“Graduated Embodiment for Sophisticated Agent Evolution and Optimization”
Position Paper on Scalability of Evolutionary Computation

Arthurine Breckenridge, Mark Boslough, and Michael Peters

Sandia National Laboratories
PO Box 5800, MS 1004, Albuquerque, NM 87185
[arbreck, mbboslo, mpeters] @sandia.gov

GECCO Workshop
June 26, 2004 – Seattle Washington, USA
Goal: To develop methods for graduated evolution for embodied agent optimization.

Method: Evolve agent behaviors in a hierarchical fashion by staging fitness through multiple levels of fidelity.
Project Accomplishments

• **Implemented Staged Optimization**
  – Showed that staged methods exhibit improved scaling (as expected).

• **Implemented Pruning**
  – Pruned trees demonstrated reduced computational load (as expected).

• **Combined hand-coded and machine-generated behaviors**
  – Can build algorithms that are truly human-machine collaborations.

• **Converted GP code to C++**
  – Allows for improved modularization; shifts burden off user.

• **Incorporated GP behaviors into UMBRA**
  – High fidelity environment for high fidelity behaviors.
Adaptive Waypoints

This encapsulation is intended to bridge the gap between toy problems and real problems.

Our method to develop levels of integration: Vertical, Horizontal, Co-Evolution, Collective Behavior.
General implementation concept: Use GP to design “compute” step

\[ t=0 \]

Sense \rightarrow Compute \rightarrow Act

\[ t=1 \]

Sense \rightarrow Compute \rightarrow Act

\[ t=2 \]

Sense \rightarrow Compute \rightarrow Act

\[ t=3 \]

Sense \rightarrow Compute \rightarrow Act
Classification of Building Blocks

Horizontal Sequential

Sense → Compute → Act

Horizontal Parallel

Sense → Compute → Act

Vertical

Sense → Compute → Act
Example: Horizontal Sequential

What is my local environment?

- Land
- Sea
- Air

Crawl → Swim → Fly → Locomotion
Example: Horizontal Parallel
Example: Vertical

Sense

Where do I want to go?

How do I get there?

Act
Example: Combined

Where do I want to go?

How do I get there?

Act

Sense

Crawl

Swim

Fly
Benefits of Converting code from C to C++: Modularization

C++ reduces and modularizes changes in code

Change optimization in C code

- Code that stays the same
- Code that changes

Change optimization in C++ code

More changes in more places mean more bugs and longer development time
Multiple levels are needed:
both horizontal and vertical

*Modeling Building Blocks according to Funge, Terzopoulos, 1999*
We are developing tools to visualize evolved behaviors
Benchmark problem optimization approaches

Single Tree Ponder

Five Tree No-Ponder

Staged Optimization

Five Tree Ponder

With Tree Pruning
Results for benchmark problem

Scaled Time (Node Hours)

Fitness

- 5 tree with policy (no ponder)
- 5 tree ponder with policy
- 2 tree ponder (albatross 3)

Graphs for different configurations:
- 1.1.15g
- 1.1.15a
- 1.1.12
- 1.1.15c
- 1.1.15d
- 1.1.15e
Results for benchmark problem

![Graph showing fitness vs. scaled time for different optimization techniques.]

- 5 tree STAGED OPTIMIZATION (with pruning)
- 5 tree STAGED OPTIMIZATION (without pruning)

Legend:
- 1.1.15g
- 1.1.15a
- 1.1.12
- 1.1.15c
- 1.1.15d
- 1.1.15e
- 3.6
- 3.7
Real-world example:
Image data collection

Sense Compute Act (Machines)

Pixel Neighborhood

If Value < Target Value
MOVE (C)

Pereceptual Cognitive Behavioral (Humans)
Graduated Embodiment example: adaptive waypoint

Allows Vertical, Horizontal, Co-Evolutionary, Collective Behavior
Graduated Embodiment example: adaptive waypoint

We have chosen this method to develop usable behaviors

Low-Fi Evolution

Mid-Fi Tuning

Hi-Fi Tuning

Co-evolutionary
Fine Tuning

Fixed interface

Adaptive Waypoint

Cheesy Flight model

6-DOF FDM Autopilot

Full 6-DOF FDM-Autopilot Atmosphere

This is the essence of Graduated Embodiment
Next: Apply GP to Umbra’s hard engineering

Can the adaptive waypoint concept be generalized to n-dimensional parameter space??

Simple target on terrain

Netfires target seeking

Netfires at launch
Why are we developing evolutionary computing technology?

- Well-suited for robotics applications.
- Simulation pushes Umbra “multiple-fidelity” capability.
- Provides context for intelligent machine systems.
- Likely to provide insight into cognition processes.
- Uses behavioral biomimetics to derive insight from nature.
Umbra Application of Genetic Programming
Running on 2X2 Tile in VIEWS Corridor Building 880/A1

2056 X 2048 Resolution taken with 720X 480 Camera