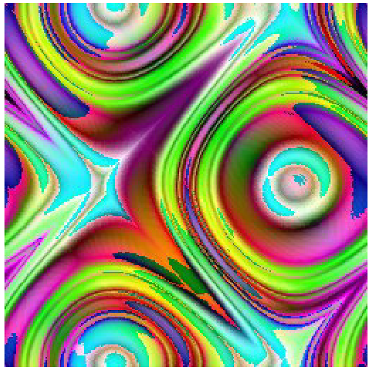
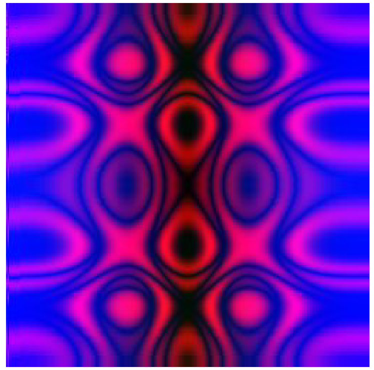


cgp evolved picture

Cartesian Genetic Programming in a nutshell



cgp evolved picture

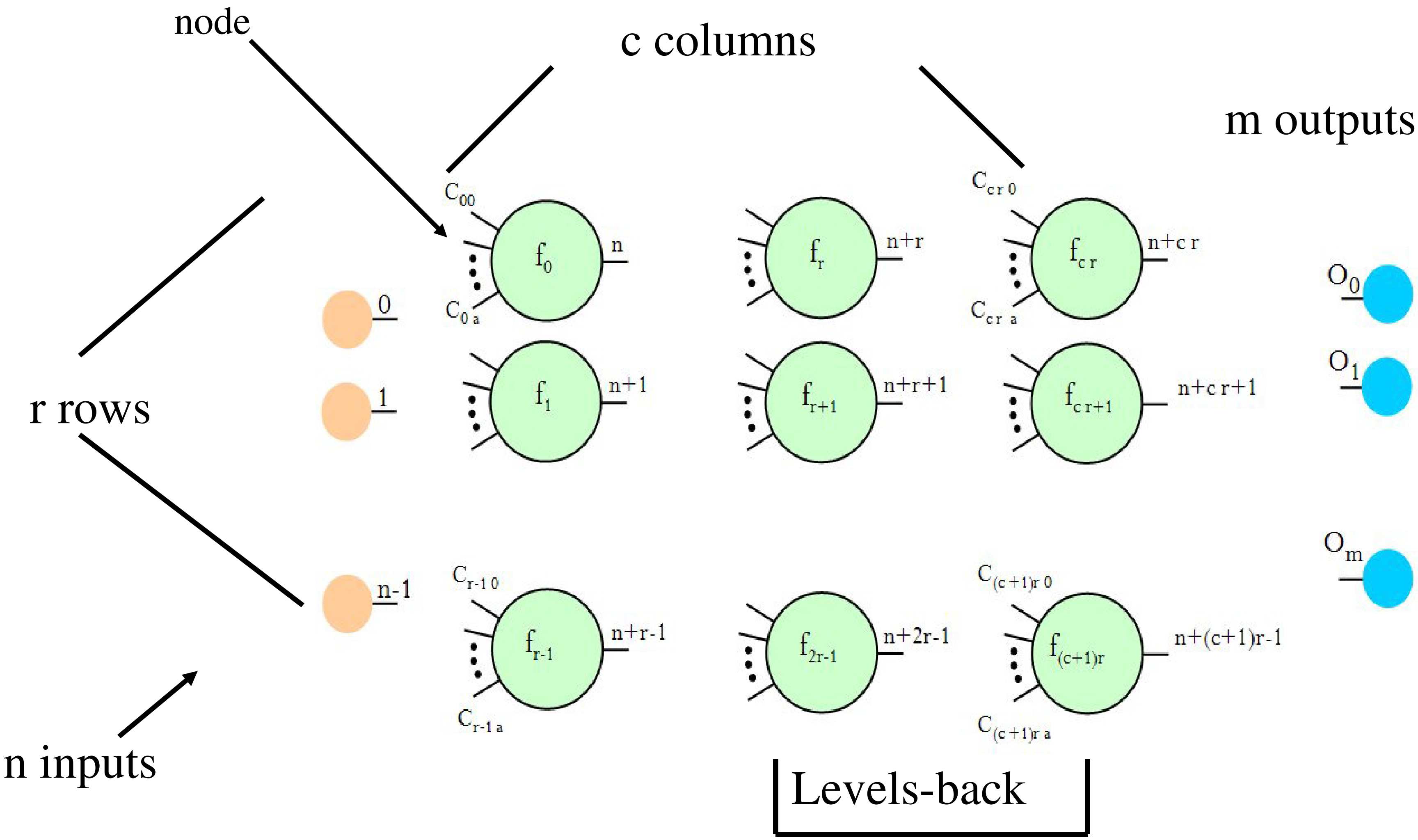
Julian F. Miller
Honorary Fellow
Dept of Electronic Enginering
http://www.elec.york.ac.uk
University of York, UK

Home site
http://www.cartesiangp.com

What is CGP?

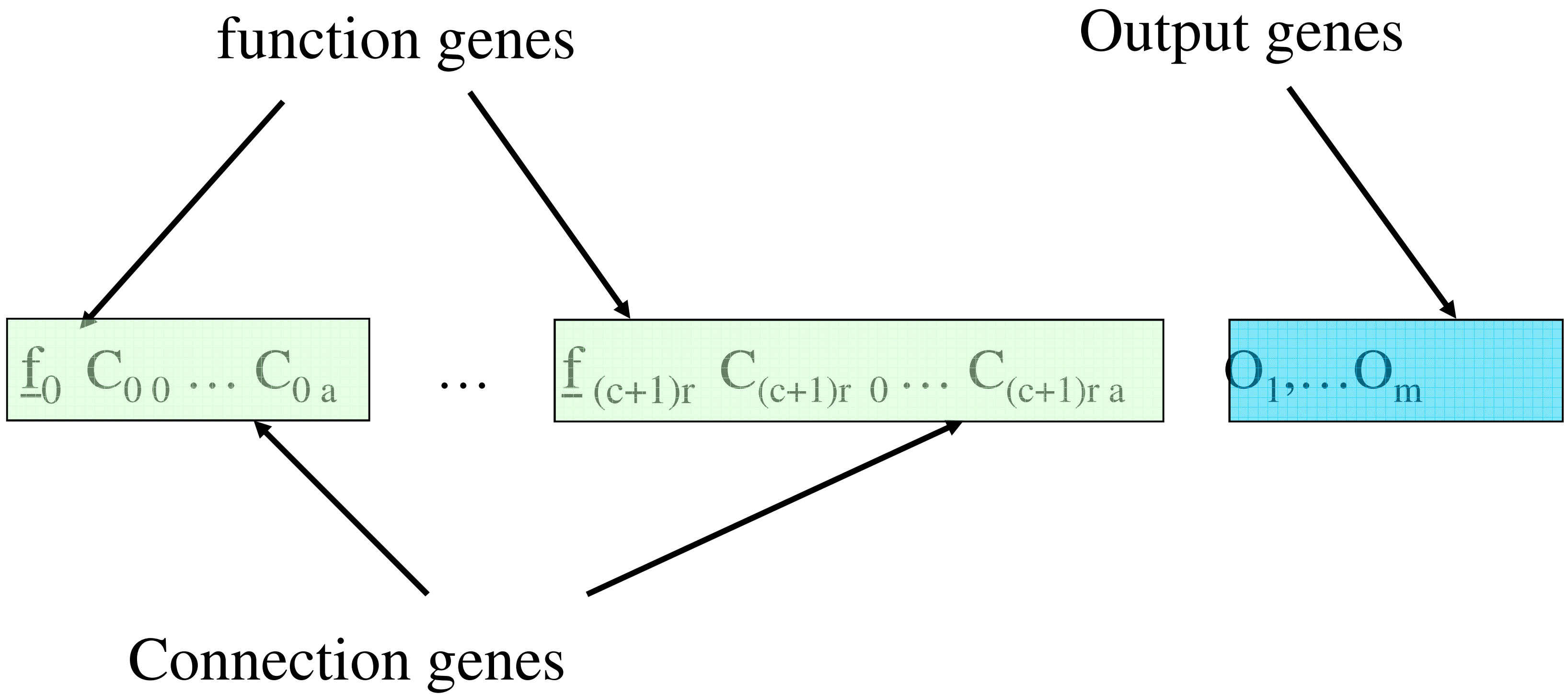
- CGP is a form of automatic computer program evolution (which itself is generally known as genetic programming).
- CGP was developed from work on the evolution of digital circuits, by Miller and Thomson 1997. First actual mention of the term *Cartesian Genetic Programming* appeared at the GECCO conference in 1999.
- The genotype is a list of integers (and possibly parameters) that represent the program primitives and how they are connected together
  - CGP represents programs as *graphs* in which there are explicit *non-coding genes*
  - CGP allows program to be evolved with more than one output
- The genes are
  - Addresses in data (connection genes)
  - Addresses in a look up table of functions (function genes)
  - Additional parameters (possibly)
- CGP easily encodes computer programs, electronic circuits, neural networks, mathematical equations and other computational structures.
- It allows a form of Darwinian evolution to evolve solutions to problems automatically and efficiently. In a sense it is an invention machine and can find unusual and efficient solutions to many problems in many fields of science.

CGP General form



Note: Nodes in the same column are not allowed to be connected to each other

CGP genotype

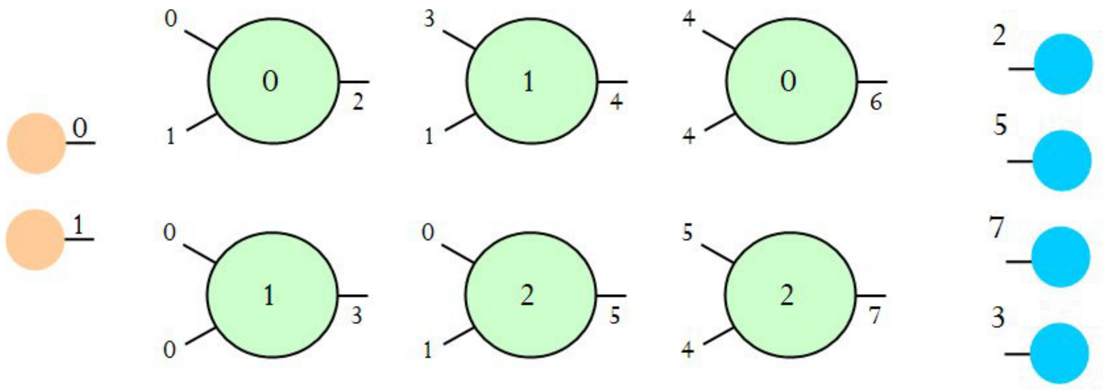


Usually, all functions have as many inputs as the *maximum* function arity

Unused connections are ignored

CGP has three parameters: number of columns, number of rows and levels-back. These control the layout and connectivity of nodes

Example



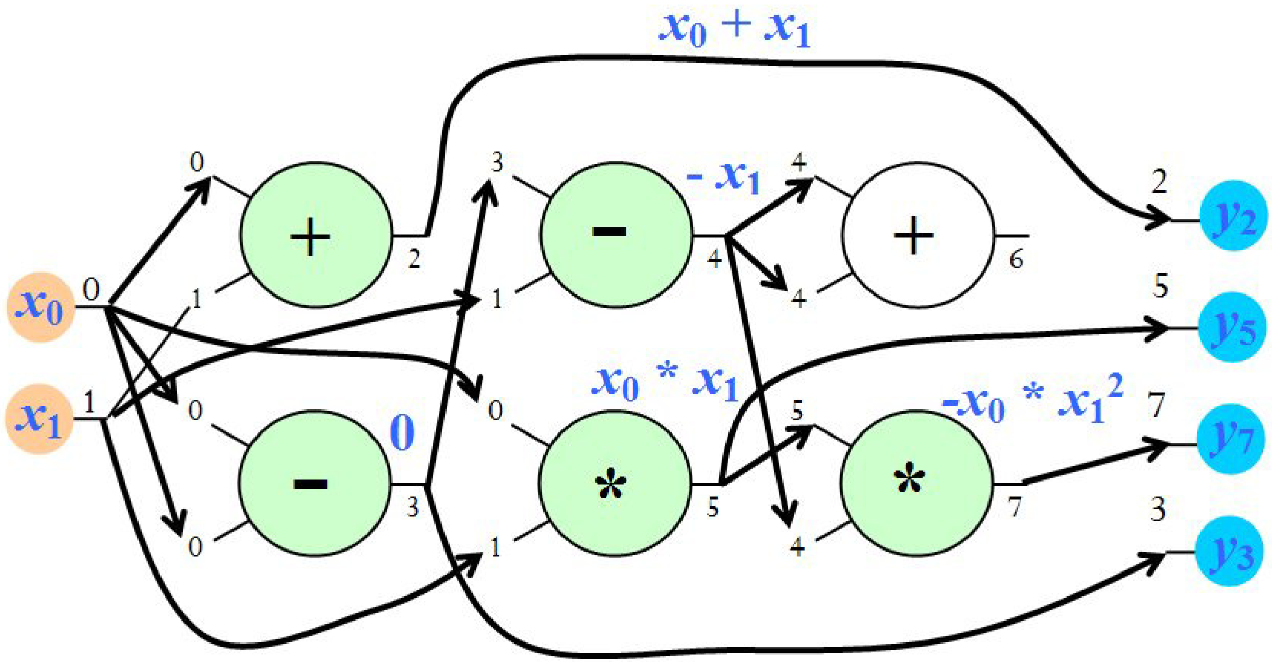
Function look-up table

Function gene (address)	Action
<u>0</u>	Add
<u>1</u>	Subtract
<u>2</u>	Multiply
<u>3</u>	Divide (protected)

Genotype

<u>0</u> 0 1 <u>1</u> 0 0 <u>1</u> 3 1 <u>2</u> 0 1 <u>0</u> 4 4 <u>2</u> 5 4 2 5 7 3

So what does the genotype represent?



$$y_2 = x_0 + x_1$$
$$y_5 = x_0 * x_1$$
$$y_7 = -x_0 * x_1^2$$
$$y_3 = 0$$

The CGP genotype-phenotype map

- When you decode a CGP genotype many nodes and their genes can be ignored because they are not referenced in the path from inputs to outputs
- These genes can be altered and make no difference to the *phenotype*, they are non-coding
- Clearly there is a many-to-one genotype to phenotype map

Decoding CGP chromosomes requires 4 simple steps

```
// L  = MaxGraph.Length
// I   = Number of program inputs
// N = Number of program outputs
bool          ToEvaluate[L]
double        NodeOutput[L+I]
```

1

```
// identify initial nodes that need to be evaluated
p = 0
do
  ToEvaluate[OutputGene[p]] = true
  p = p + 1
while (p < N)

// work out which nodes are used
p = L-1
do
  if (ToEvaluate[p])
     x = Node[p].Connection1
     y = Node[p].Connection2
    ToEvaluate[x] = true
    ToEvaluate[y] = true
  endif
  p = p - 1
while ( p >= 0)
```

2

```
// load input data values
p = 0
do
  NodeOutput[p] = InputData[p]
  p = p + 1
while (p < I)
```

3

```
//Execute graph
p = 0
do
  if (ToEvaluate[p])
     x = Node[p].Connection1
     y = Node[p].Connection2
     z = NodeFunction[p].Function
     NodeOutput[p+I] = ComputeNode(NodeOutput[x], NodeOutput[y],z)
  endif
  p = p + 1
while (p < L)
```

4

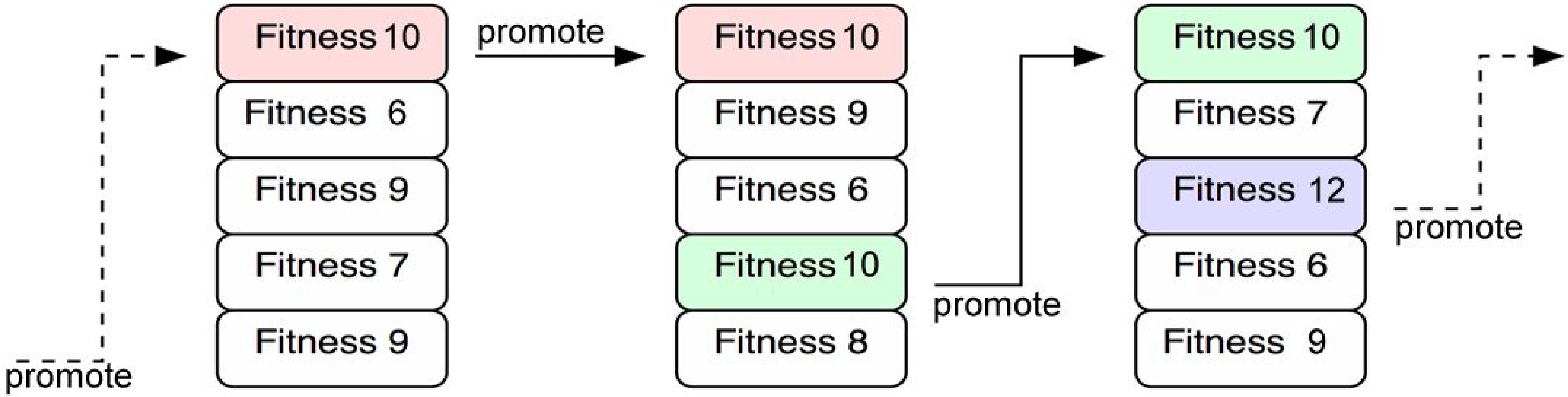
Point mutation

- Most CGP implementations only use mutation.
- Carrying out mutation is very simple. It consists of the following steps.

  The genes must be chosen to be valid alleles

```
//Decide how many genes to change:num_mutations
while (mutation_counter < num_mutations)
{
        get gene to change
        if (gene is a function gene)
          change gene to randomly chosen new valid function
        else if (gene is a connection gene)
          change gene to a randomly chosen new valid connection
        else
          change gene to a new valid output connection
}
```

Genotypes are evolved with an Evolutionary Strategy



- CGP often uses a variant of a simple algorithm called (1 + 4) Evolutionary Strategy
  - However, an offspring is always chosen if it *is equally as fit* or has better fitness than the parent (most important)