Evolving Data Structures with Genetic Programming

W. B. Langdon

Dept of Computer Science

University College London.

email W.Langdon@cs.ucl.ac.uk

Introduction

- GP increasingly important to AI
- Programs = algorithms + **data structures**
- GP needs to evolve both

Summary of Presentation

- Benefits of evolving data structures
- Results – Stack, Queue
- Initial List results (not in paper)
- Where we go from here

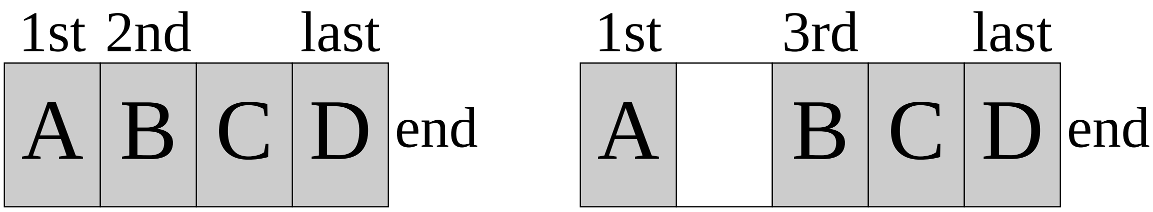
Why Evolve Data Structures?

- Trends in software: abstraction++

Human	GP
Machine code	
Assembler	
Fortran iv, etc.	Single tree
Structured code	ADF, GP-Lib, GP-AR
Object Oriented	**Abstract Data Types**

- Functional and data abstraction
- $\rightsquigarrow$ easier to evolve
- Evolve data structures

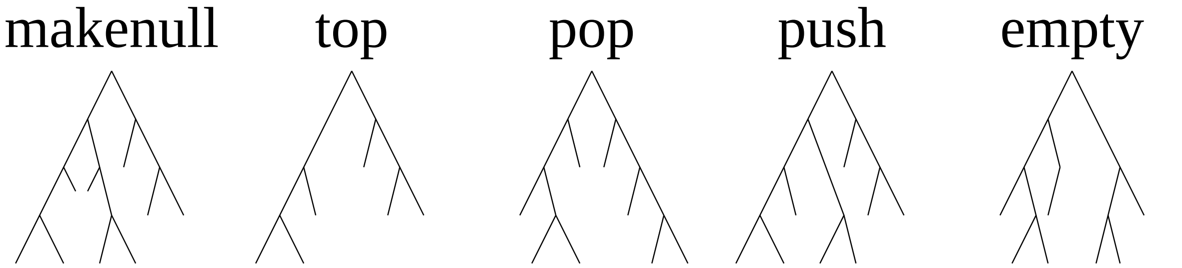
List



- 10 operations:

function	names
initialize	Makenull
update	Insert, Delete
cursor	Next, Previous
absolute	First, End
access	Retrieve, Locate (find)
output	Printlist

Multi-tree Architecture

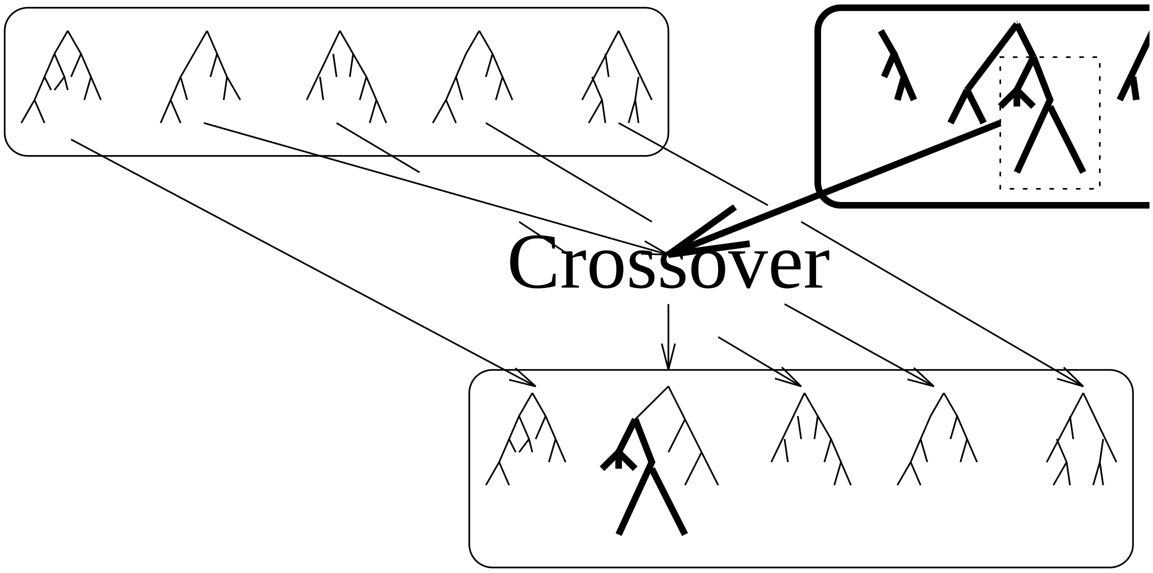


One individual within the population
(Stack - 5 trees)

- Each individual in population implements all 10 operations

- Each operation coded by own tree
- Tree for each ADF

Multi-tree Crossover



- Crossover in one tree at a time
- One offspring
- Tree chosen at random
- Directed choice (list only)

Terminals and Functions

Arithmetic	$+$, $-$	all
Constants	0, 1, max (10)	all
Inputs	ARG1, ARG2, arg1	
Variable	aux1	
	Set_Aux1	
Memory	read, write, swap	
Output	print	prt_adf
Control	PROG2	
	forwhile, i0	adf1
Procedures	adf1, FUNC	
	ins_adf, del_adf, loc_adf, prt_adf	FUNC
	End, First, Next^{ref}, Prev^{ref}	

Forwhile

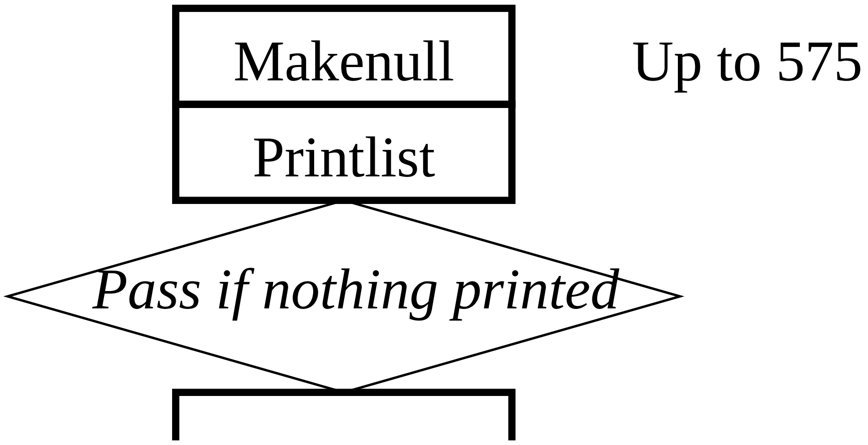
- 3 arguments: start, end, loop_code
  Exit forwhile i0>end or loop_code= 0 or timeout (32). Return i0.
- forwhile cannot be nested
- i0 only in loop_code part of forwhile

Automatically Defined Functions

- 2 arguments: integer, function
- FUNC: call 2^{nd} argument
- 4 adfs (insert, delete, locate, printlist)
- Hidden adfs access own arg1 and ARG1 of original caller

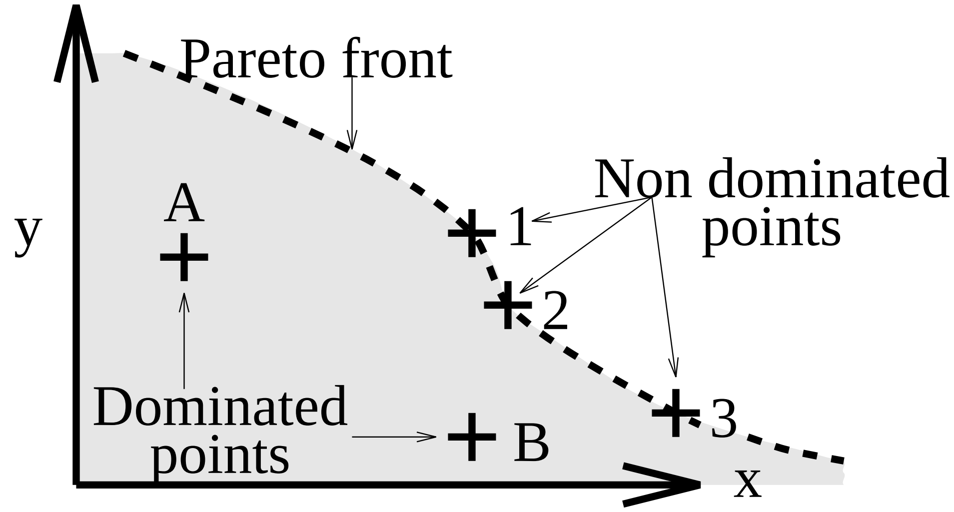
Fitness Function

- Run on fixed test sequences (20)



- Reward each operation which contributes to passing each check
- Penalties for excess CPU and memory
- Fitness has 12 separate components

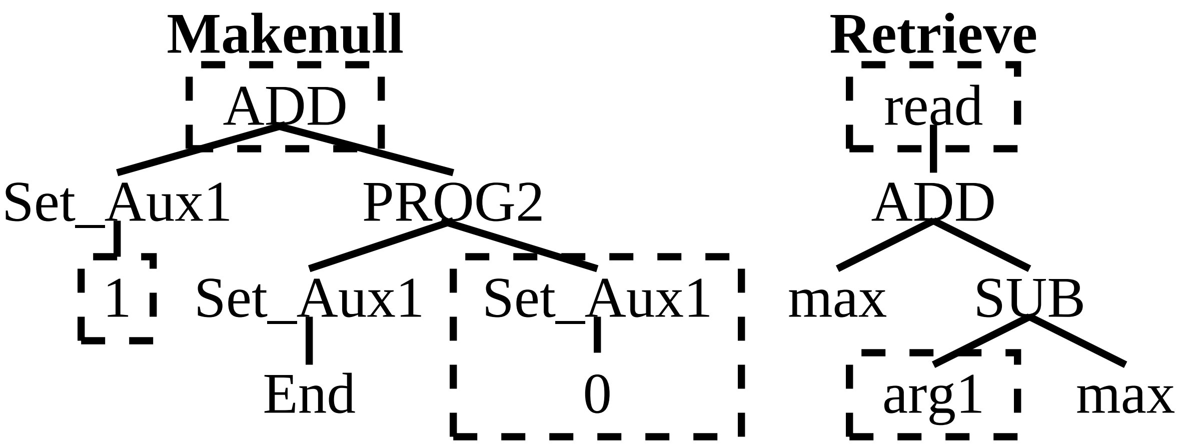
Pareto Fitness (queue and list)

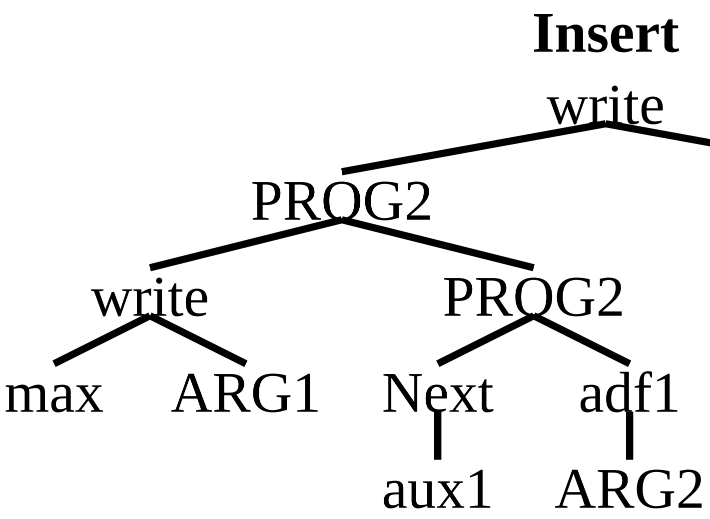


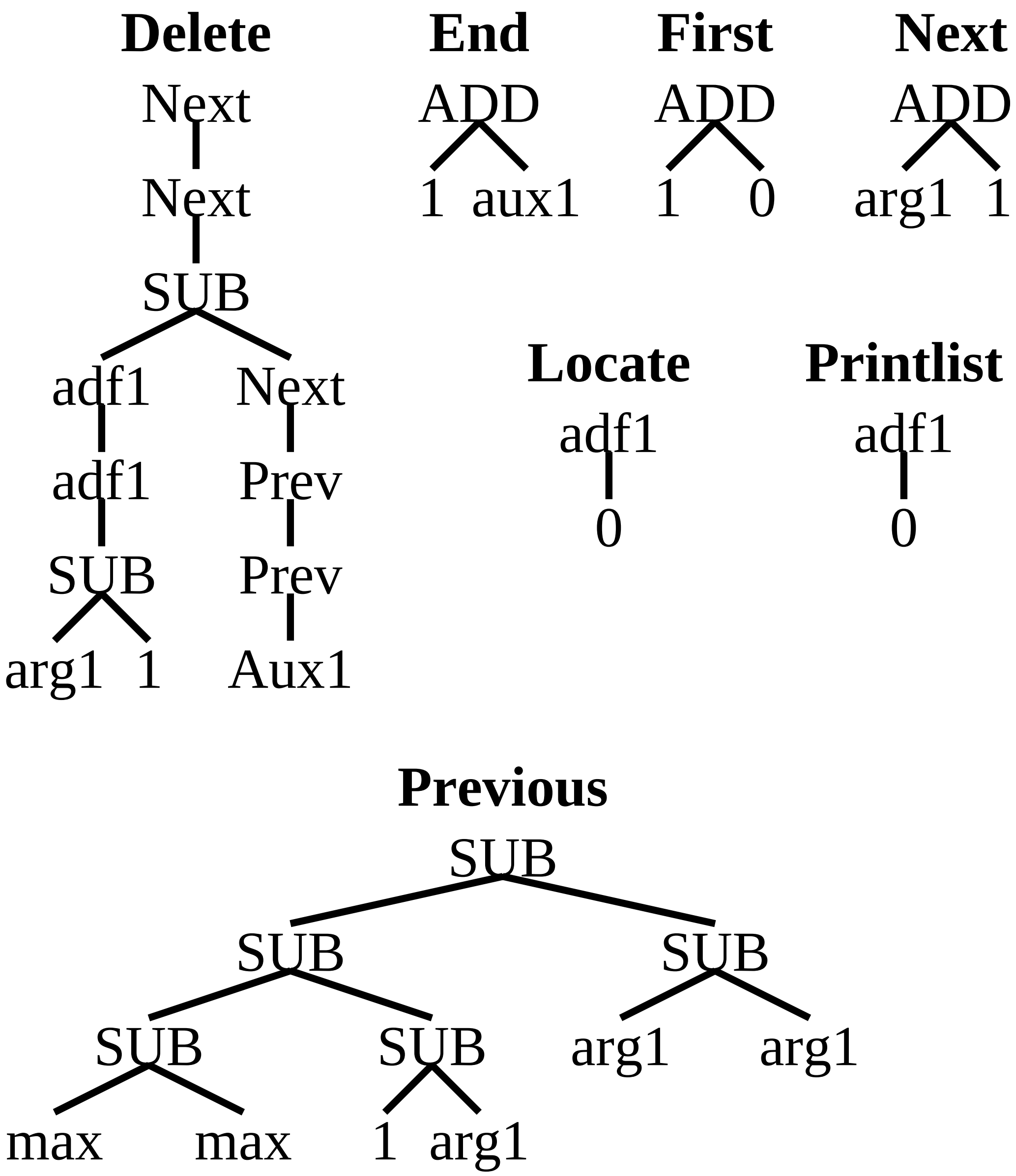
- Tournament selection (size 4)
- Select non-dominated
- 2+ compare with sample of pop, select least dominated (list only)
- A dominated by 1, B dominated by 3
- $\leadsto$ spread of fitness values
- $\leadsto$ more diverse population

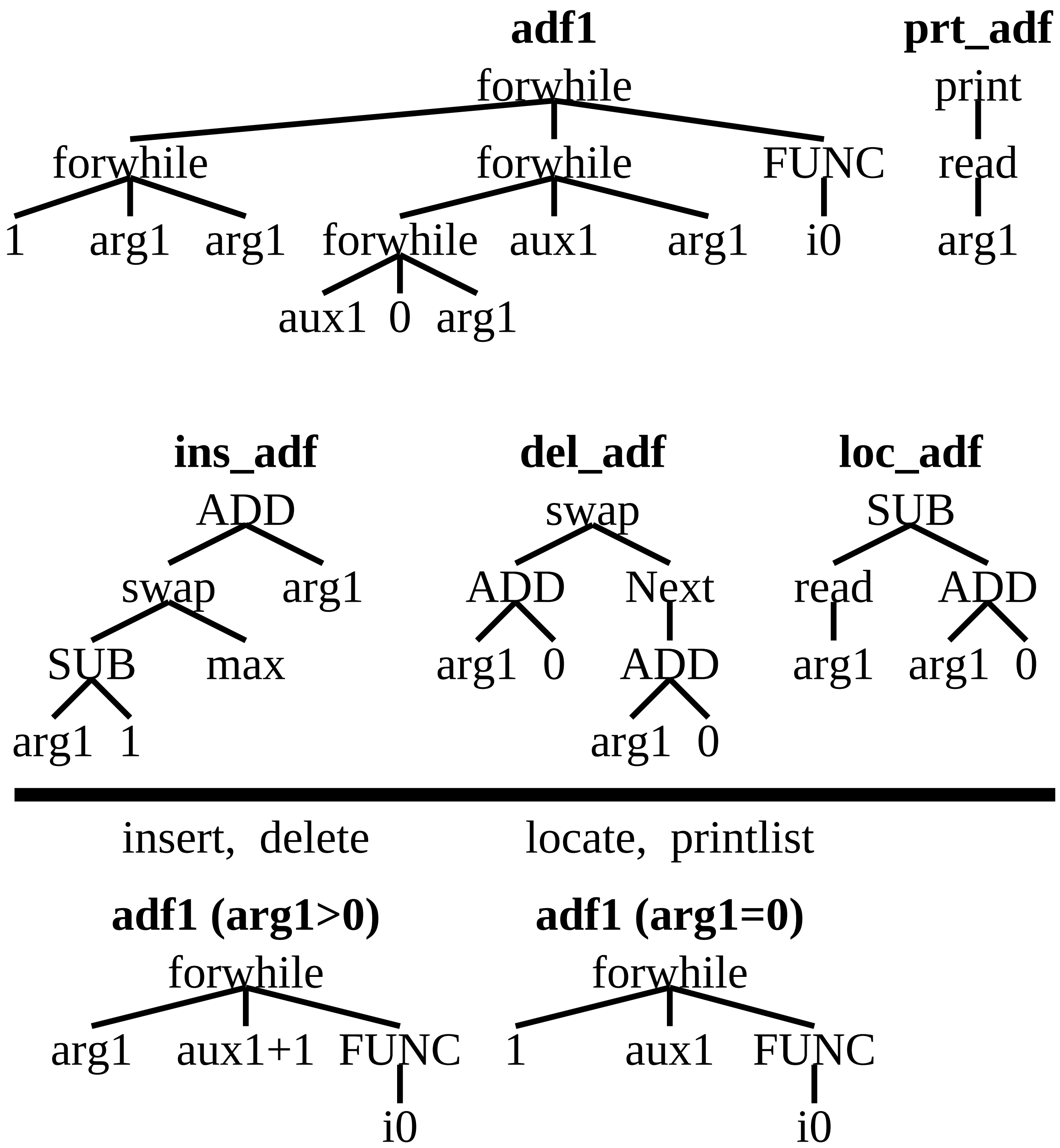
Results

- All 10 can be evolved together
- Incremental solution (cf. maintenance)
  Evolve 8 then evolve delete and locate






Delete
Next
Next
SUB
adf1
Next
adf1
Prev
SUB
Prev
arg1
1
Aux1
End
ADD
1
aux1
First
ADD
1
0
Next
ADD
arg1
1
Locate
adf1
0
Printlist
adf1
0
Previous
SUB
SUB
SUB
SUB
SUB
arg1
arg1
max
max
1
arg1


adf1
prt_adf
forwhile
print
forwhile
forwhile
FUNC
read
1
arg1
arg1
forwhile
aux1
arg1
i0
arg1
aux1
0
arg1
ins_adf
del_adf
loc_adf
ADD
swap
SUB
swap
arg1
ADD
Next
read
ADD
SUB
max
arg1
0
ADD
arg1
arg1
0
arg1
1
arg1
0
insert, delete
locate, printlist
adf1 (arg1>0)
adf1 (arg1=0)
forwhile
forwhile
arg1
aux1+1
FUNC
1
aux1
FUNC
i0
i0

Problems

- Fitness function cover all operations
- Avoid GP over fitting
  - random data
  - small and large values
  - initialize to same pattern
- Fitness function GP deceptive
- Premature convergence

  e.g. just remember last value added
- multiple partial solutions

  Can't cross different solution types?
- Long runtime fitness saving & caching

Future work

- Demonstrate GP using data structures
- General framework to evolve data structures as GA solves problem
- Full investigation of GP and demes and pareto selection

Conclusions

- Why evolve data structures? Power of abstraction. Easier to evolve answer

- Push down Stack, FIFO Queue, List

- Multi-tree architecture, function set, pareto fitness function

- Next steps, general framework

W. B. Langdon, UCL, London

email W.Langdon@cs.ucl.ac.uk
WWW http://www.cs.ucl.ac.uk/staff/W.Langdon/

Try it yourself!

ftp site `cs.ucl.ac.uk`		
directory	file	contents
`genetic/gp-code`	`GPdata-1-dec-94.tar.Z`	**C++ code**
`genetic`	`simple-gp.c`	generate C code
`genetic`	`park-miller.cc`	random numbers
`genetic/biblio`	`gp-bibliography.bib`	
`genetic/papers`		