Evolutionary Design Systems (cont.)

Rob Saunders
FITNESSES

PHENOTYPES

GENOTYPES

EXPRESSION

EVALUATION

SELECTION

FITNESSES
Genotypes
TREE GENOTYPES

CROSSOVER
GRAPH GENOTYPES
GENETIC PROGRAMMING

```plaintext
( CODE.QUOTE ( INTEGER.POP 1 )
  CODE.QUOTE ( CODE.DUP INTEGER.DUP 1 INTEGER.- CODE.DO INTEGER.* )
  INTEGER.DUP 2 INTEGER.< CODE.IF )
```
Reproduction
Phenotypes
Phenotypes
Phenotypes
Phenotypes

<table>
<thead>
<tr>
<th>Genotype</th>
<th>Phenotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2082</td>
<td></td>
</tr>
<tr>
<td>0.8937</td>
<td></td>
</tr>
<tr>
<td>0.0094</td>
<td></td>
</tr>
<tr>
<td>0.1403</td>
<td></td>
</tr>
<tr>
<td>0.7170</td>
<td></td>
</tr>
<tr>
<td>0.3533</td>
<td></td>
</tr>
<tr>
<td>0.5513</td>
<td></td>
</tr>
<tr>
<td>0.8141</td>
<td></td>
</tr>
<tr>
<td>0.9915</td>
<td></td>
</tr>
<tr>
<td>0.0800</td>
<td></td>
</tr>
<tr>
<td>0.8884</td>
<td></td>
</tr>
<tr>
<td>0.1334</td>
<td></td>
</tr>
<tr>
<td>0.2935</td>
<td></td>
</tr>
<tr>
<td>0.8940</td>
<td></td>
</tr>
</tbody>
</table>
Rule-Based Expression

Rules

00 $\rightarrow$ $\begin{array}{c} \square \\square \end{array}$

10 $\rightarrow$ $\begin{array}{c} \square \quad \square \quad \square \quad \square \end{array}$

01 $\rightarrow$ $\begin{array}{c} \square \quad \square \end{array}$

11 $\rightarrow$ $\begin{array}{c} \square \quad \square \quad \square \quad \square \end{array}$
Rule-Based Expression

Genotype
01 10 10 11 11 00
Rule-Based Expression

Genotype
01 10 10 11 11 00

Expression
Rule-Based Expression

Genotype

01 10 10 11 11 00

Expression
Rule-Based Expression

Genotype

01 10 10 11 11 00

Expression
Rule-Based Expression

Genotype

01 10 10 11 11 00

Expression
Rule-Based Expression

**Genotype**

```
01 10 10 11 11 00
```

**Expression**

![Expression Diagram]
Rule-Based Expression

01 10 10 11 11 00

Expression
Rule-Based Expression

Genotype

01 10 10 11 11 00

Expression
Rule-Based Expression

Genotype
01 10 10 11 11 00

Expression
Rule-Based Expression

Genotype

01 10 10 11 11 00

Expression

Phenotype
Fitnesses
Fitness Functions

- Quantify the optimality of a solution
  - Allowing genotypes to be compared
- Should correlate with design goal(s)
  - Can be difficult to do in practice
Fitness Functions
Fitness Landscapes

Hill-Climbing
Global vs Local Optima
Rugged Landscapes
Smooth Landscapes

\[ 100 - \pi \cdot \text{radius}(x,y) + \text{radius}(x,y) \cdot \cos(R(x,y)) \]
Representation Changes
Multi-Objective Fitness Functions
Multi-objective Fitness Functions

- Designs often have multiple objectives
- Standard GAs often combine individual fitness functions using a weighted sum
  - $F = xA + yB + zC$
- This leads to problems because unfit designs can seem fit because they score highly in just one of the objectives
Multi-objective Fitness Functions

- Example of a problem with weighted sum
  - $F = 5 + 0 + 0$
  - $F = 1 + 1 + 1$

- Linear sum would favour first design over second even though it fails in 2/3 of the design objectives
  - Some GAs limit the valid range of the individual fitnesses to avoid this problem
Multi-objective Fitness Functions

- Another way to combine fitness functions is to multiply them together
  - \( F = 5 \times 0 \times 0 \)
  - \( F = 1 \times 1 \times 1 \)

- Multiplying fitnesses together implies some sort of cooperative (or inter-dependent) relationship between the objectives
Pareto Fronts

A [design] is a member of the Pareto front if no other single [design] has been found which is better in all objectives.

Figure: http://research.et.byu.edu/growth/ParetoSet.htm

Dominated Solutions
Multi-objective Strategies

- Switching objectives - optimise objectives individually to explore trade-offs
- Parameter variation - change weighting of objectives to explore trade-offs
- Pareto selection - select individuals based on dominance to eliminate inferior designs

Reference: http://www.calresco.org/lucas/pmo.htm
Multi-objective Strategies

- Fitness sharing - lower fitness of closely related designs to encourage diversity
- Restricted mating - allow only closely related design to mate to keep good building blocks
- Reinitialisation - introduce random designs to introduce novelty