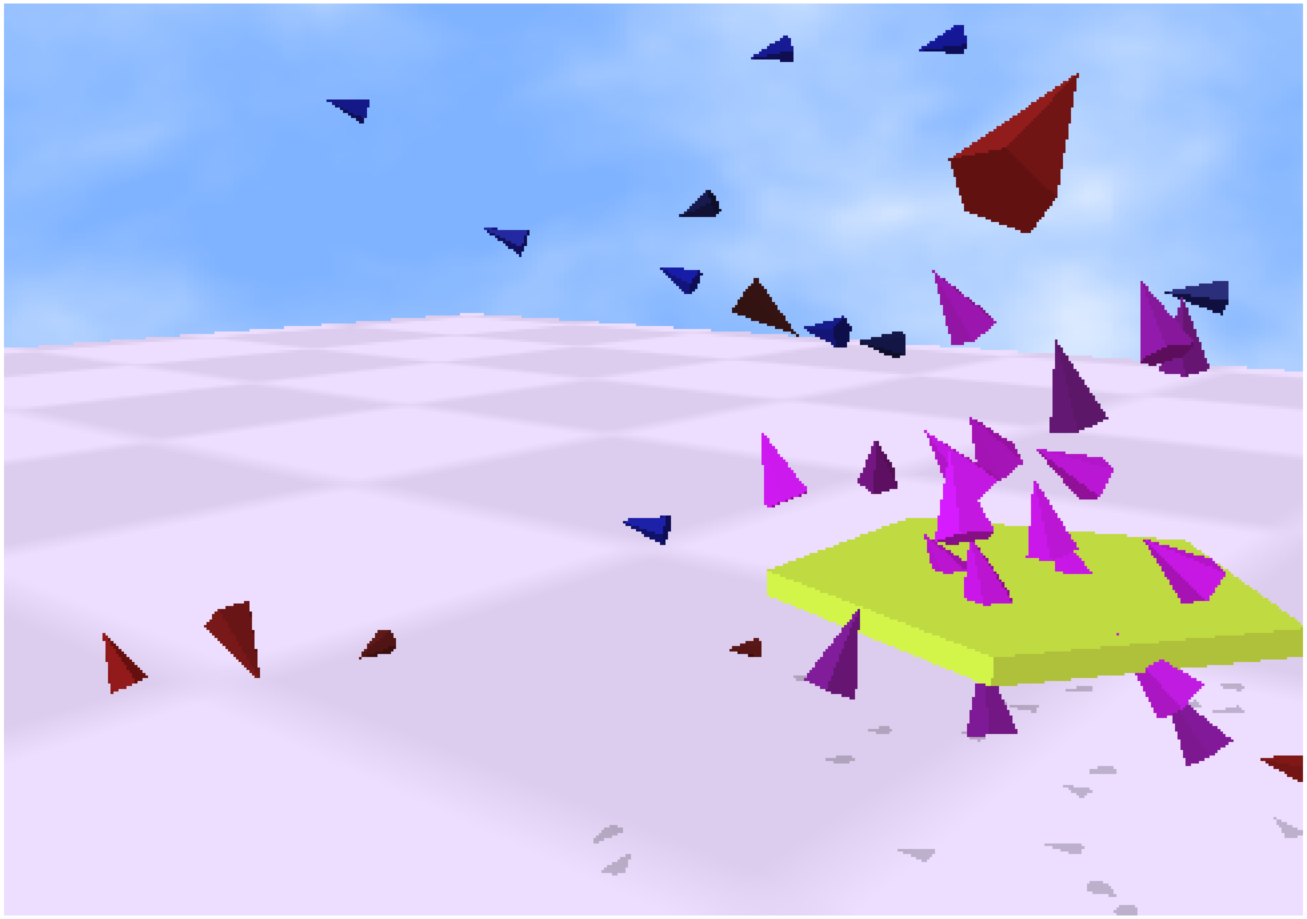
Multi-type, Self-Adaptive Genetic Programming as an Agent Creation Tool

Lee Spector, Hampshire College

Alan Robinson, Hampshire College/UCSD

lspector@hampshire.edu, http://hampshire.edu/lspector

Thanks also to Jon Klein.



Overview

Goals

Technologies: Push, PushGP, Pushpop, Breve (by Jon Klein)

Results:

- PushGP:
  - Evolved transport network agents
  - Evolved “Opera” agents
  - Confirmed/extended Van Belle/Ackley effect
- Pushpop:
  - Reliable auto-diversification
- Breve:
  - Evolved goal-directed 3D swarms **[DEMO]**

Future

Goals

1. Provide technologies for the automated production of agents for complex, dynamic environments.

2. Develop self-adaptive (self-configuring) evolutionary computation systems in the service of Goal #1.

3. Investigate general properties of self-adaptive evolutionary systems using the technologies developed for Goal #2.

The Push Programming Language for Evolutionary Computation

Designed for the expression of evolving programs within an evolutionary computation system.

Simplifies the evolution of 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.

Push

Stack-based, like Forth or Postscript

Multiple stacks, one for each type

Types are hierarchical

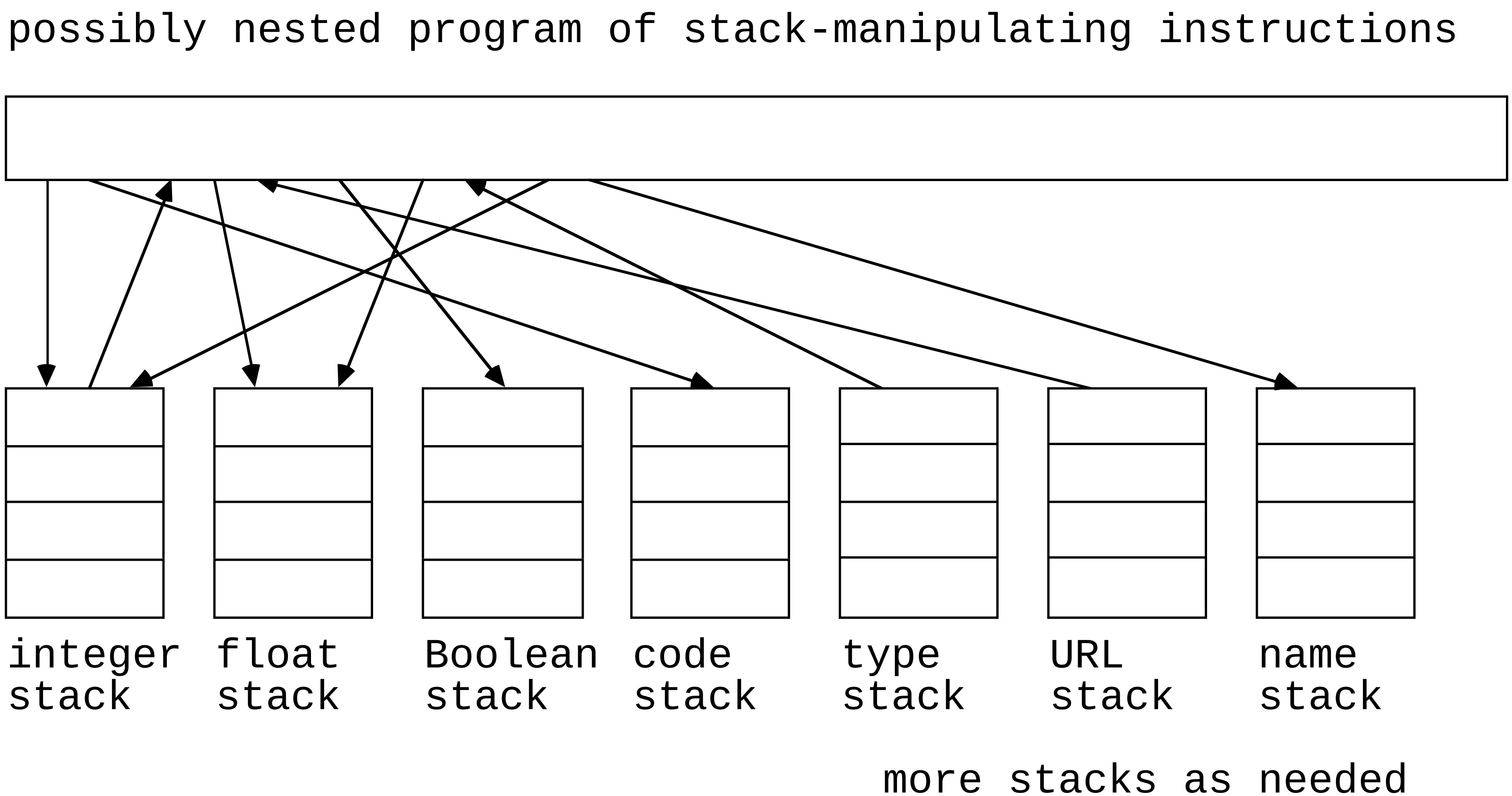
Type constants on a **type** stack/bottom

Missing argument? NOOP

Code type/stack -> advanced features

Runtime resource limits

Push Architecture



Push Examples

```
(integer 2 3 +)

(integer 2 3 + float 2.72 3.14 +)

(2 3 2.72 3.14 integer + float +)

(2.72 integer 2 3.14 3 + float +)

((integer) (2 (3)) +)

(code quote (integer 2 3 +) do)
```

Factorial in Push

```
(quote (pop 1)
 quote (code dup
        integer dup
        1 - do *)
 integer dup 2 < if)
```

Factorial with Names

```
(code
   quote (quote (pop 1)
          quote (integer dup 1 -
                 code factorial get do
                 *)
          integer dup 2 < if)
   factorial set
   factorial get do)
```

The Push Type Hierarchy

```
- push-base-type: dup, pop, swap, rep, =[boolean],
                  set[name], get[name], convert[type],
                  pull[integer], noop
 - number: +, -, *, /, >[boolean], <[boolean]
   - integer: rand, pull, /
   - float: rand
 - boolean: not, and, or, nand, nor, rand
 - expression: quote, car, cdr, cons, list, append, subst,
               container, length[integer], size[integer],
               atom[boolean], null[boolean], nth[integer],
               nthcdr[integer], member[boolean],
               position[integer], contains[boolean],
               insert[integer], extract[integer],
               instructions[type], perturb[integer],
               other[integer], other-tag[float],
               elder[integer], neighbor[integer],
               rand[integer]
   - code: do, do*, if[boolean], map
   - child:
 - type: rand
 - name: rand
```

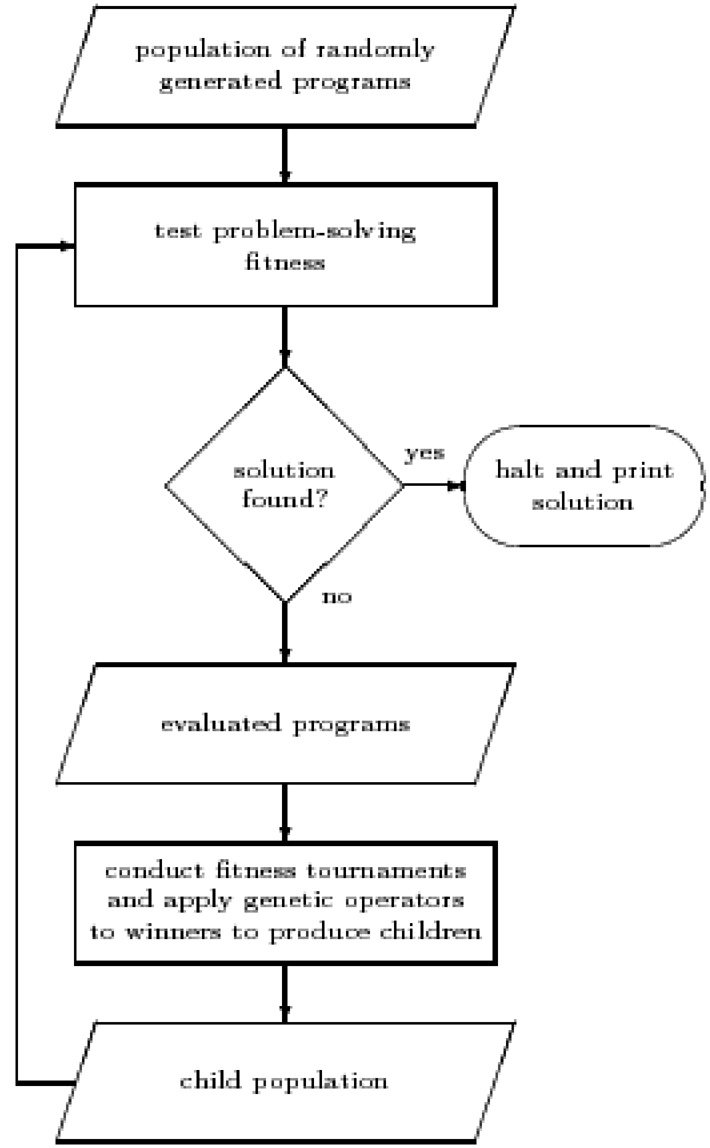
Inheritance, multi-stack access, subsets

PushGP

Evolves Push programs
using (mostly) standard GP.

Multiple types handled without
syntactic constraints.

Evolves modules and
control structures automatically.



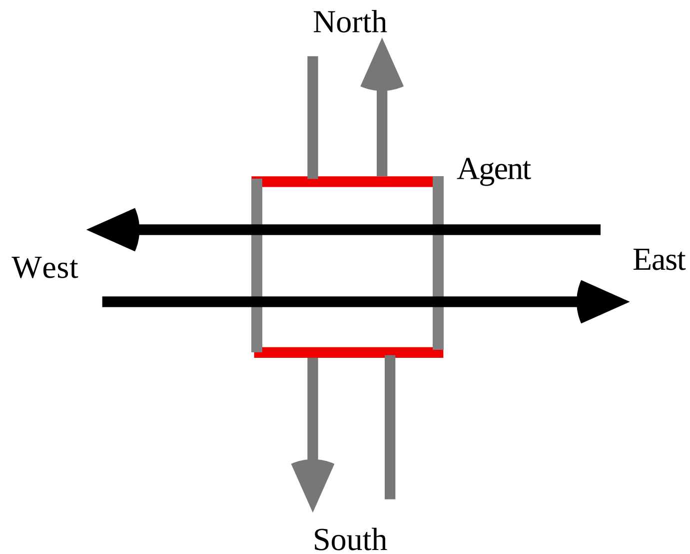
Size Control via Size-Fair Genetic Operators

With Raphael Crawford-Marks, proceedings of *GECCO 2002*.

Table 3: Results for 6-Bit Multiplexor, sorted by computational effort.

Crossover Method	Mutation Method	Successful Runs	Average Solution Size	Average Size Limit Replications (Gen. 25)	Average Size Limit Replications (Gen. 49)	Computational Effort
Fair	Fair	30/100	19.80	0.46	28.56	1870000
Fair	Node Sel	36/100	27.58	71.41	428.67	1885000
Naive	Fair	32/100	27.53	127.00	410.82	2080000
Naive	Node Sel	26/100	30.96	389.41	749.47	2520000
Fair	Naive	26/100	32.27	623.75	1388.20	2635000
Node Sel	Naive	23/100	37.57	1375.40	1725.29	2835000
Node Sel	Fair	26/100	27.96	325.13	673.92	3120000
Naive	Naive	26/100	37.92	972.08	1519.34	3200000
Node Sel	Node Sel	18/100	31.11	697.06	1014.76	4320000

Evolved Transport Network Agents

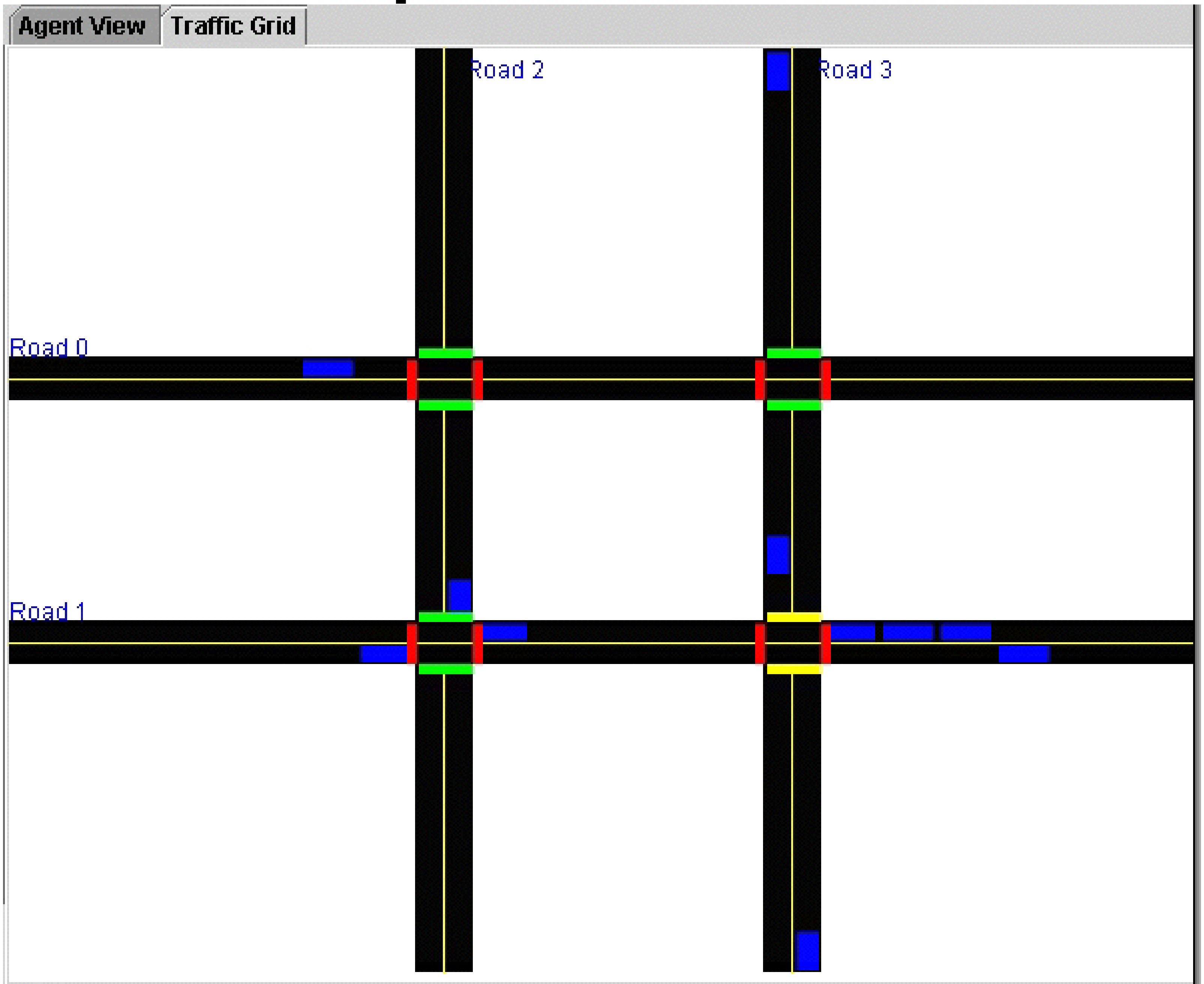


Collaboration with Selfridge/Feurzeig/Benyo (MIT/BBN).

Four linked flow corridors per intersection.

“N/S/E/W” arbitrary; nothing rectilinear/2D in underlying network transit simulation.

BBN Transport Network Simulator



Evolved TNAs: Control/Metrics

Agent controls “green time” in one direction.

Metrics available to agent:

- Green time
- Average windowed wait
  - Per corridor
  - “Global”
- Maximum wait
  - Per corridor
  - “Global”

Evolved TNAs: Fitness/Cases

Minimize global wait time.

Average over many different flow density/variability configurations.

Fitness Case	North Bound	South Bound	East Bound	West Bound
1	.25	.25	.25	.25
2	.1	.1	.9	.8
3	.1	.1	.8	.8
4	.1	.1	.7	.7
5	.1	.1	.6	.6
6	.1	.1	.5	.5
7	.1	.1	.4	.4
8	.1	.1	.3	.3
9	.1	.1	.2	.2
10	.1	.1	.1	.1
11	.9	.9	.1	.1
12	.8	.8	.1	.1
13	.7	.7	.1	.1
14	.6	.6	.1	.1
15	.5	.5	.1	.1
16	.4	.4	.1	.1
17	.3	.3	.1	.1
18	.2	.2	.1	.1
19	.1	.1	.1	.1
20	.9	.9	.1	.1
21	.3	.3	.5	.5
22	.9	.01	.01	.01

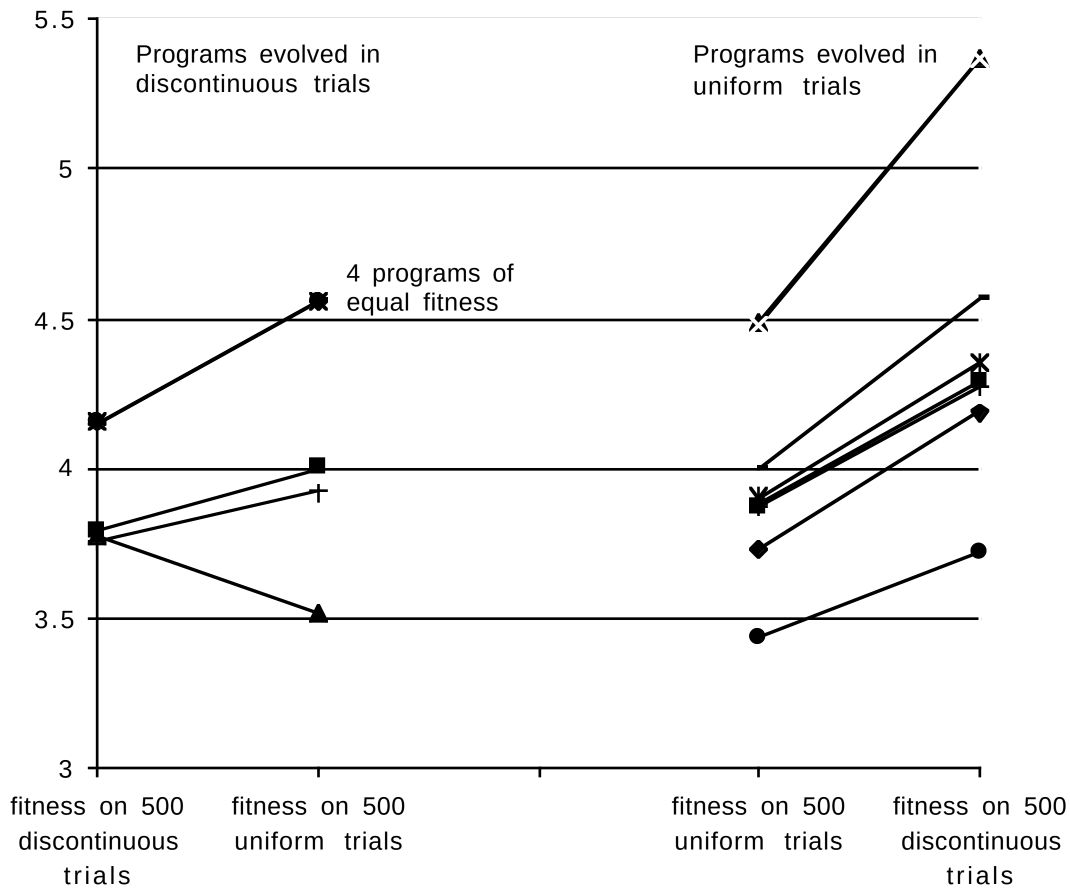
Evolved TNAs: Agent

NewTimeGreen = OldTimeGreen
+ WindowedAverageWait(northCorridor)
+ WindowedAverageWait(southCorridor)
+ WindowedAverageWait(eastCorridor)
+ WindowedAverageWait(westCorridor)
+ MaxWait(southCorridor)
+ MaxWait(westCorridor)
- MaxWait(northCorridor)

Evolved TNAs: Performance

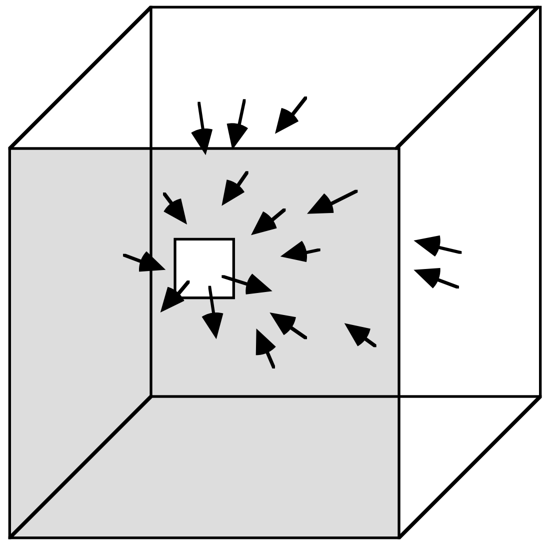
Behavior	Fitness (summed average wait values across all fitness cases)
Evolved agent	1.3
Constant time-green of 0.5	3.1
Constant time-green of 0.2	3.0
Constant time-green of 0.8	2.4

Discontinuous/Uniform Evolutionary Environments



Programs evolved in uniformly variable environments were more immediately reactive to changes in their environments.

Evolved “Opera” Agents



Collaboration with Crespi/Cybenko/Russ/Santini (Dartmouth).

Evolve decentralized and coordinated 3D navigation.

Addition of “vector” data type improves performance.

With Alan Robinson, to appear in *Late-Breaking Papers of GECCO 2002*.

Confirmation/Extension of Van Belle/Ackley Effect

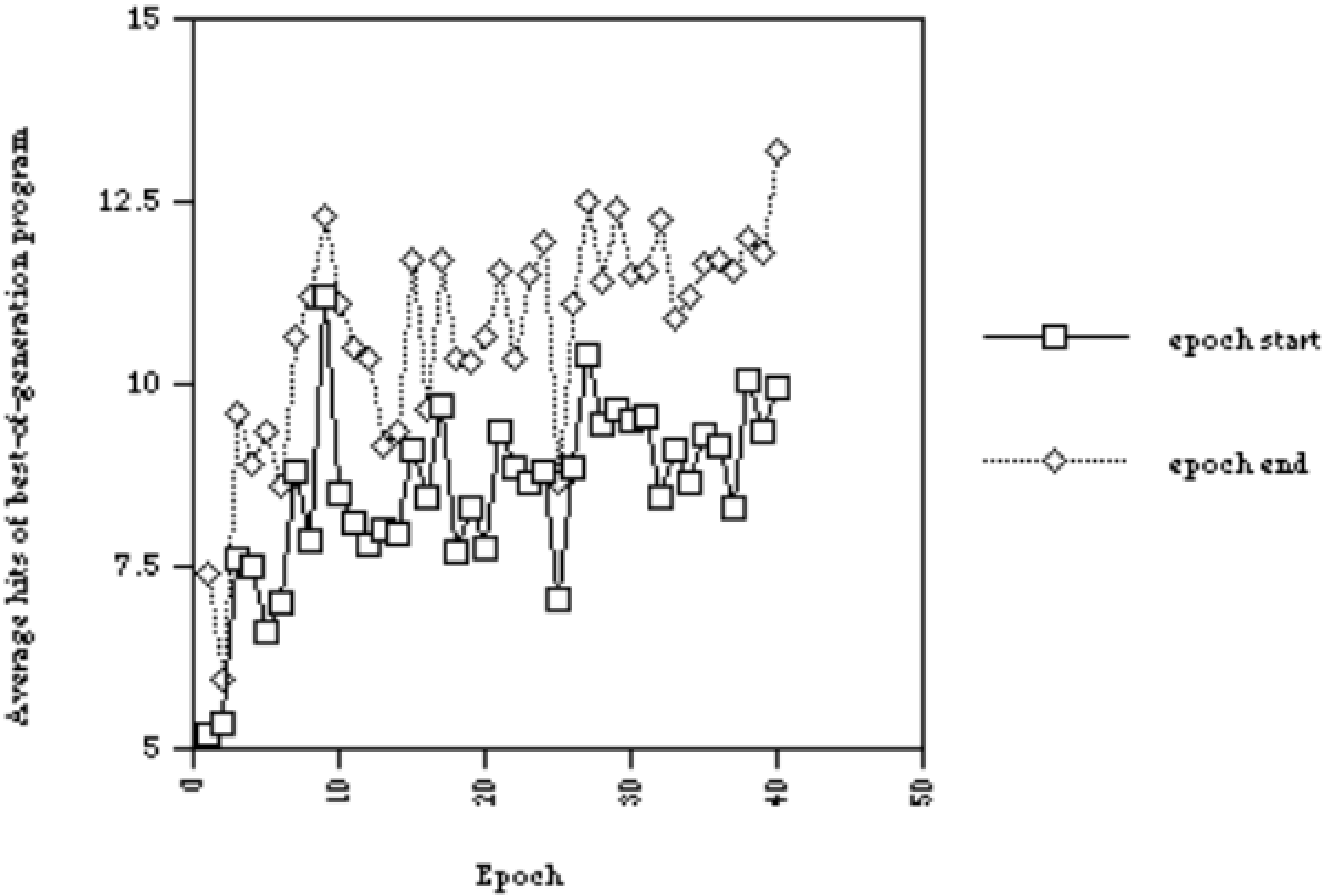
Collaboration with Van Belle/Ackley (UNM).

Evolution in a dynamically changing environment (A*sin(A*x), with randomly changing A). Modularity allows adaptation via isolation of constant/variable features of the environment.

Van Belle/Ackley Effect Parameters

PARAMETER	VALUE
Population size	1000
Tournament size	5
Max generations	200
Fitness cases	50
Mutation %	45
Crossover %	45
Reproduction %	10
Mutation operators	standard, fair (0.25), perturb (50)
Crossover operators	standard, fair (0.25), uniform
Instruction set	ephemeral-random-integer, ephemeral-random-float, ephemeral-random-boolean, ephemeral-random-symbol, convert, =, rep, swap, pop, dup, max, min, >, <, /, *, −, +, pulldup, pull, exp, log, cos, sin, not, or, and, nth, list, cons, cdr, car, quote, map, if, do*, do, integer, float, boolean, type, code

Van Belle/Ackley Effect Results

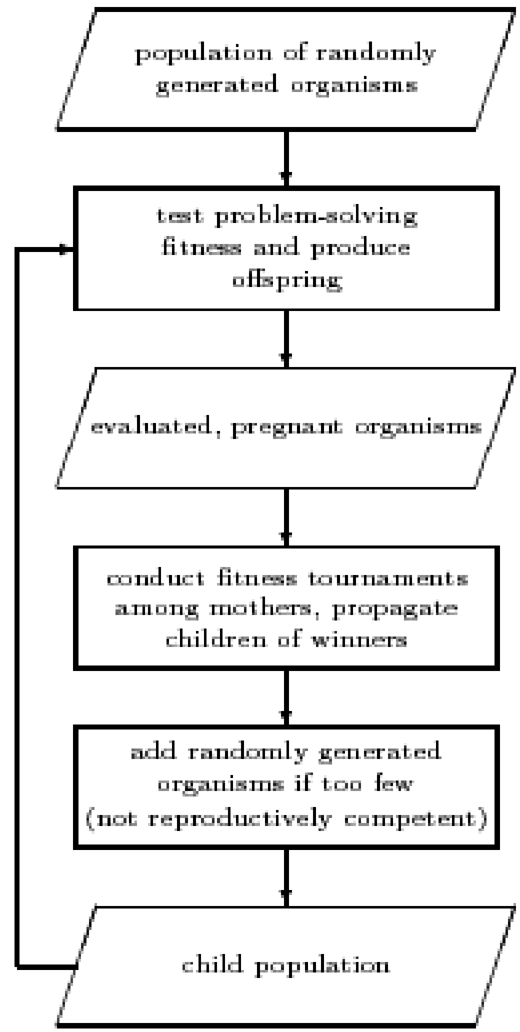


Autoconstructive Evolution: Pushpop

Individuals make their own children.

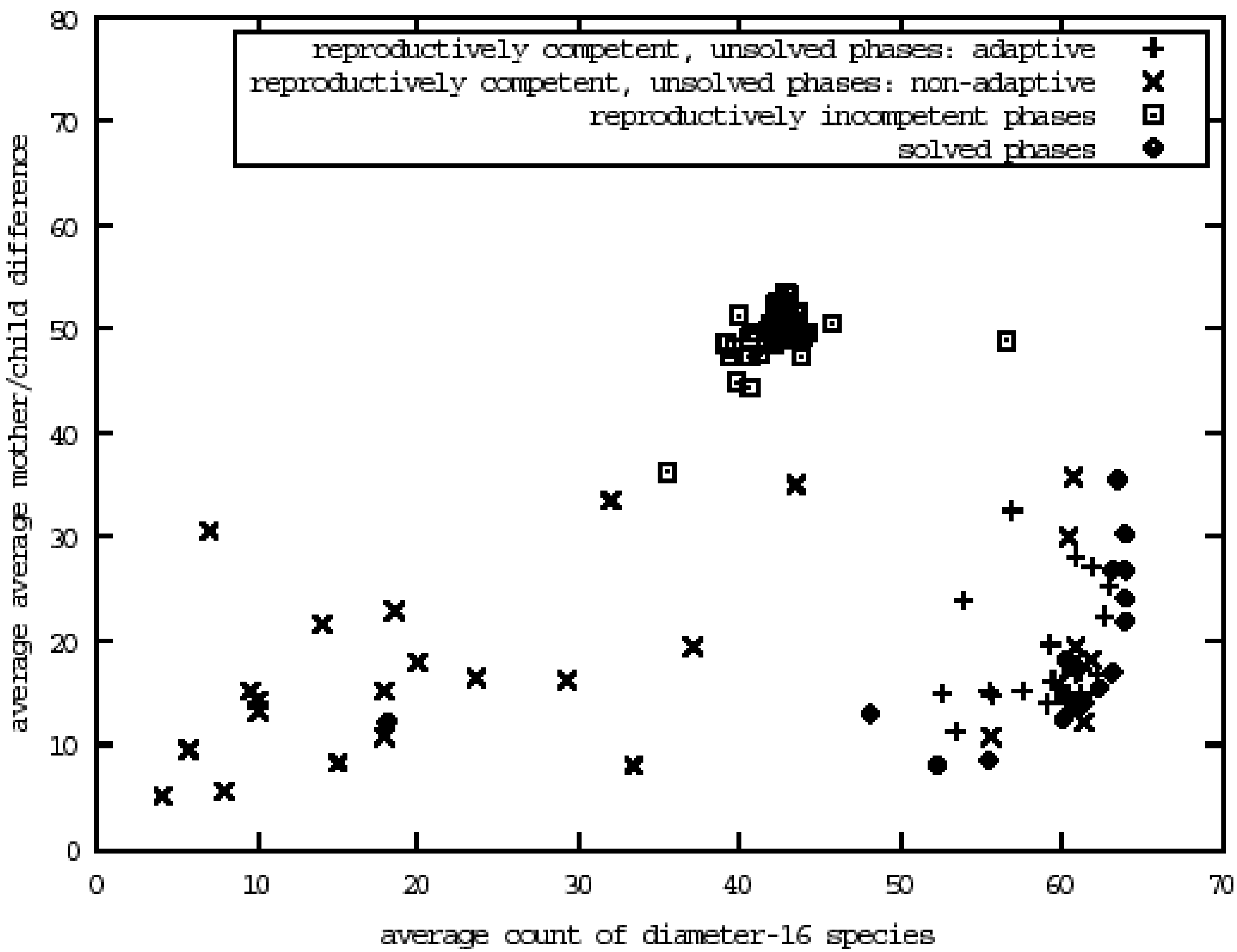
The machinery of reproduction and diversification (and thereby the machinery of evolution) evolves.

Radical self-adaptation.



Adaptive Populations of Pushpop Programs are Reliably Diverse

Partial explanation for emergence of diversifying reproduction in biology.



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.

$$\begin{aligned}
\text{acceleration} = \; & p_1\text{*[away from crowding others vector]} \\
& + p_2\text{*[towards world center vector]} \\
& + p_3\text{*[average neighbor velocity vector]} \\
& + p_4\text{*[towards neighbor center vector]} \\
& + p_5\text{*[random vector]}
\end{aligned}$$

On-Line Evolution of Goal-Directed Swarms

Changes to Breve/swarm:

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

Evolving Goal-Directed Swarms Demo

[“flock nicely” presets, randomize and evolve]

Future

Enhance complexity/realism of environments for agent evolution.

Build capability for evolution of arbitrary (Push) agent programs into 3D Breve environment.

Integrate MIT/BBN elementary adaptive modules into agent evolution system.