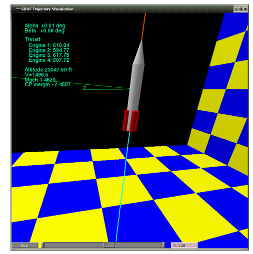
Neuroevolution of Behavior

Risto Miikkulainen

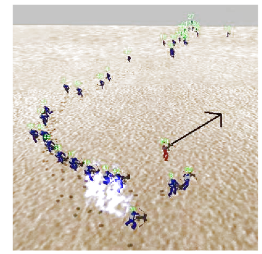
February 20, 2026

Introduction to Neuroevolution of Behavior

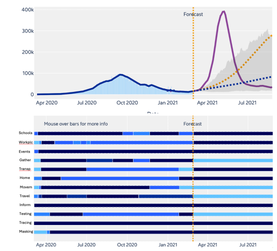
- Neuroevolution aims to construct agents that behave intelligently in simulated or real environments.
- Behavior is optimized at multiple levels:
  - Control tasks: locomotion for robots, production in bioreactors.
  - Behavioral strategies: navigation, gameplay, cognitive tasks.
  - Decision strategies: business, healthcare, societal decisions.



(Gomez & Miikkulainen 2003)



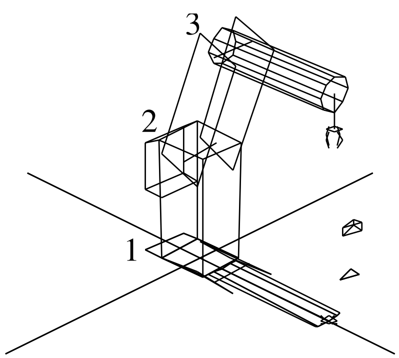
(Stanley et al. 2005)



(https://evolution.ml/demos/npidashboard/)

Neuroevolution for Control

- Neuroevolution has been applied to various control tasks, demonstrating creative solutions.
- Agents evolve to compensate for challenges such as hardware failures.
  - E.g. controlling a robotic arm when a motor fails:

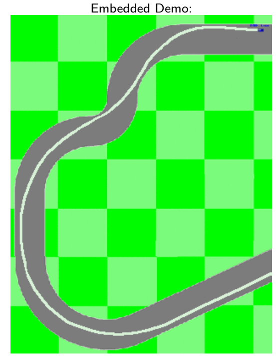


(Moriarty & Miikkulainen 1996)

Creative Problem Solving

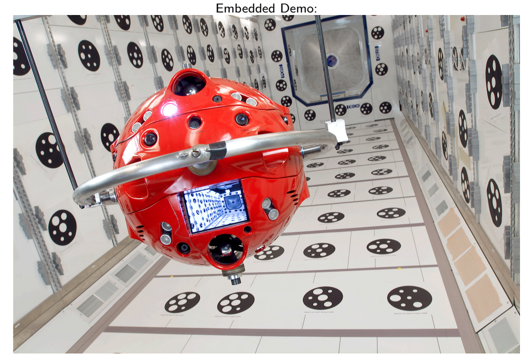
- Neuroevolution can find solutions not immediately obvious to human designers.
  - Driving a race cars by maximizing speed instead of minimizing distance.
  - Stopping spacecraft by rotating it around.

Embedded Demo:



(Stanley et al. 2005)

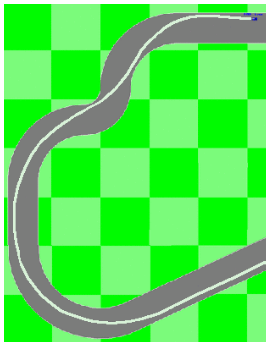
Embedded Demo:



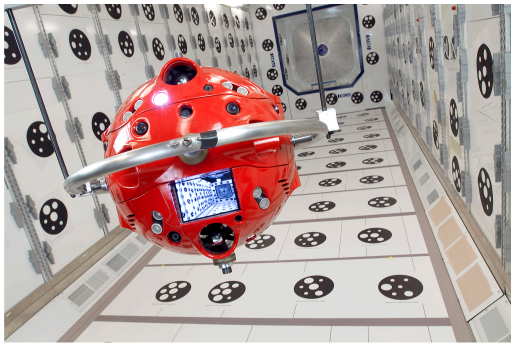
(Sit & Miikkulainen 2005 with permission)

Challenge: Robustness

- Robust control is difficult:
  - Environments can be dynamic, nonlinear, and noisy.
  - Conditions can change over time (e.g., sensor failure, obstacles, ice...).
- Neural networks can handle noise, nonlinear effects, and partial observability.
- Evolution needs to see enough such variation to be effective.



(Stanley et al. 2005)



(Miikkulainen 2005 with permission)

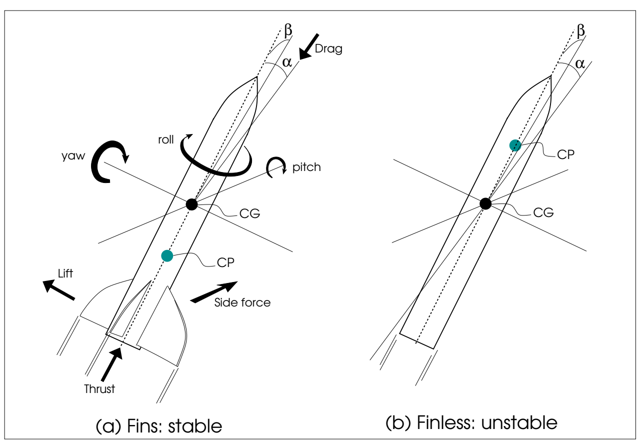
Example: Controlling a Finless Rocket

- Task: Stabilize a finless version of the Interorbital Systems RSX-2 sounding rocket
  - Scientific measurements in the upper atmosphere
  - 4 liquid-fueled engines with variable thrust
  - Without fins will fly much higher for same amount of fuel

(Interorbital Systems 2003)

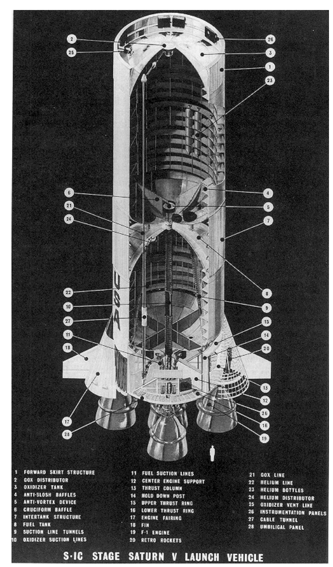
Rocket Stability

- Drag from fins pulls the Center of Pressure (CP) behind Center of Gravity (CG)
- Without fins, need active control.



(Gomez & Miikkulainen 2003)

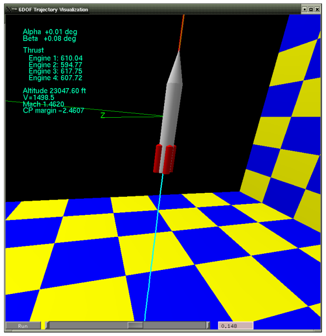
Active Rocket Guidance



(NASA 2004)

- Used on large scale launch vehicles (Saturn, Titan)
- Typically based on classical linear feedback control
- High level of domain knowledge required
- Expensive, heavy

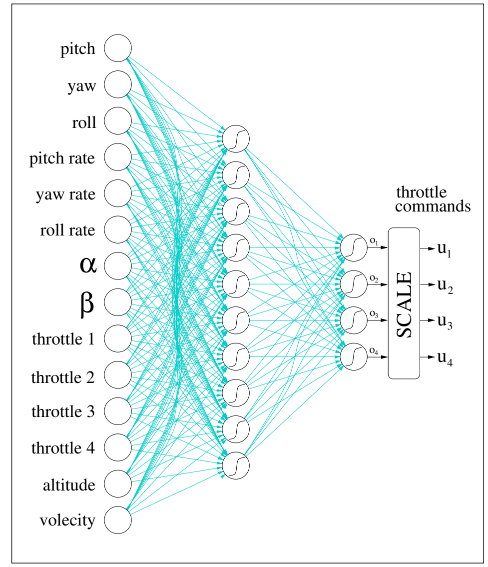
Simulation Environment: JSBSim



(Gomez & Miikkulainen 2003)

- General rocket simulator
- Models complex interaction between airframe, propulsion, aerodynamics, and atmosphere
- Used by IOS in testing their rocket designs
- Accurate geometric model of the RSX-2

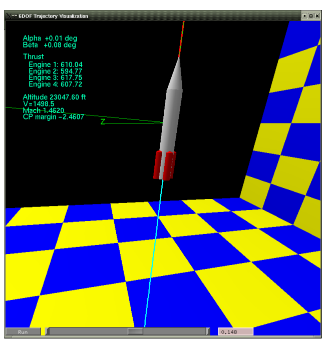
Rocket Guidance Network



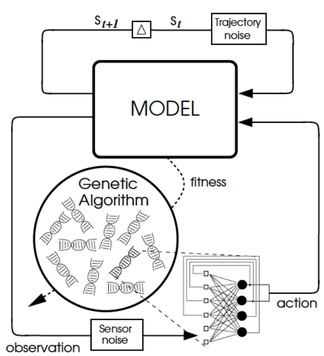
(Gomez & Miikkulainen 2003)

Idea: Adding Noise to Encourage Robust Control

- One approach to robust control is adding trajectory noise.
- Trajectory noise forces the controller into situations where it must recover.
- This method is more effective than sensor noise because it doesn't confuse the agent.



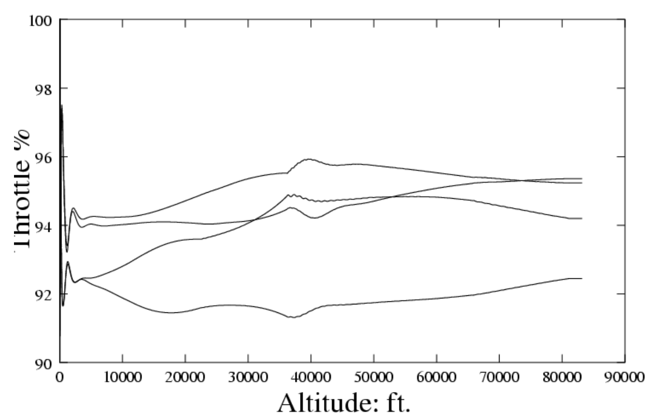
(Gomez & Miikkulainen 2003)



(Gomez & Miikkulainen 2004)

Results: Control Policy

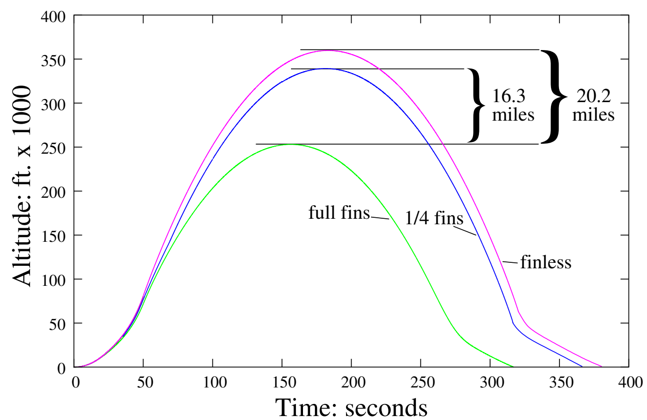
- Accurate control in the beginning.
- Flies through atmospheric disturbance later.



(Gomez & Miikkulainen 2003)

Results: Apogee

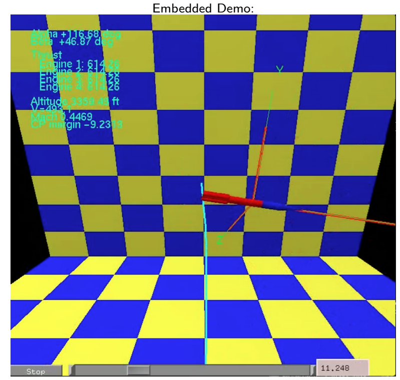
- Flies 20 miles higher without fins!
  (much of it coasting in thin air)



(Gomez & Miikkulainen 2003)

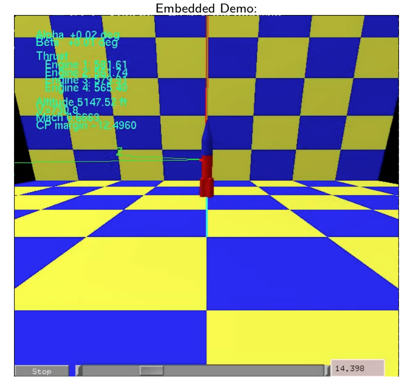
Finless Rocket Control Demo

Embedded Demo:



No active stabilization

Embedded Demo:



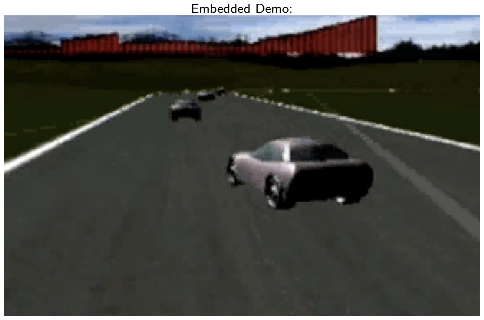
Evolved active stabilization

(https://nn.cs.utexas.edu/pages/research/rocket/)

Challenge: Generalizing to Novel Situations

- Even with robust control, handling significant changes remains a challenge.
  - Training on every possible scenario is not feasible.
  - Need to come up with systematic approaches to extrapolate.

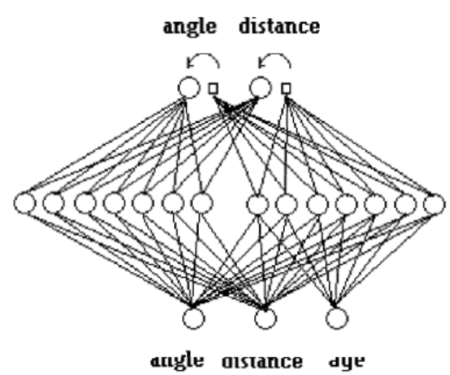
Embedded Demo:



(https://nn.cs.utexas.edu/pages/research/neat-warning/)

Idea 1: Teacher Networks for Enhanced Learning

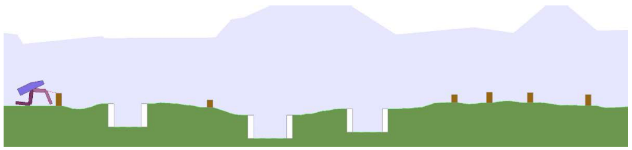
- Teacher networks generate learning targets for controllers that learn via backprop.
- Teachers are evolved based on how well the controller performs after training.
- E.g. in creating a controller that forages for food:
  - With extra input for the age of the controller.
  - Optimal training inputs do not correspond to correct targets!
  - Instead, they create maximally effective learning experiences



(Nolfi & Parisi 1994)

Idea 2: Coevolution of Problems and Solutions

- In some cases, problems and solutions can be coevolved together, encouraging robust behavior.
- E.g. POET: coevolution of obstacle courses and runners.
- It starts with simple obstacle courses and gradually complexifies them as agents evolve better behaviors.
- This process leads to more general and robust solutions

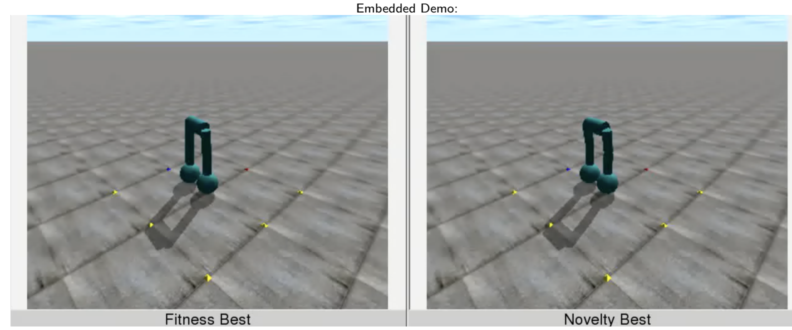


Demo link: `https://youtu.be/D1WWhQY9N4g?si=tmSrFmD8GNeNvA6L`

Idea 3: Novelty Search

- Novelty search rewards diversity in behavior rather than just goal achievement.
- This method encourages exploration, leading to more generalized and robust solutions.
- Example: Novelty search discovered a dynamic, fast bipedal walk, while fitness-based search failed.

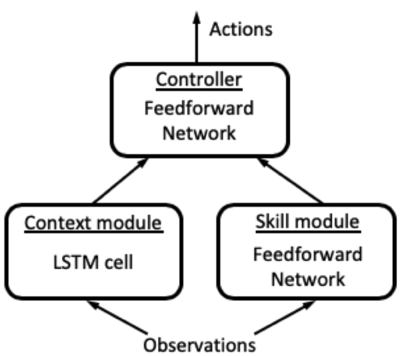
Embedded Demo:



(Lehman & Stanley 2011)

Idea 4: Modeling the Context Explicitly

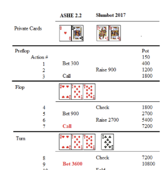
- The system can be designed with three components:
  - Skill network: Takes actions.
  - Context network: Models the environment.
  - Decision network: Uses context to modulate skill actions.
- This allows the controller to adapt actions based on the environment.



(Tutum & Miikkulainen 2020)

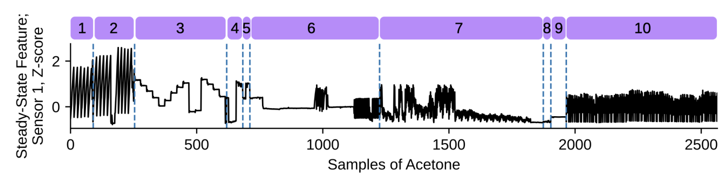
Context in Various Domains

- Opponent modeling in poker
  - Learn basic game play against canonical opponents
  - Track play by novel opponents; modulate play accordingly
  - Can generalize to much better opponents
- Context+Skill in physical games
  - Evaluated in FlappyBall, LunarLander, CARLA
- Tracking continuously changing environments
  - E.g. modeling sensor drift in odor recognition



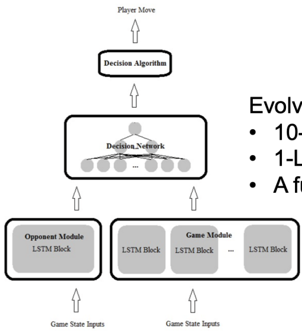
(Li 2018)

(Tutum et al. 2021)



(Warner et al. 2024)

Adapting to Novel Opponents in Poker



Evolve weights of poker-playing NN

- 10-LSTM Game Module integrates over each game
- 1-LSTM Opponent Module integrates over each opponent
- A fully connected Decision Network makes moves

(Li & Miikkulainen 2017)

Adapting to Novel Opponents in Poker (2)

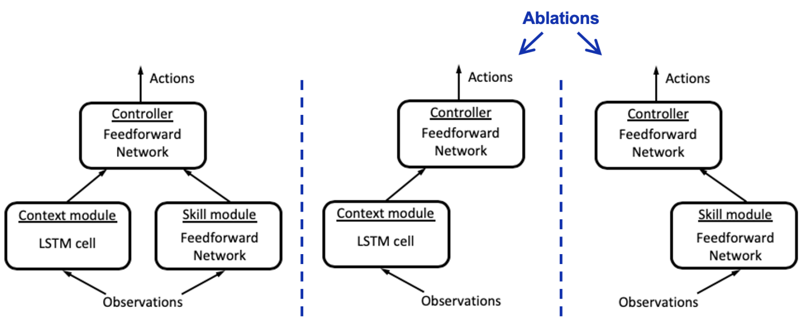
Opponent	Evolved LSTM	Slumbot 2017
Scared Limper	999	702
Calling Machine	46114	2761
Hothead Maniac	42333	4988
Candid Statistician	9116	4512
Random Switcher	8996	2102
Loose Aggressive	20005	2449
Tight Aggressive	509	284
Half-a-Pro	278	152
Slumbot 2017	19	

Adapts strategy dynamically according to opponent

- Exploits weaknesses better than Slumbot (in mBB)
- Ties against Slumbot (although evolved with only weak opponents)

Can cope robustly with novel game play

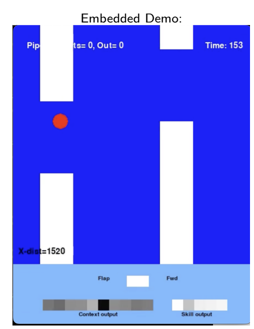
Adapting to Changing Worlds in Physical Games



Context+Skill Network (CS) | Context-only Network (C) | Skill-only Network (S)

(Tutum et al.2021)

The FlappyBall Domain

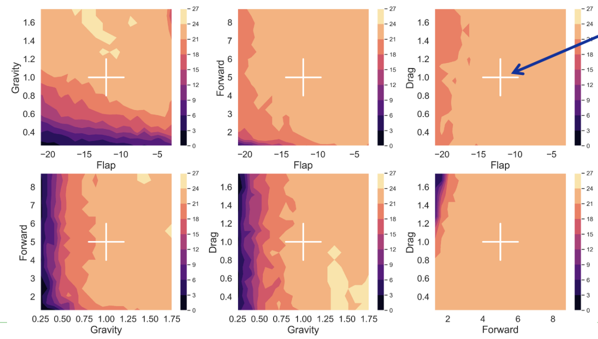


(Tutum 2021)

- Extension of Flappy Bird: FlapFwd, Drag
- **Inputs**: 6 numerical state values
  - Vertical position, distance to next pipe
  - Horizontal and vertical velocity
  - Height of the upper and lower pipe
- **Outputs**: select FlapUp, FlapFwd, glide
- **Objectives**:
  - Safety: Don't hit pipes, ceiling, ground
  - Performance: Fly fast
- **Task Variation:**
  - Strength of Gravity, Drag, FlapUp, FlapFwd

Illustration of Extrapolation

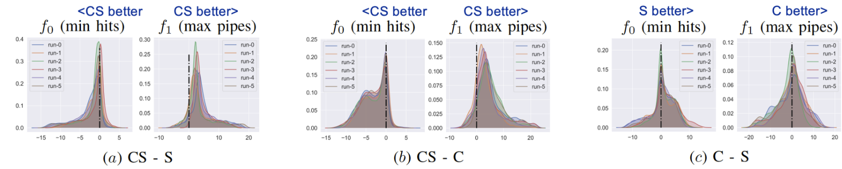
FB Performance; CS-S; Lighter is better; variation across pairs of parameters



(Tutum & Miikkulainen 2020)

- Training tasks distributed on the white cross
- Testing tasks distributed outside the cross: Require interpolation and significant extrapolation

Generalization in FlappyBall

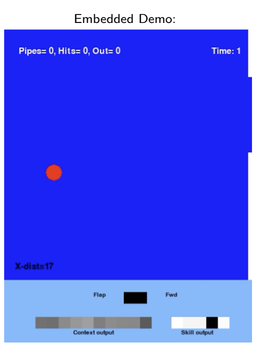


(Tutum et al.2021)

- Best networks from 5 independent evolutionary runs evaluated in new tasks
  - Effect of Gravity, Drag, FlapUp, FlapFwd varied +/- 75% (instead of +/-20% during evolution)
  - All parameters varied simultaneously; 10,000 tasks created randomly
- CS performs better than S and C in both objectives
- S is better than C in safety, the same in performance

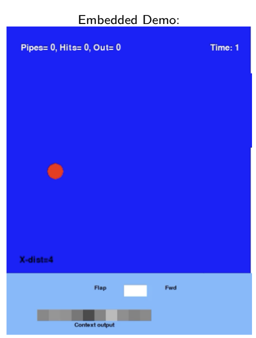
Example Behaviors in FlappyBall

- Extrapolated conditions: F=-7.0, G=0.58, Fwd=8.75, D=0.58



Context+Skill Network (CS)

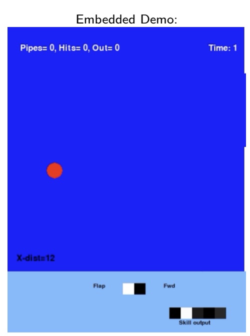
Pipes = 21
Hits = 0



Context-only Network (C)

Pipes = 15
Hits = 6

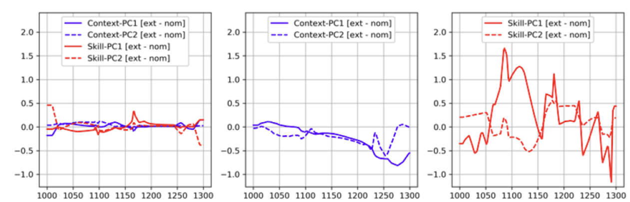
(Tutum 2021)



Skill-only Network (S)

Pipes = 16
Hits = 5

Modulation by Context

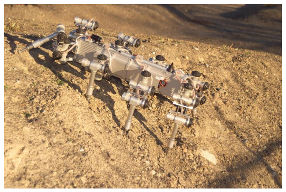


(Tutum et al.2021)

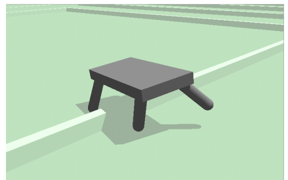
- Output of Context and Skill modules mapped to 2D with PCA
- Difference in an extrapolated task and the nominal task plotted
- Differences are smaller in CS than in C-only and S-only
  - Decision network needs to deal with less variance
  - Easier to generalize
  - CS evolves to make new tasks look more familiar
- Allows coping robustly in novel situations

Multilegged Walking

- Navigate rugged terrain better than wheeled robots
- Controller design is more challenging
  - Leg coordination, robustness, stability, fault-tolerance, ...
- Hand-design is generally difficult and brittle
- Large design space often makes evolution ineffective



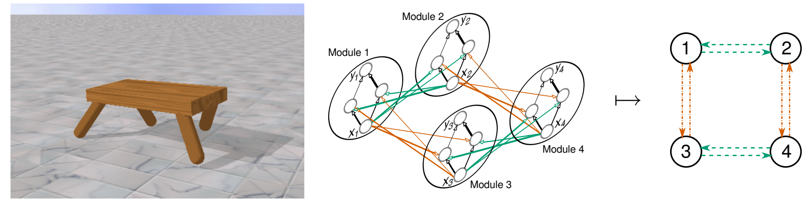
(Bluck 2009)



(Valsalam 2010)

Idea 5: Symmetry Evolution Approach

- Symmetry evolution approach
  - A neural network controls each leg
  - Connections between controllers evolved through symmetry breaking
  - Connections within individual controllers evolved through neuroevolution



(Valsalam 2010)

Versatile, Robust Gaits

- Symmetric gaits such as trotting and pacing are easier to evolve initially.
- Different gaits on flat ground
  - Pronk, pace, bound, trot
- When facing more complex terrains, symmetry-breaking allows for more adaptive gaits.
  - For example, an agent might switch from a bound gait to a trot to overcome obstacles.
  - This automatic adaptation makes control more robust across various terrains.

Embedded Demo:

Different gaits

Embedded Demo:

Obstacle field

(https://nn.cs.utexas.edu/?walkingtables-demo)

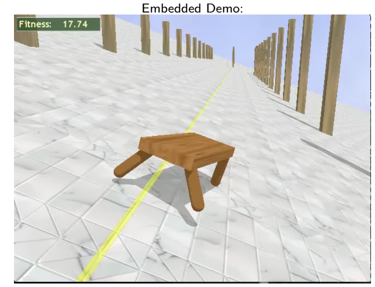
Innovative, Effective Solutions

- As challenges increase, symmetry can be broken to evolve more complex gaits.
- Asymmetric gait on inclines
  - One leg pushes up, others forward
  - Hard to design by hand

Embedded Demo:

Handcoded

Embedded Demo:

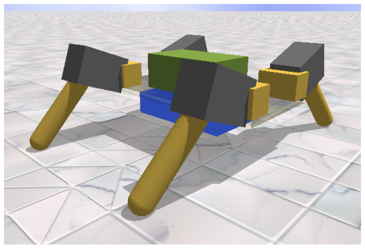


Evolved

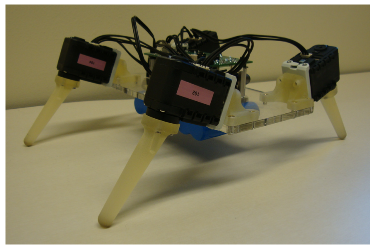
(https://nn.cs.utexas.edu/demos/enso-robots/)

Challenge: Transferring Solutions to Physical Robots

- Simulations are clean and deterministic.
- The real world is noisy, nondeterministic, and includes external factors.
- Transfer from simulation to reality is difficult but critical.



Simulated

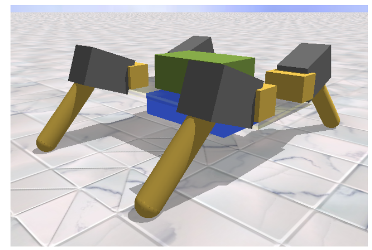


Real

(Valsalam & Miikkulainen 2012)

Transferring to Quadruped Robot

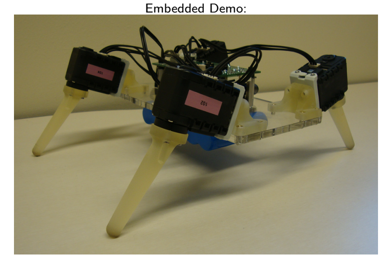
- A robot custom-built at Hod Lipson's lab (Cornell U.)
  - Standard motors, battery, controller board
  - Custom 3D-printed legs, attachments
  - Simulation modified to match
- General, robust transfer
  - Noise to actuators during simulation
  - Generalizes to different surfaces, motor speeds



Simulated

(Valsalam & Miikkulainen 2012)

Embedded Demo:



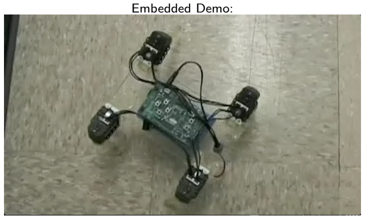
Real

(https://nn.cs.utexas.edu/demos/enso-realrobots/)

Compensating for Damage

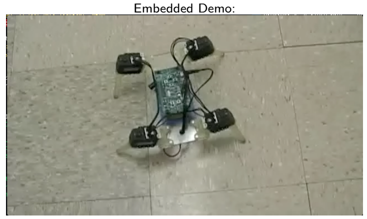
- Neuroevolution evolves controllers that can cope with imperfections and even take advantage of them.
- Example: Evolved asymmetric gait for a four-legged robot with one inoperative leg.
- This shows that neuroevolution transfers well to physical robots and can solve unexpected issues.

Embedded Demo:



Handcoded

Embedded Demo:

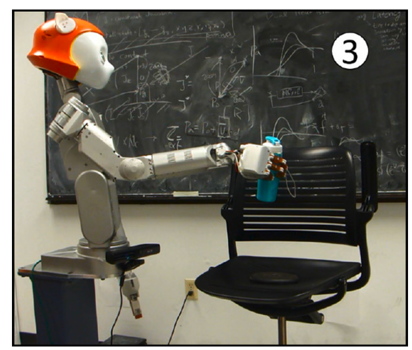


Evolved

(https://nn.cs.utexas.edu/demos/enso-realrobots/)

Simulating Physical Challenges in Neuroevolution

- Simulations can be extended with factors like wind, friction, and uneven terrain.
- Stochastic noise can be added to simulate imperfections in sensors and effectors.

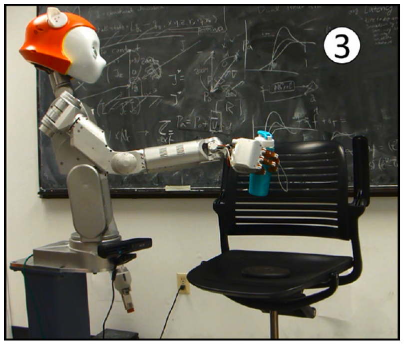


(Huang 2014)

Dreamer robot

Recent Advances in Robotics Simulators

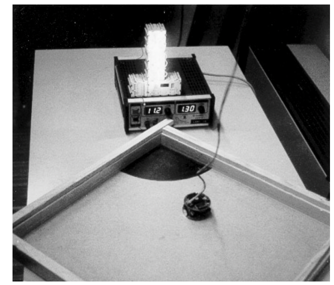
- Modern robotics simulators have become highly accurate, supporting direct transfer to physical robots.
- Example: NEAT with Graspit! simulator for robotic grasping, transferred to the Dreamer robot's Mekahand.
- Controllers can handle sensor inaccuracies, novel objects, and imprecise computation.



(Huang 2014)
Dreamer robot

Evolutionary Robotics: Evolving Control in Hardware

- Evolutionary robotics emerged in the 1990s to evolve controllers and sometimes hardware directly.
- Example: Evolving homing behavior in the Khepera mobile robot.
- Neural networks developed an internal topographic map to navigate efficiently.



(Floreano & Mondada 1996)

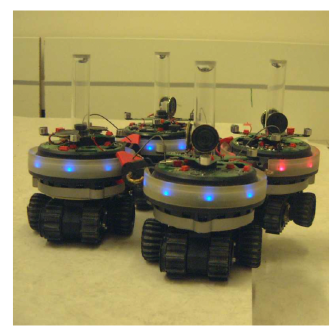
Coevolving Morphology and Control

- Neuroevolution can coevolve both the controllers and the hardware.
- Example: Locomotion starts with eel-like robots and evolves into legged designs.
- This process creates more robust gaits than evolving directly for legged robots.
- GOLEM: Hardware designs and controllers coevolved in simulation, then 3D printed and tested physically.

Demo link:
`https://youtu.be/qbUyWZZ_a9g`

Swarm Robotics: Evolving Collective Behavior

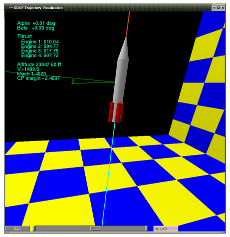
- Swarms of robots exhibit collective behavior that single robots cannot.
- Example: Robots forming a train to traverse gaps that individual robots cannot cross.
- Neuroevolution can evolve both collective and individual behaviors for the swarm.



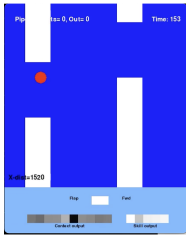
Demo link:
`https://youtu.be/i3ernrkZ91E`

Conclusion: Evolving Robust Control

- Robust control is essential for generalization and adaptability in complex environments.
- Techniques like noise injection, coevolution of controllers with teachers and problems, novelty search, explicit context representation, and symmetry help build this robustness.
- Advanced simulators, noise injection, and coevolution with hardware make transfer possible.



(Gomez & Miikkulainen 2003)



(Tutum 2021)

(https://nn.cs.utexas.edu/demos/enso-robots/)