Evolution of Multi-Agent Systems
by multi-type, self-adaptive genetic programming

Taskable Agent Software Kit  Principal Investigators’ Meeting
February 19-20, 2003

Lee Spector
Hampshire College
lspector@hampshire.edu, http://hampshire.edu/lspector
Outline

- Guiding questions and quadchart
- Push/Breve/SwarmEvolve updates/integration
- Goal/target dynamics
- New Diversity Metrics
- Emergence of collective organization (x2)
- Integration of Elementary Adaptive Modules
Guiding Questions

Can evolution be used to help discover effective controllers for multi-agent systems?

Can evolution be used to help discover design principles for multi-agent systems?

Can evolution be used to help analyze controllers/principles for multi-agent systems?

Can we provide general evolution-based software for a Taskable Agent Software Kit?
## Evolution of Multi-Agent Systems

by multi-type, self-adaptive genetic programming

**Hampshire College**

<table>
<thead>
<tr>
<th>Design Problem/Solution Approach</th>
<th>Experiment/Analysis Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Mobile, potentially evasive goals</td>
<td>• Baseline: infinite stability or goal random walk</td>
</tr>
<tr>
<td>• Self-adaptive evolution of agent controllers</td>
<td>• Baseline: infinite reward per goal</td>
</tr>
<tr>
<td>• Push language for evolved agent programs</td>
<td>• Baseline: multiple input gradient descent</td>
</tr>
<tr>
<td>• Breve 3D/physical simulation environment</td>
<td>• Baseline: no communication/coordination</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Coverage (depletion of “food” supply)</td>
<td>• General purpose agent evolution software</td>
</tr>
<tr>
<td>• Response delay for categorical changes</td>
<td>• Emergence of collective behavior (mult. forms)</td>
</tr>
<tr>
<td>• Individual lifetimes, parsimony, diversity</td>
<td>• Analysis of stability/adaptation interactions</td>
</tr>
<tr>
<td>• Information and energy sharing</td>
<td>• New diversity metrics</td>
</tr>
</tbody>
</table>
Push

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

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

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

Simplifies 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.
Autoconstructive Evolution

The ways in which programs reproduce and diversify are themselves products of evolution.

Agents control their own mutation rates, sexuality, and reproductive timing.
Push/Breve Integration

- C-language Push interpreter plugin (original was Lisp, Java versions by others).
- Push interpreter per Breve agent.
- Breve agents can perform/evolve arbitrary computations.
- Push/Breve callbacks implement sensors/effectors.
- XML specification for Push standardization.
SwarmEvolve 1.0-1.5

Acceleration = \( p_1 \) [away from crowding others vector] 
+ \( p_2 \) [to world center vector] 
+ \( p_3 \) [average neighbor velocity vector] 
+ \( p_4 \) [to neighbor center vector] 
+ \( p_5 \) [random vector] 
+ \( p_6 \) [away from other species vector] 
+ \( p_7 \) [to closest energy source vector]

Genotype = \([p_1, p_2, p_3, p_4, p_5, p_6, p_7]\).

Various energy costs (collisions, species outnumbered, etc.).

Upon death (energy = 0), parameters replaced with mutated version of fittest (max age \(*\) energy) of species.
SwarmEvolve 1.5

- Food consumption/growth, redesigned feeders (goals/targets).
- Birth near mothers.
- Corpses.
- Food sensor, inverse square signal strength.
- GUI controls and metrics.
SwarmEvolve 2.0

- Behavior (including reproduction) controlled by evolved Push programs.
- Energy conservation.
- Facilities for communication, energy sharing.
- Enhanced user feedback (e.g. diversity metrics, agent energy determines size).
Goal/Target Dynamics

Implemented:
- Linear drift
- Random walk

Planned:
- Evasive
- Flocking
- Many parameters

Potentially: integrate with Alphatech problem generator/TTAS
SwarmEvolve GUI

Stability (1/P(drift)) = 1000

Iteration Statistics
- Iteration: 364
- Population size: 105
- Food supply: 13.1280%

Epoch Statistics
- Epoch: 3
- Births: 186
- Spontaneous births: 0
- Deaths: 164
- Reproductive mutations: 177
- Rate: 0.38193
- Diversification: 15.0376
- Energy to similar: 75
- Energy to dissimilar: 489
- Servos: 3093

Automatic camera control
New Diversity Metrics

\[ \text{diversity}(P) = \frac{\sum_{i \in P} \left| \{j \in P : \Delta(i,j) > \delta \} \right|}{|P| - 1} \]

- Average, over all agents, of proportion of remaining population considered “other”
- Genotypic (e.g. code, code size)
- Phenotypic (e.g. color, behavior, signals)
- Reproductive/developmental
Collective Organization

- Multicellularity in SwarmEvolve 1.0.
- Energy sharing in SwarmEvolve 2.0.
Multicellularity

- 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

- Peripheral agents: defensive organs.

- Central agents: digestive/reproductive organs.
Multicellularity
Energy Sharing

- Example evolved strategy: Reckless goal-seeking + sharing.

- Functional instructions of evolved code:
  ( toFood feedOther myAge spawn randF )

- Accelerates directly toward nearest goal, feeds others, and turns random colors.

- Evolved mutation regime: rate $\propto 1/\text{age}$.

- High goal coverage, low lifetimes.
Energy Sharing
Other Strategies
Servo/EAM Integration

- Now: single EAM per agent.
  Potentially: any number, any architecture.

- Now: servo EAM only.
  Potentially: all EAM types.


- Initial indications: high utility.
Evolved Servo Use

Example:

(spawn ( ( ( V* VECTORX myLocation ( CODECHILD ( servo OR ) ( setServoGain mutate ) ) ) ) * ( DUP MAKEVECTOR ( foodIntensity ) ) ) ) ( CONS ( myHue ( crossover FALSE ) ( OR ) ( setServoSetpoint otherProgram friendHue ) ) ) + ( QUOTE ) ) ( VECTORX V* ) ( ( NULL NTH ( AND ) ) ) ( CODECHILD randF ( V- foodIntensity ) VECTORZ ) ( DO* FALSE ( myAge crossover NOOP ( feedFriend ) ) ( V* ( NULL ) ) ) )
OEF Correspondence

- Target dynamics ↔ energy source dynamics.
- Several OEF metrics apply (e.g. task service rates, delays).
- Some divergences incidental (e.g. floating targets, specific vehicle control parameters).
- Some divergences necessary (e.g. no evolution without death) but many “lessons learned” should generalize.
Group Linkages

- MIT/BBN re: Elementary Adaptive Modules.
- UNM re: modularity and evolvability.
- UMass re: mining of SwarmEvolve data.