Applying Evolutionary Computation to Robotics

Adrian Thomas Schiller

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

28 April 2014

The Big Picture

- **Problem**: A robot is faced with a problem where the solution is not immediately obvious
- **Potential Solution**: Evolutionary computation (EC) is a process which can can solve difficult problems in programming
- **Issue**: Since a robot interacts with the physical world, EC is slower by many magnitudes
- **Solution**: By using simulation and applicable evolutionary strategies, it is possible to use EC to evolve robots

Outline

1. Background
   - Evolutionary Computation
   - Artificial Neural Networks
2. Research Cases
3. Simulation
4. EvolutionaryProcess
5. Results
6. Conclusion

Evolutionary Computation

- Evolutionary Computation (EC) is a problem solving technique which mimics natural selection
- EC requires:
  - A candidate representation of a potential solution
  - A population of randomly generated candidates
  - A fitness function

Evolutionary Computation: Process

- Candidates are evaluated
- The best performing candidates are selected
- Selected candidates undergo Transformations to repopulate the population
- Process repeats until some limit is reached

Example: "One Max"

- Goal: Evolve an array consisting of the most ones from 20 candidate arrays with 10 bits

$$20 \begin{cases} [1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1] \\ [0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0] \\ \dots \\ [1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0] \end{cases}$$

Fitness Evaluation

- The fitness function $F()$ is used to calculate the fitness of each candidate
- In this case, the function counts the number of ones in the array

$$F([1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1]) = 6$$
$$F([0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0]) = 4$$
$$\ldots$$
$$F([1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0]) = 5$$

Cross-Over

- The top 10 candidates are selected to create a new population
- Top candidates are randomly selected to cross-over with one another

$$C_1 = [1, 1, 1, 0, |0, 0, 1, 0, 0, 1] \qquad C_x = [1, 1, 1, 0, 0, 1, 1, 0, 1, 0]$$

$$C_2 = [0, 0, 1, 1, |0, 1, 1, 0, 1, 0] \qquad C_y = [0, 0, 1, 1, 0, 0, 1, 0, 0, 1]$$

Mutation

- Some created candidates may undergo mutation:

$$C_x = [1, 1, 1, 0, 0, 1, 1, 0, 1, 0]$$

$$C_x' = [1, 1, 1, 0, 1, 1, 1, 0, 1, 0]$$

Repeat

- Once the population is recreated, the process repeats until eventually a limit is reached
- In this case, the limit would be iterations, time, or finding an array of all 1's

$$F([1,1,1,0,1,1,1,0,1,0]) = 7$$

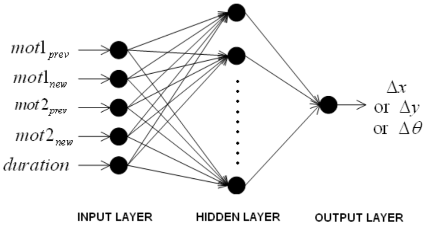
$$F([0,0,1,1,0,0,1,0,0,1]) = 4$$

$$\dots$$

$$F([1,0,1,1,0,1,0,0,1,1]) = 6$$

Artificial Neural Networks (ANN)

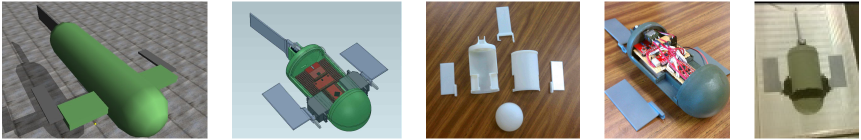
- ANNs are a collection of nodes with weighted vertices.
- The purpose of the network is to develop a functional relationship from the input to the output



Pretorius *et al*.

Outline

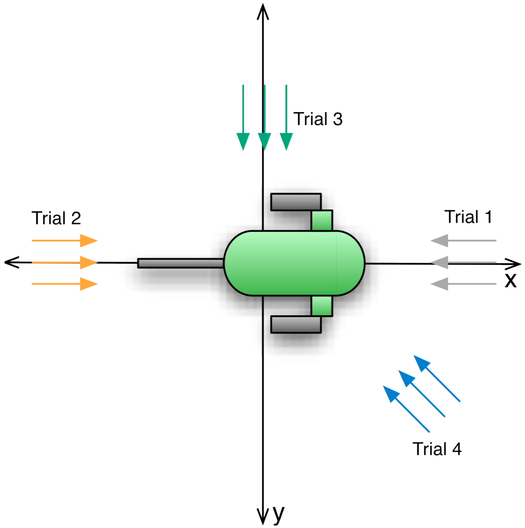
Station Keeping Robot



Moore *et al*.

- Moore *et al*. developed the station keeping robot was
- Goal: to maintain position in a body of water

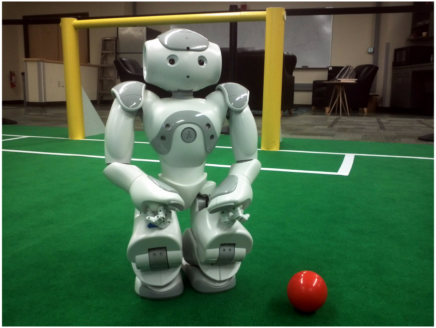
Station Keeping Robot



Moore *et al*.

Walking Robot

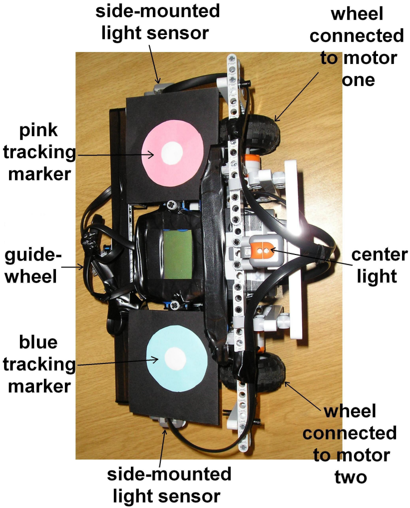
- Farchy *et al*. modified the code of the Aldebaran Nao robot
- Goal: to increase walking speed



Farchy *et al*.

Coordinate Tracking Robot

- Pretorius *et al*. created a Lego Mindstorms robot
- Goal: to evolve an internal navigation controller



Pretorius *et al*.

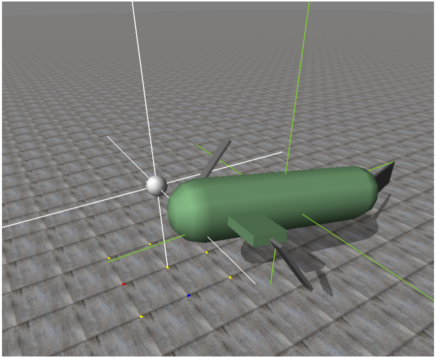
Outline

Simulation

- Defined as representing the characteristics or behaviors of one system through the use of another
- Error caused from inaccuracies of simulation is known as Transitivity

Station Keeping Robot: Simulation

- Used ODE to replicate environment and robot
- No fluid dynamics



Moore *et al*.

Walking Robot: Simulation

- Uses SimSpark (also ODE)
- Not a perfect representation

Farchy *et al*.

Coordinate Tracking Robot: Simulation

- An overhead camera captured heading/orientation of robot from arbitrary motor commands
- A testbed of 5000 commands were sent to the robot and captured by the camera, creating map of command and position

Outline

Station Keeping Robot

- Evolved a separate candidate for each of the trials
- Population size of 100 candidates
- Evolved for 2000 generations
- The entire process was repeated 25 times for each of the four trials

Station Keeping Robot: Neural Network

- Input:
  - Current 3D coordinates, (x, y, z)
  - The difference between current and desired coordinate (x, y, z)
  - The output of the previous output (servo speeds and oscillations)
- Output:
  - oscillation of the rear fin
  - speed of the left flipper
  - speed of the right flipper

Station Keeping Robot: Fitness Function

$$\text{fitness} = \sum_t (10 - d_t(x, y, z))$$

where

$$d_t(x, y, z) = \begin{cases} 10, & \text{if distance}_t(x, y, z) > 10 \\ \text{distance}_t(x, y, z), & \text{otherwise} \end{cases}$$

- After a 60 second setup phase, fitness was evaluated every 250ms for 60 seconds

Walking Robot: Parameter optimization

- Farchy *et al*. wanted to optimize several parameters to increase speed

Parameter	Description
$stepPeriod$	Number of frames to take two steps.
amp_{swing}	Amplitude of the swing calculation.
$knee$	Base of the leg lifting calculation.
$startLength$	Used in calculating initial ramp up.
v_{short}	Factor for the leg lifting calculation.
a_{short}	Amplitude of the leg lifting calculation.
ϕ_{short}	Offset of the leg lifting calculation.
v_{swing}	Factor for the swing calculation.
ϕ_{swing}	Offset for the swing calculation.
$gyro_{\text{hipPitch}}$	Body pitch factor for calculating hip pitch.
$gyro_{\text{kneePitch}}$	Body pitch factor for calculating knee pitch.
$gyro_{\text{hipRoll}}$	Body roll factor for calculating hip roll.
$gyro_{\text{ankleRoll}}$	Body roll factor for calculating ankle roll.
$scale_{\text{roll}}$	Scale for sensor value of body roll.
$offset_{\text{pitch}}$	Offset for sensor value of body pitch.
$scale_{\text{pitch}}$	Scale for sensor value of body pitch.
$fwdOffset$	Offset to have the robot walk in place.

Farchy *et al*.

Walking Robot: Fitness Functions

- Used two fitness functions for two separate runs
    - *omniWalk*

$$\text{fitness\_}g = \left(\sum_t (\text{DistanceTraveled}_t)\right) - \text{fallingPenalty}$$

    - *WalkFront*

$$\text{fitness\_}w = \text{ maxVelocity() in 15 seconds}$$

Walking Robot: Grounded Simulation Learning

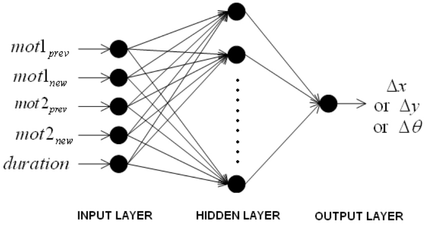
- Farchy *et al*. used Grounded Simulation Learning (GSL) when evolving candidates
- The point of GSL is to add human guidance in the evolution process
- This is done by examining the physical robot with an evolved candidate implementation, and isolating particular attributes

Coordinate Tracking Robot

- Population of 250 candidates
- Evolved for 15,000 generations
- Process repeated three times for each ANN

Coordinate Tracking Robot: Artificial Neural Network

- Navigational controller consists of three ANN:
  - The x-coordinate,
  - The y-coordinate,
  - And the angle
- Inputs of the ANNs:
  - Current Motor speeds
  - Current length of time
  - Previous Motor speeds



Pretorius *et al*.

Coordinate Tracking Robot: Fitness Function

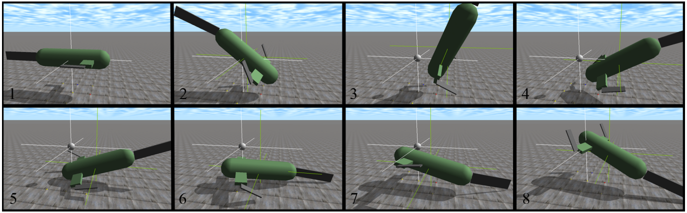
- Used the Mean Squared Error (MSE) as the fitness function

$$\text{fitness} = \frac{1}{N} \sum_{p=1}^{N} \sum_{i=1}^{O} (t_{pi} - a_{pi})^2,$$

- N is the size of the testbed (5,000)

Outline

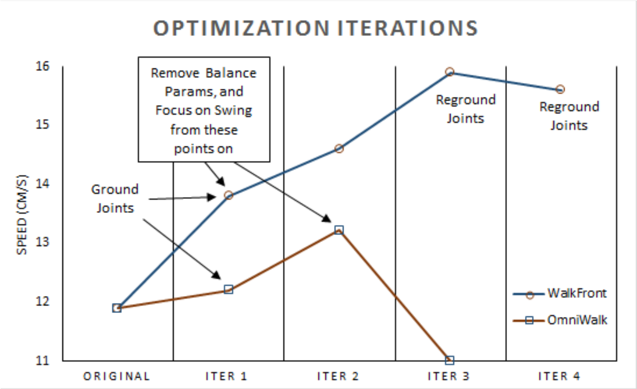
Station Keeping Robot: Results



Moore et al.

- Each trial had a candidate which successfully maintained the position
- When the flow was coming from behind, the evolved candidate would flip end-over-head to orient itself (http://y2u.be/UufbnEGFwV4)

Walking Robot: Results



Farchy *et al*.

Coordinate Tracking Robot: Results

- Each of the ANNs evolved for 12 hours
- Pretorius *et al*. noted that the results were reasonably accurate

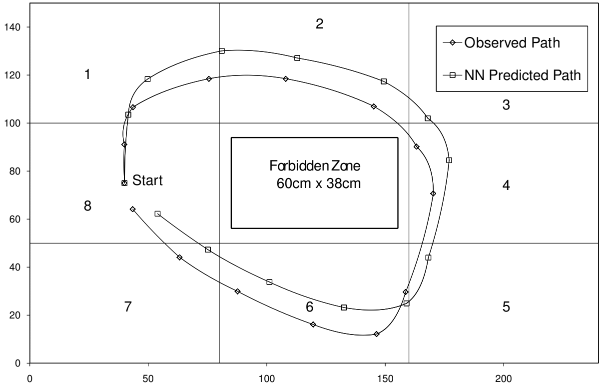
NN Simulator	Final MSE	Average absolute error
change in angle	26.412	3.585 degrees
change in y-coordinate	12.909	2.143 cm
change in x-coordinate	18.559	2.782 cm

Pretorius *et al*.

Coordinate Tracking Robot: Navigation Test

- Using the evolved ANNs, a navigation test was made for a practical application
- The test was evolved to:
  - Drive the robot in a circle around a 3x3 grid,
  - Not leave the grid or touch the middle square

Coordinate Tracking Robot: Results



Pretorius *et al*.

Outline

Conclusion

- By using a simulation, the evolutionary process can occur at a significantly faster rate
- Evolutionary robotics could be applied if:
  - the robotics problem is well defined,
  - the robot and environment can be simulated,
  - and the simulated robot's success can be quantified

Thank you to Nic McPhee, Elena Machkasova, and Alex Jarvis

Any Questions?

asdghdsa