Genetic Programming
of Autonomous Agents

Scott O'Dell

Advisors
Dr. Joel Schipper
Dr. Arnold Patton
Outline

- Introduction to Genetic Programming
- Project Summary
- Project Description
- Preliminary Results
- Schedule
Outline

• Introduction to Genetic Programming
• Project Summary
• Project Description
• Preliminary Results
• Schedule
What is Genetic Programming?

- Machine learning technique
- Evolution: Survival of the fittest
- Leverages Randomness
- Program evolved to solve a task
Randomly Generate Individual Genomes

Evaluate Fitness of Current Generation

Generation Limit or Fitness Reached?

Create New Current Generation From previous Generation

Return Individual with Highest Fitness

NO

YES
GP Flowchart

1. Randomly Generate Individual Genomes
2. Evaluate Fitness of Current Generation
3. Generation Limit or Fitness Reached?
   - YES: Return Individual with Highest Fitness
   - NO: Create New Current Generation From previous Generation
4. NO: Go back to Evaluate Fitness of Current Generation
Creating Random Genomes

- **Primitive Set**
  - **Function Set**
    - Accepts arguments
    - Returns value
  - **Terminal Set**
    - No arguments
    - Represents value
    - May have side effects

- **Requirements**
  - Sufficiency
  - Closure
Creating Random Genomes

Wall-Following Robot

- Single proximity sensor on front
- Independent wheels

Primitive Set

- Function Set
  - If-wall-ahead
- Terminal Set
  - Forward
  - Left
  - Right
Creating Random Genomes

Function Set
- If-wall-ahead

Terminal Set
- Forward
- Left
- Right

Generation 0

Genome #1
- If-wall-ahead
- Forward
- If-wall-ahead
- Right
- Forward

Genome #2
- If-wall-ahead
- Right
- Left

Genome #3, #4, #5, ...

Program Trees
Evaluating Fitness

• Fitness Function
  • Evaluates effectiveness of programs
  • Assigns fitness score
  • Must differentiate “poor” and “very poor” performance
  • Determines likelihood of “reproduction”

• Solutions optimized for fitness function
  • NO MATTER WHAT!
Evaluating Fitness

Wall-Following Robot

- Find a wall
- Follow wall w/o extra movements

Fitness Function

- Simulate robot
- Score: # of wall adjacent spaces occupied
- Limited time
GP Flowchart

- Randomly Generate Individual Genomes
- Evaluate Fitness of Current Generation
- Generation Limit or Fitness Reached?
  - NO: Create New Current Generation From previous Generation
  - YES: Return Individual with Highest Fitness
Creating a New Generation

• Selection methods
  • Fitness Proportional
    – Chance of being chosen proportional to fitness score
  • Tournament
    – Group chosen at random, highest score wins
Creating a New Generation

• Genetic Operators
  • Crossover (sexual reproduction)
  • Reproduction (asexual reproduction)
  • Mutation
Genetic Operator: Crossover

Parent 1

If-wall-ahead
forward

If-wall-ahead
right
forward

Offspring

If-wall-ahead
right

If-wall-ahead
left

Parent 2
Genetic Operator: Reproduction

If-wall-ahead

forward

If-wall-ahead

right

forward

If-wall-ahead

forward

If-wall-ahead

right

forward
Genetic Operator: Mutation

Parent

If-wall-ahead
forward
right
forward

If-wall-ahead
forward

Offspring

If-wall-ahead
right
forward

If-wall-ahead
forward

Random Tree

Right
forward

Genetic Operator: Mutation

If-wall-ahead
forward
right
forward

If-wall-ahead
forward

If-wall-ahead
forward

Right
forward
Creating a New Generation

**Generation X**

- Fitness = 5
- Fitness = 15
- Fitness = 2
- Fitness = 0
- Fitness = 7

**Generation X+1**

- Fitness = ?
- Fitness = ?
- Fitness = ?

**Crossover**

**Reproduction**

**Mutation**
Randomly Generate Individual Genomes

Evaluate Fitness of Current Generation

Generation Limit or Fitness Reached?

Create New Current Generation From previous Generation

Return Individual with Highest Fitness

YES

NO
Outline

• Introduction to Genetic Programming
• **Project Summary**
• Project Description
• Preliminary Results
• Schedule
Perimeter Maintenance

- Military application
- Control programs for autonomous agents
- Maximize perfect perimeter around base
- Maximize coverage of large perimeter
Perimeter Maintenance

---

- Base
- Starting Positions
- Guard Agents
- Enemy Agents
- Line of Sight
- Capture Areas
Outline

- Introduction to Genetic Programming
- Project Summary
  - **Project Description**
- Preliminary Results
- Schedule
Top Level

- Written in Ruby
- Easy to interface with different languages
- Processor intensive tasks, faster languages
Function Set

- prog
  - Evaluate 2 arguments sequentially
- ifGreater
  - pseudo code: if(1st > 2nd) then 3rd else 4th
  - Perform actions based on sensors
- +, -, *, /, and %
  - standard arithmetic calculation
  - develop sensor weighting systems
Terminal Set

- perim
  - returns Manhattan distance from the base,
  - make decisions based distance from base.
- forward, left, and right
  - moves agent
- i
  - random integer
  - generated during creation of genome
Simulator

- Genome controls agent
- Interactions determine fitness
- Initially, grid-based
- Later, continuous, add noise
Perimeter Maintenance

- Base
- Starting Positions
- Guards
- Enemies
- Line of Sight
- Capture Areas
Fitness Function

- Positive points
  - Guard captures enemy
  - Distance at which enemy is captured
- Negative points
  - Enemy enters perimeter
  - Guard collision
Robotic Platform

- Time permitting
- Genome interpreter
- Primitive set defines
  - Motor control routines
  - Sensor processing routine
Outline

- Introduction to Genetic Programming
- Project Summary
- Project Description
- **Preliminary Results**
- Schedule
Preliminary Results

- Completed
  - GPES block
  - Primitive set for grid navigation
  - Grid-based simulator / fitness function
Preliminary Results

- Maximizing perfect perimeter
  - Perimeter = 0 (enemy must hit base)
  - Guard sensor range = 4
  - Fitness function
    - distance from base at which enemies were captured
  - # of Generations = 50
  - Population of each generation = 1,000
  - 80% crossover, 15% reproduction, 5% mutation
## Preliminary Results

| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |
| . . . . . . . . . . | 4 | . . . . . . . . . . |

---

Bradley University - Scott O'Dell

36
Preliminary Results

- Maximizing coverage of large perimeter
  - Perimeter = 9
  - Guard sensor range = 4
  - Fitness function
    - # of enemies captured
  - # of Generations = 50
  - Generation population = 1,000
  - 80% crossover, 15% reproduction, 5% mutation
Preliminary Results

- Find optimal points, but...
  - Boring
  - Deploy and Post
  - Shouldn't they move?
- Problem: asymmetries of grid domain
  - Found points that use asymmetry as advantage
  - Cannot move and maintain advantage
Preliminary Results

From here, 7 units on the perimeter can be protected

From here, only 5 units on the perimeter can be protected
Preliminary Results

- Less boring results
  - Co-evolution of enemies
  - Continuous domain
  - Noise
Outline

- Introduction to Genetic Programming
- Project Summary
- Project Description
- Preliminary Results
- **Schedule**
Schedule

• Main goals scheduled for before spring break

• Robotic platform work placed after spring break
  • Time permitting
  • Decision based on results with continuous simulator
  • Research of platform made in parallel with simulator work
# Schedule: Completed Work

<table>
<thead>
<tr>
<th>Week of</th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 3</td>
<td>Genome Class</td>
</tr>
<tr>
<td>October 10</td>
<td>Generation Class</td>
</tr>
<tr>
<td>October 17</td>
<td>Genetic Programming Evolutionary Sequence Class</td>
</tr>
<tr>
<td>October 24</td>
<td>Grid-Based Simulator</td>
</tr>
<tr>
<td>October 31</td>
<td>Fitness Function, Terminal Set, Function Set, and initial Simulations</td>
</tr>
<tr>
<td>November 7</td>
<td>Code Refactoring</td>
</tr>
<tr>
<td>November 14</td>
<td>Capstone Project Deliverables</td>
</tr>
<tr>
<td>November 21</td>
<td>Thanksgiving Break</td>
</tr>
</tbody>
</table>
Schedule: Future Work

<table>
<thead>
<tr>
<th>Week of</th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 9</td>
<td>Enemy Co-evolution and Heterogeneous Teams</td>
</tr>
<tr>
<td>January 16</td>
<td>Continuous Simulator</td>
</tr>
<tr>
<td>January 23</td>
<td>Graphics for Continuous Simulator</td>
</tr>
<tr>
<td>January 30</td>
<td>Interface Code for Continuous Simulator and Simulations</td>
</tr>
<tr>
<td>February 6</td>
<td>Add Noise to Continuous Simulator</td>
</tr>
<tr>
<td>February 13</td>
<td>Code Refactoring</td>
</tr>
<tr>
<td>February 20</td>
<td>Simulations with Noise, Modification of Fitness Function</td>
</tr>
</tbody>
</table>
## Schedule: Future Work

<table>
<thead>
<tr>
<th>Week of</th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 27</td>
<td>Simulations with Modified Fitness Function</td>
</tr>
<tr>
<td>March 6</td>
<td>Collect Results and Create Presentation</td>
</tr>
<tr>
<td>March 13</td>
<td>Spring Break</td>
</tr>
<tr>
<td>March 20</td>
<td>Research Robotic Platform</td>
</tr>
<tr>
<td>March 27</td>
<td>Prepare Robotic Platform</td>
</tr>
<tr>
<td>April 3</td>
<td>Write Program Tree Interpreter for Robotic Platform</td>
</tr>
<tr>
<td>April 10</td>
<td>Load Evolved Program onto Robotic Platform and Debug</td>
</tr>
<tr>
<td>April 17</td>
<td>Evaluate Performance, Modify Simulator, New Simulations</td>
</tr>
<tr>
<td>April 24</td>
<td>Load Newly Evolved Program onto Robotic Platform</td>
</tr>
</tbody>
</table>
Questions?