvious behaviors that agents struggle to learn.
  - For instance how to go around a wall.
- Advice simplifies early training and allows evolution to focus on complex behaviors.

"If there is a wall in front, go around!

Example Advice

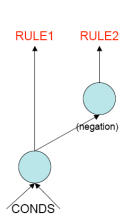
- "If a wall is in front, move forward and turn right"
  { **if** wall_ahead $>$ 0.1 **then** { output move_forward 1.0 turn 0.5 } }

  "If a wall is at 45° left, move forward and turn right slightly"
  { **if** wall_45deg_left $>$ 0.5 **then** { output move_forward 1.0 turn 0.1 } }
- Encoded in a simple grammar & translated into knowledge-based neural networks (KBANN)
- Added to the champions of the top 5 species

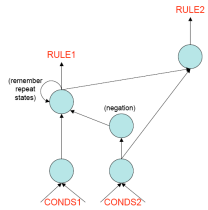
"If there is a wall in front, go around!

Advice Grammar

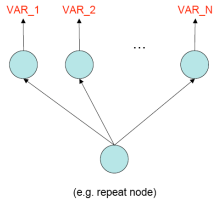
RULE	$\longrightarrow$	{ if CONDS then RULE [else RULE] } |
		{ when CONDS repeat RULE until CONDS [then RULE] } |
		{ output VARSTR }
CONDS	$\longrightarrow$	TERM | CONDS and TERM
TERM	$\longrightarrow$	false | true | variable { $\leq$ | $<$ | $\geq$ | $>$ } value
VARSTR	$\longrightarrow$	variable strength | VARSTR variable strength



if-then



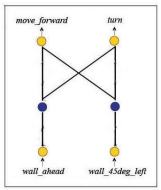
repeat



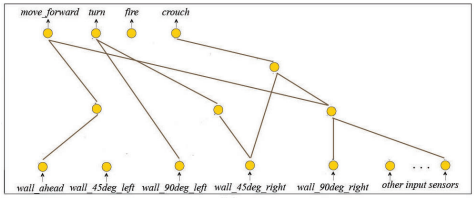
output

Adding Advice to a NEAT Network

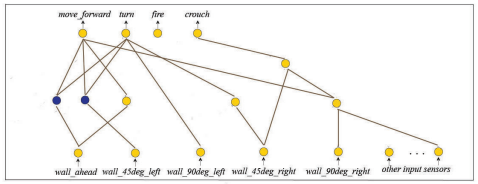
- KBANN network is added to NEAT networks; treated as a complexification
- Continues to evolve
  - If it is useful, it stays
  - Otherwise it is discarded or converted to something else
- Confidence values control how rigid or flexible the advice is.
- Can be given online as advice, or early as initial behavior.



Advice subnet



Existing NEAT network

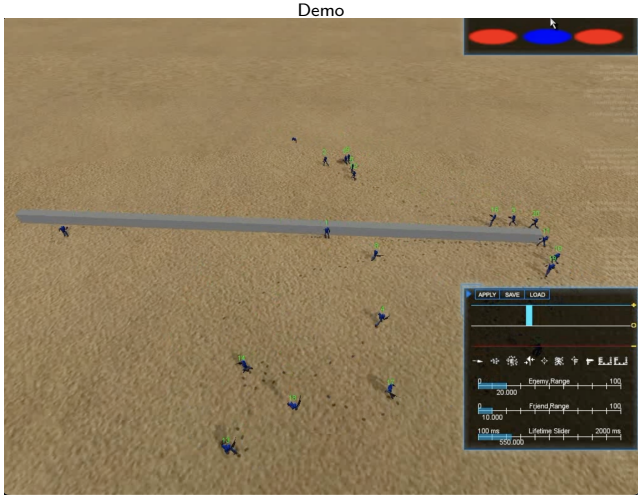


Network with advice added

Demo: Rule-Based Advice in NERO

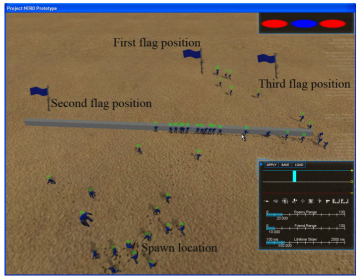
- Advice helps agents get around the wall right away.

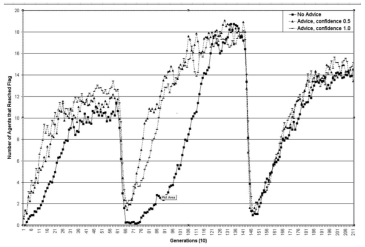
Demo



Advice in Continual Evolution

- Evolution eventually integrates the advice into the network for future tasks.
- Quickly adapts when task requirements change (e.g. switching directions).
- Advice accelerates learning but does not constrain long-term evolution.

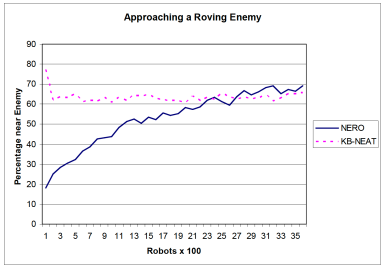




Seeding the Population

- KBANN+NEAT can also be used to seed an initial population
- E.g. approaching a roving enemy
  - With knowledge, approach right away
  - Otherwise, learn to do so in 3mins
- Does not delay adaptation to new situations
  - E.g. Approaching an enemy that is shooting

Demo



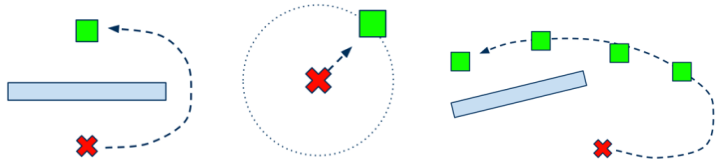
Training by Examples

- Some behaviors hard to formulate in rules, but can be demonstrated by humans
- E.g. take control of a NERO agent, demonstrate behavior
- Measure distance from the recorded path
  - Incorporate into the fitness function
  - Or train with backpropagation
- Injecting human knowledge as examples

Demo

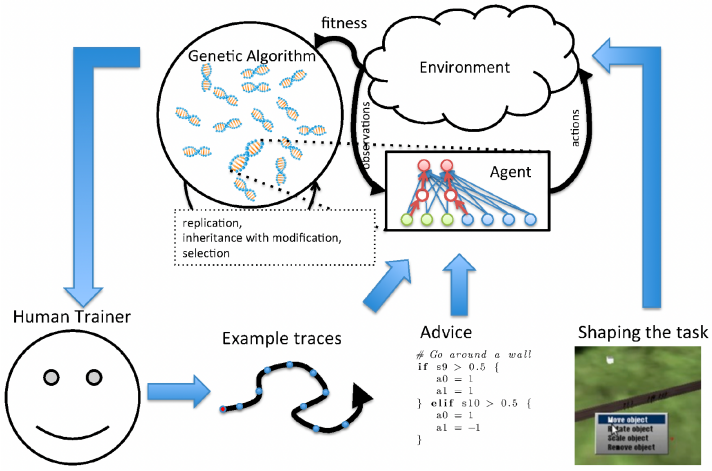
Human Subject Study: Evaluating Interaction Methods

- Study evaluated different approaches: advice, examples, and shaping.
- Each method plays a unique role depending on the complexity and nature of the task.
  - Advice: Best for tasks with general rules (e.g., catching a moving target).
  - Examples: Effective for specific, concrete behaviors (e.g., navigating a wall).
  - Shaping: Ideal for tasks that can be broken into simpler steps (e.g., traversing waypoints).



Active Human-Guided Neuroevolution

- The system could request advice, examples, or shaping when needed.
- The system identifies areas where human input would be most beneficial.
- This approach allows human knowledge and machine exploration to work together effectively.



NERO Conclusion

- NERO is a pioneering machine-learning game
  - "A video game created by eggheads".
  - A neuroevolution experiment can be an interesting game.
- A roadmap for combining human and machine creativity.
  - As in RHEA, but interactive.
- Difficult to transfer to game industry.
  - Engineers, game designers love it; management doesn't.
  - They have a winning formula already.
  - Lack of control on the outcome is scary.
- But times are changing...