5
Genetic programming

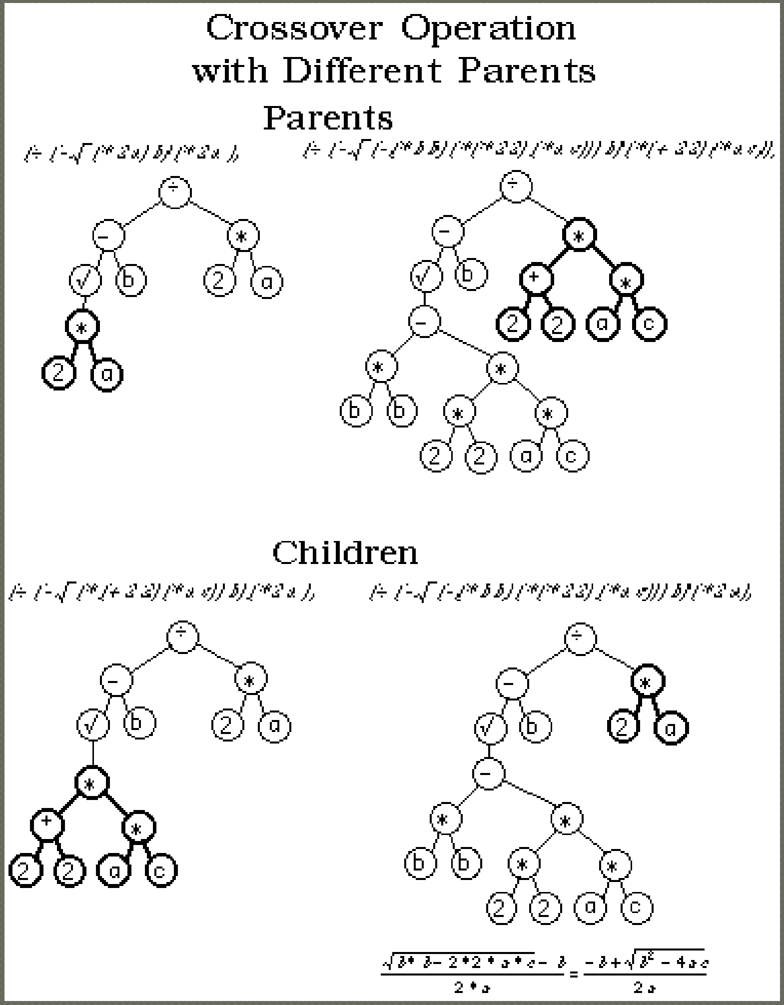
Cedric Gondro

The Centre for Genetic Analysis and Applications
University of New England

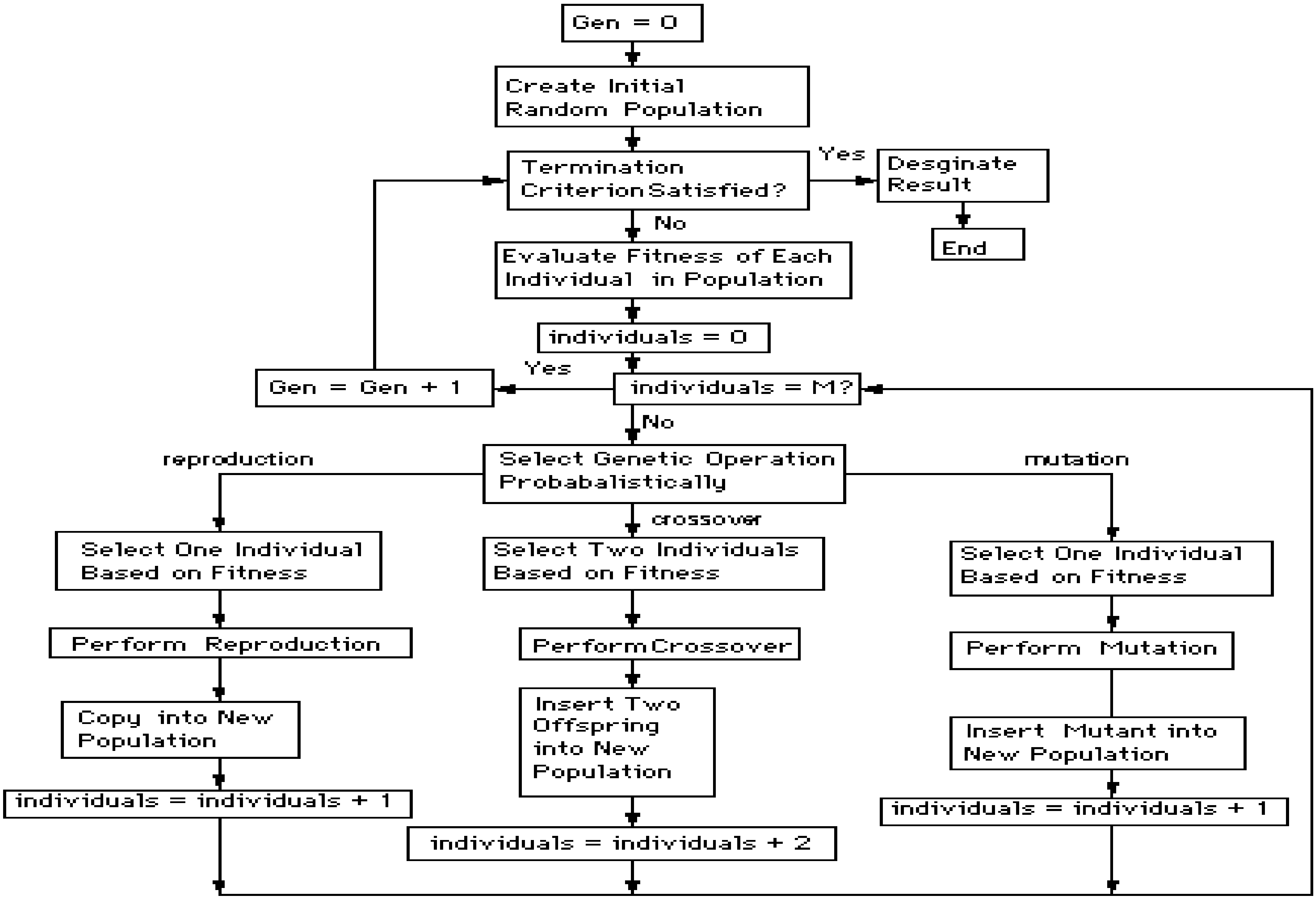
une
University of
New England

Genetic Programming

- Evolve structure as executable computer programs
- Originally parse tree structures
- LISP
- Many variants:
  - Tree GP (list)
  - Linear GP (stack)
  - Page based GP (--)
  - Gene Expression Programming (Array)
  - Multi Expression Programming (Array)
- Adds structure search to the parameter optimization



Flowchart for Genetic Programming



Genetic Operators

Operators that yield the best results in the different GPs:

- Mutation:
  - Single point
  - Deletion
  - Insertion
- Crossover:
  - 1 point crossover
  - 2 point crossover
  - Transposition
- Selection:
  - Tournament 2
- Steady State

Gene Expression Programming

- **GP methods tend to yield similar results, main differences are in convergence time, processing speed and ease of implementation.**
- Well suited for languages that do not support symbolic expressions.
- All organisms yield valid syntaxes.
- Since all solutions are valid there is a better coverage of the solution space.
- No or little repair overhead
- Simple GA search operators can be used

Gene Expression Programming

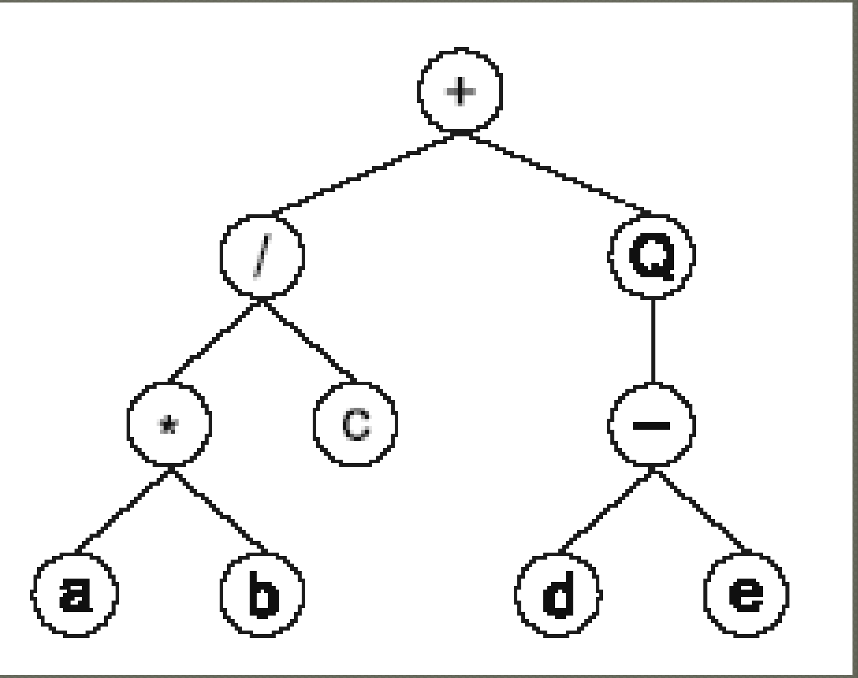
Organism = Head + Tail

Head -> terminals and functions
Tail -> terminals only

Tail= head*(n-1)+1

$$\frac{a \cdot b}{c} + \sqrt{d - e}$$

```
0123456789
+/Q*c-abde
```



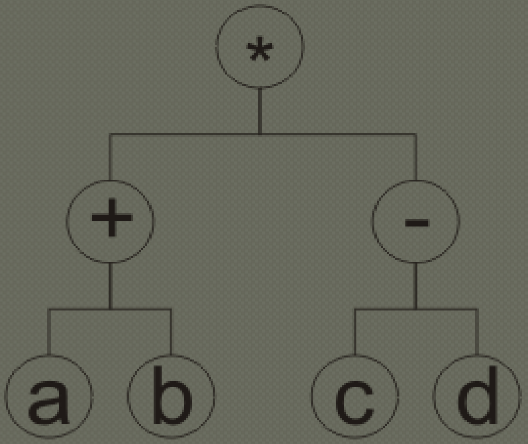
*Build recursively from the array
from the number of instructions in ORF.

An example

$$y=(a+b)*(c-d)$$

terminal set T={a,b,c,d)

function set F=(+,*,-}



0	1	2	3	4	5	6	7	8	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
*	+	-	a	b	c	d	*	-	-	**a**	**a**	**b**	**c**	**c**	**d**	**d**	**d**	**b**	**b**	**a**

GEP gene of size 21 with head (h) of size 10 and tail (t) of size 11
***t = h(n-1)+1*. The highest arity of the function set is 2, thus *n* = 2**
ORF = 7

Search operators - mutation

In the tail replace only with terminals

A single change can yield big differences!

- Single point mutation
- block mutation
  - Up to 20% of gene size
  - 10% of point mutation probability

Gene

A

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
*	+	-	a	b	c	d	*	-	-	a	a	b	c	c	d	d	d	b	b	a

B

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
*	+	-	a	*	c	d	*	-	-	a	a	b	c	c	d	d	d	b	b	a

ORF

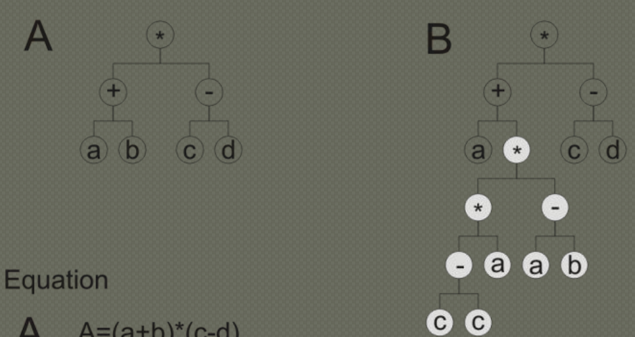
A

0	1	2	3	4	5	6
*	+	-	a	b	c	d

B

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
*	+	-	a	*	c	d	*	-	-	a	a	b	c	c

Expression Tree (ET)



Equation

A A=(a+b)*(c-d)

B B=(a+((c-c)*a)*(a-b))*(c-d)

Search operators - crossover

- One-point crossover
  - cuts both parents at the same position and the remainder of the gene downstream from the cut point is swapped to form the offspring
- Two-point crossover
  - selects a block of the same size, starting at the same position in both parents and this block is swapped in the offspring
- Three-point crossover
  - is a simple extension of two-point crossover where instead of one block being swapped, two blocks are swapped

Gene Expression Programming Notes

- Fixed sized arrays are faster than jagged arrays.
- Organism size is best defined as 5x the expected expression size.
- Population size of 1000/10000 seem to yield the best ratio convergence/speed.
- Use 1 deme for each expression – improves performance and convergence, instead of different chromosomes for each expression.

Handling breakdowns and constraints

- Division by zero
- Negative square roots
- Time delays

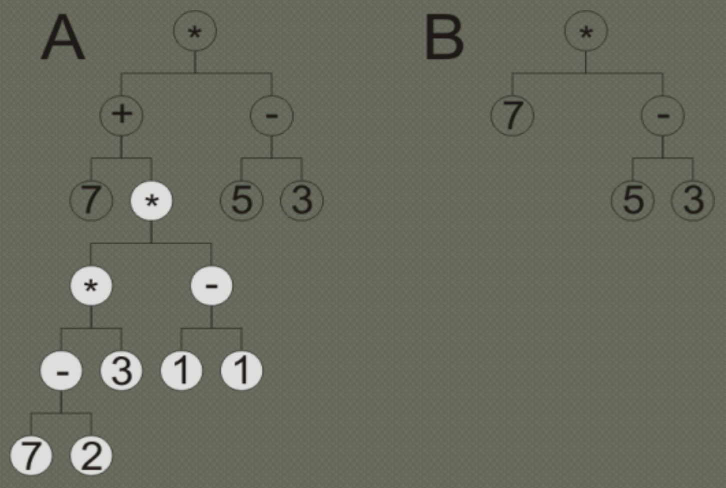
- Prune
- Penalize
- Repair

- Constraints
  - cheap and easy solution: average value of the parameter range

Bloat

- Mainly due to convergence/stagnation
- Growth without changes to the fitness value

- Limit maximum size and tree depth
- Weighted fitness function
- Code editing and tree pruning



Parameter x Model Discovery

y=a*pow(x,3)+b*pow(x,2)+c*x

- DE, GA, ES, EP – ideal for finding parameters of a model
  - a, b, c and x
- Genetic Programming – ideal for structure discovery (less ideal for parameters)
  - Parameters: a, b, c and x
  - Terminals: *, pow, +
  - Function: y=a*pow(x,3)+b*pow(x,2)+c*x

University of New England