Unwitting Distributed Genetic Programming

via Asyncronous JavaScript and XML

Jon Klein and Lee Spector
School of Cognitive Science, Hampshire College

This work was supported by NSF Grant No. 0308540

Introduction

- GP takes time
- Fortunately, GP scales well
- More fitness test evaluations = more results
- Lots of unused computation out there

Distributed GP

- GP is embarrassingly parallel: use more machines for more fitness tests
- Several existing systems/frameworks for distributed GP, including the Distributed Genetic Programming Framework (Weise & Geihs, 2006)
- Existing systems for other evolutionary computing paradigms

Problems with Distributed GP

- Require client-side software installation
- Require client-side motivation
- Require client-side permission

Unwitting Distributed Genetic Programming

- Solve GP problems without (you) running any fitness tests
- All fitness tests run, *unwittingly*, by unaffiliated web users
- A.K.A. "parasitic computing"— see *Nature* 412, August 2001
- See also *GECCO-2007* workshop paper by Merelo et al.

AJAX

- Asynchronous JavaScript + XML = interactive web applications
- Send data back and forth between client and server from a fully loaded webpage
- Buzzwordy!
- Light-weight, ubiquitous, *generally* innocuous
- “Web 2.0”: Google Apps, Digg, Amazon use AJAX for interactive web pages

Push3 Language

- Designed for evolutionary computation
- Multi-type stack based language
- Very simple syntax
- Unusually powerful semantics
- Easy to implement

Push3

- **KEY IDEA**: Stack-based postfix language with one stack per type: integer, float, vector, Boolean, name, code, exec, ...
- Syntax-independent handling of multiple data types.
- Code and exec stacks support use and evolution of subroutines (any architecture), recursion, evolved control structures, and meta-evolutionary mechanisms.

Push3 Syntax

program ::= instruction | literal | (program*)

Push3 Semantics

- To execute program P:
  1. Push P onto the `EXEC` stack.
  2. While the `EXEC` stack is not empty, pop and process the top element of the `EXEC` stack, E:
     (a) If E is an instruction: execute E (accessing whatever stacks are required).
     (b) If E is a literal: push E onto the appropriate stack.
     (c) If E is a list: push each element of E onto the `EXEC` stack, in reverse order.

Sample Push3 Instructions

Stack manipulation instructions (all types)	POP, SWAP, YANK, DUP, STACKDEPTH, SHOVE, FLUSH, =
Math (INTEGER and FLOAT)	+, −, /, *, >, <, MIN, MAX
Logic (BOOLEAN)	AND, OR, NOT, FROMINTEGER
Code manipulation (CODE)	QUOTE, CAR, CDR, CONS, INSERT, LENGTH, LIST, MEMBER, NTH, EXTRACT
Control manipulation (CODE and EXEC)	DO*, DO*COUNT, DO*RANGE, DO*TIMES, IF

A Simple Push3 Program

```
( 2 3 INTEGER.* 4.1 5.2 FLOAT.+ TRUE FALSE
BOOLEAN.OR )
```

Resulting stacks:

```
BOOLEAN STACK: ( TRUE )

CODE STACK: ( ( 2 3 INTEGER.* 4.1 5.2
              FLOAT.+ TRUE FALSE BOOLEAN.OR
              ) )

FLOAT STACK: ( 9.3 )

INTEGER STACK: ( 6 )
```

Scrambled

```
( 4.1 2 ( TRUE ) ( 3 5.2 ( FALSE ) ) FLOAT.+
BOOLEAN.OR INTEGER.* )
```

Resulting stacks:

```
BOOLEAN STACK: ( TRUE )

CODE STACK: ( ( 4.1 2 ( TRUE ) ( 3 5.2
( FALSE ) ) FLOAT.+ BOOLEAN.OR INTEGER.* ) )

FLOAT STACK: ( 9.3 )

INTEGER STACK: ( 6 )
```

Better Living through Code Manipulation

You get *ALL* of this for *FREE!* (or at least real cheap)

- Subroutines (with evolved architecture)
- Iterators (standard and evolved)
- Recursion and combinators
- Evolved control structures
- Evolved genetic operators

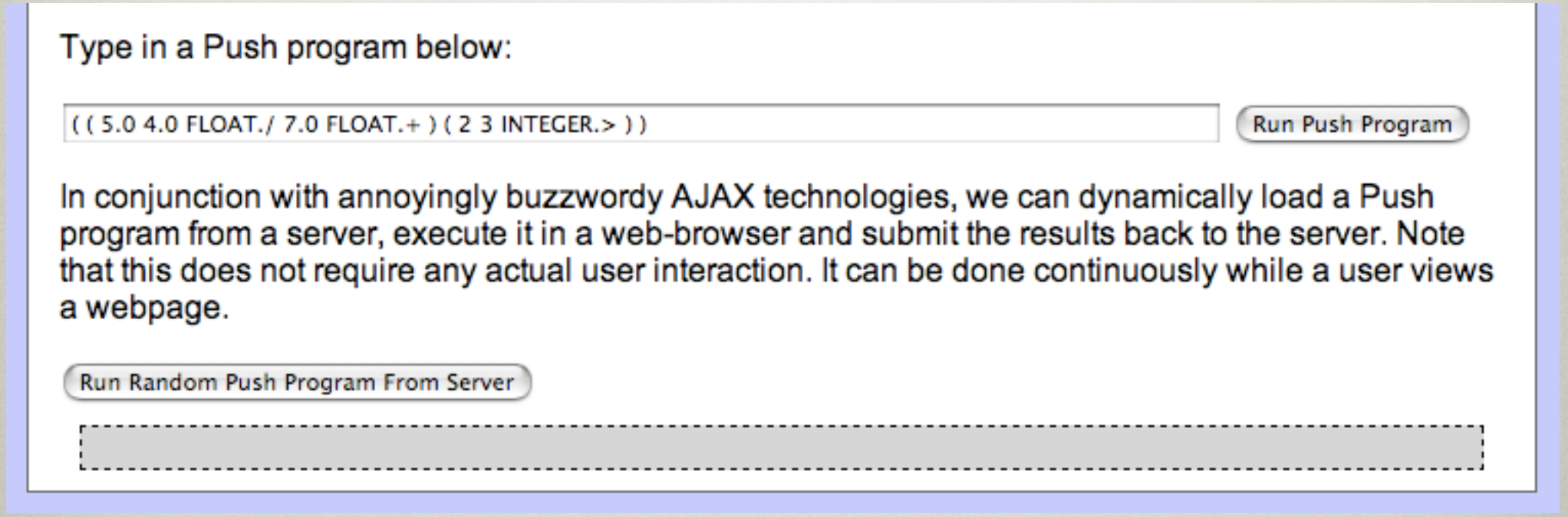
PushScript

- Lightweight (<30k) JavaScript Push implementation
- Supports all standard Push3 stack types, most Push3 instructions
- Runs in most web broswers including Internet Explorer, Firefox, Safari, iPhone*
- **Requires NO software installation: loads automatically with webpage**

*which is Safari anyway, but it's just fun to say that our system runs on the iPhone

Interactive Demo

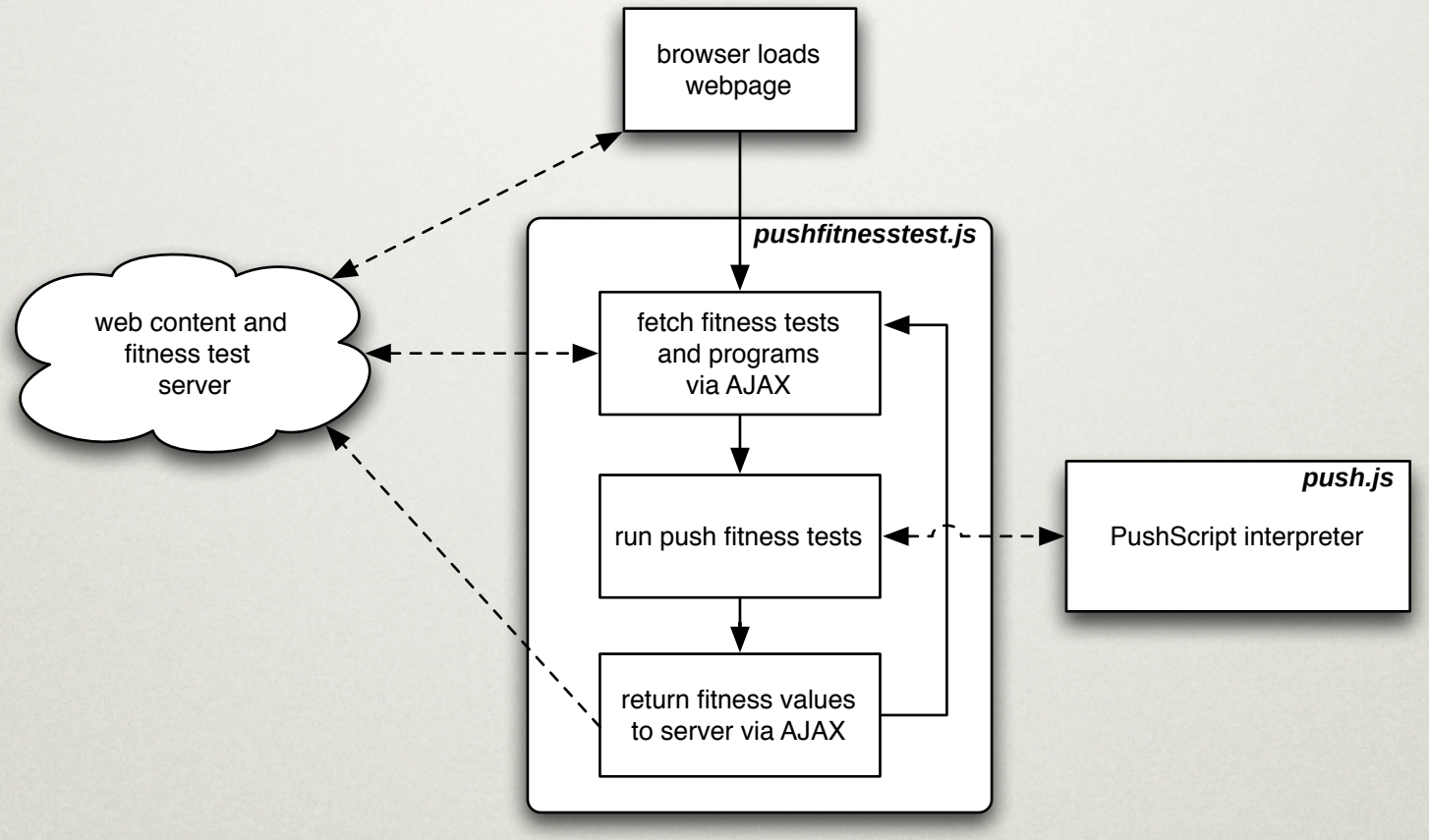
http://www.spiderland.org/PushScript



Server-Side Code

- **Lightweight server implementation to avoid server-side bottlenecks**
- New fitness cases sent as XML via PHP script
- Data collection via PHP scripts
- New generations generated via *breve* script, using the C++ Push3 implementation

Process


browser loads webpage
pushfitnesstest.js
web content and fitness test server
fetch fitness tests and programs via AJAX
run push fitness tests
return fitness values to server via AJAX
push.js
PushScript interpreter

Problems

- 5 simple symbolic regression problems we've studied previously
- Deployed on a low traffic website (*breve:* http://www.spiderland.org/breve)
- **Proof of concept question: can unwitting computation be used to solve our GP problems without (us) running fitness tests, and no voluntary user participation?**

Parameters

Problems	1. $8 * x * x * x + 3 * x * x + x$ 2. $x * x * x + x * x + x$ 3. $x * x * x - 2 * x * x - x$ 4. $x * x * x * x + x * x * x + x * x + x - 8$ 5. $x * x * x * x * x * x - 2 * x * x * x * x + x * x - 2$
Input (x) values	1-8
Fitness	sum of absolute value of errors
Crossover rate	40%
Fair mutation rate	40%
Deletion mutation rate	5%
Duplication rate	15%
Population size	2000
Maximum program size	50
Tournament size	7
Ephemeral random constants	integers from -10 to 10
Instruction set (Dec. 10 problems 1, 2 and 3)	`FLOAT.+, FLOAT.-, FLOAT.*, FLOAT./, FLOAT.POP, FLOAT.DUP FLOAT.SWAP, INPUT`
Instruction set (Dec. 10 problems 4 and 5, Jan. 15 all)	`INTEGER.+, INTEGER.-, INTEGER.*, INTEGER./, INTEGER.POP, INTEGER.DUP, INTEGER.SWAP, INPUT`

Results

- Yes! We can solve symbolic regression problems.
- Very, very slowly.
- Several hours to solve a problem which takes a few minutes on the desktop
- **Probably not practical for simple problems, but...**

... IT CAN BE PRACTICAL IF:

- ... to compliment local computation on more open-ended problems
- ... local fitness computation takes longer than about .5 seconds per fitness test (on a low-traffic server)
- ... the system is deployed on a very high traffic website

Stealing?

- Short answer, “yes” with an “if”, long answer, “no”, with a “but”
- No more computation than typical AJAX applications
- ... but we’re using the computation for our own benefit

Future Work

- New problem classes: implement domain-specific functions for use with PushScript.
- New problems: which is to say, “real” problems.
- Faster fitness test evaluation: PushJava? PushFlash?