Automatic Programming of Agents via Multi-type, Self-Adaptive Genetic Programming

Lee Spector, Hampshire College
lspector@hampshire.edu, http://hampshire.edu/lspector
Assistance by Jon Klein
Overview

Approach and Critical Elements

Technologies:
- Push, PushGP, Pushpop, Breve, SwarmEvolve

Recent Results:
- Emergence of collective/multicellular organization
- Environmental/genetic stability and adaptation
- Push/Breve integration v. 0.1

Demo:
- SwarmEvolve 1.5

Next Steps
Approach & Critical Elements

**Design Approach:** Self-adaptive, multi-type genetic programming for automated or semi-automated agent design.

**Critical Elements:** Autonomy, coordination, adaptation, control, evolution.

**Problems:** Can agents be automatically generated for complex, dynamic environments? Can agents evolve to become more adaptable to changing environments?

**Metrics:** Wait time, event response delay, agent lifetime, code parsimony/diversity, evolutionary computational effort, task completion.

**Toolkit:** Push programming language for evolved agent programs, PushGP genetic programming system, Pushpop autoconstructive evolution system.
The Push Programming Language for Evolutionary Computation

Goal: Scale up GP/agents techniques for human-competitive performance in complex, dynamic environments.

Evolve agents that may use:
- multiple data types
- subroutines (any architecture)
- recursion
- evolved control structures
- evolved evolutionary mechanisms

Push supports all of this using simple, mostly standard GP techniques.

Stack-based language with one stack per type; types include integer, float, Boolean, **code**, child, type, name.
PushGP

Evolves Push programs using (mostly) standard GP.

Multiple types handled without syntactic constraints.

Evolves modules and control structures automatically.
Autoconstructive Evolution: Pushpop

Individuals make their own children.

The machinery of reproduction and diversification (and thereby the machinery of evolution) evolves.

Radical self-adaptation.
Breve: a 3D Environment for the Simulation of Decentralized Systems and Artificial Life

Written by Jon Klein, http://www.spiderland.org/breve

Simplifies the rapid construction of complex 3D simulations.

Object-oriented scripting language with rich pre-defined class hierarchy.

OpenGL 3D graphics with lighting, shadows, and reflection.

Rigid body simulation, collision detection/response, articulated body simulation.

Runge-Kutta 4th order integrator or Runge-Kutta-Fehlman integrator with adaptive step-size control.
Breve Swarm

by Jon Klein, after Craig Reynolds

acceleration = \( p_1 \) *[away from crowding others vector]  
+ \( p_2 \) *[towards world center vector]  
+ \( p_3 \) *[average neighbor velocity vector]  
+ \( p_4 \) *[towards neighbor center vector]  
+ \( p_5 \) *[random vector]
SwarmEvolve

On-Line evolution of goal-directed swarms

Multiple species
  \( p_6^* \) [away from crowding other species vector]

Randomly moving energy sources:
  \( p_7^* \) [towards closest energy source vector].

Energy costs:
  • Colliding with one another
  • Being outnumbered (by species) in neighborhood
  • Giving birth
  • Surviving (per simulation cycle)

Upon death (energy = 0), parameters replaced with mutated version of fittest of species

Fitness metric = age * energy
SwarmEvolve 1.5

- Food consumption/growth
- Birth near mothers
- Corpses
- Food sensor, inverse square signal strength
- GUI controls and metrics
- Feeders redesigned, increased in number
- OEF correspondence increasing

[view movie]
Emergence of collective/multicellular organization

Observed behavior: a cloud of agents hovers around an energy source. Only the central agents feed, while the others are continually dying and being reborn.

Can be viewed as a form of emergent collective organization or multicellularity.

Facilitated by “birth at death location” implementation.

To appear in proceedings of Beyond Fitness: Visualising Evolution, a workshop at ALife 8.

[view movie]
Environmental/Genetic Stability and Adaptation

Food supply as a function of environmental stability and mutation rate:

<table>
<thead>
<tr>
<th>MUTATION</th>
<th>low</th>
<th>med</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>STABILITY</td>
<td>low</td>
<td>54%</td>
<td>17%</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>43%</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>55%</td>
<td>14%</td>
</tr>
</tbody>
</table>

Preliminary data (2 runs/condition) averaged over first 10,000 time steps of each run.
[Demo: SwarmEvolve 1.5]
Next Steps

Enhance complexity/realism/OEF integration.

Species-specific controls and metrics.

Structured feeder behavior; agent-responsive.

Leverage Push/Breve integration for evolution of arbitrary agent control programs and group (species) distinctions.

Integrate MIT/BBN elementary adaptive modules.

Provide “evolution” components for Taskable Agent Software Kit.
Multi-Type, Self-Adaptive Genetic Programming for Complex Applications

New Ideas

- Richly heterogeneous data can be flexibly integrated in programs produced by stack-based genetic programming.
- Explicit code manipulation allows for automatic emergence of modules and evolved program architecture.
- Self-adaptive construction of evolutionary mechanisms enhances fit to problem environments.

Impact

- Evolved agents for heterogeneous, dynamic environments.
- Broader range of applications for automatic programming technologies.
- Automatic programming with less configuration by users.

Schedule

<table>
<thead>
<tr>
<th>Port new GP systems to Beowulf cluster</th>
<th>Integration with agent environments</th>
<th>Alternative building-blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 01</td>
<td>Feb 02</td>
<td>Feb 03</td>
</tr>
<tr>
<td>Benchmarking</td>
<td>Application/analysis</td>
<td></td>
</tr>
<tr>
<td>Analysis of evolved agents</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hampshire College: Lee Spector