GA Issues

Peter G. Anderson,

Computer Science Department

Rochester Institute of Technology,

Rochester, New York

anderson@cs.rit.edu http://www.cs.rit.edu/

February 2004
Abstract
A discussion of the parameters for GAs and how to go about systematically, intelligently determining their values.
Parent Selection

Selection is a primary tool of a GA. (The other main tool is crossover.)

Here is a common technique: let $F = \sum_{j=1}^{\text{popsize}} \text{fitness}_j$

Select individual $k$ to be a parent with probability $\frac{\text{fitness}_k}{F}$

There are some problems here:
- fitnesses shouldn’t be negative
- probabilities should be “right;” avoid skewing by super heros.
Parent Selection: Rank

Here is another technique.
Order the individuals by fitness rank
Worst individual has rank 1. Best individual has rank POP_SIZE

Let $F = 1 + 2 + 3 + \cdots + POP_SIZE$
Select individual $k$ to be a parent with probability $\frac{\text{rank}_k}{F}$

Benefits of rank selection:
• the probabilities are all positive
• the probability distribution is “even”
• tournament selection implements this
Parent Selection: Rank Power

Yet another technique.

Order the individuals by fitness rank. Worst individual has rank 1. Best individual has rank POP_SIZE.

Let \( F = 1^s + 2^s + 3^s + \cdots + POP\_SIZE^s \)
Select individual \( k \) to be a parent with probability \( \frac{rank_k^s}{F} \)

benefits:
- the probabilities are all positive
- the probabilities can be skewed to use more “elitist” selection
- \((s + 1)\)-element tournament selection implements this
Crossover Methods

Crossover is a primary tool of a GA. (The other main tool is selection.)

Common techniques for bit string representations:

**One-point crossover**: Parents exchange a random prefix

**Two-point crossover**: Parents exchange a random substring

**Uniform crossover**: Each child bit comes arbitrarily from either parent

(We need more clever methods for permutations & trees.)
Select three individuals, A, B, and C.

Suppose A has the highest fitness and C the lowest.

Create a child like this.

```c
for(i = 0; i < length; i++) {
    else child[i] = 1 - C[i];
}
```

We just suppose C is a “bad example.”
Crossover Methods & Schemas

Crossovers try to combine good schemas in the good parents. The schemas are the good genes, building blocks to gather.

The simplest schemas are substrings.

1-point & 2-point crossovers preserve short substring schemas.

Uniform crossover is uniformly hostile to all kinds of schemas.
Crossover for Permutations (A Tricky Issue)

Small-alphabet techniques fail. Some common methods are:

- OX: ordered crossover
- PMX: partially matched crossover
- CX: cycle crossover

We will address these and others later.
Crossover for Trees

These trees often represent computer programs.

Think Lisp

Interchange randomly chosen subtrees of parents.
The Breeding Pool

steady-state algorithm simply:

• selected two parents (via a tournament)
• selected two losers (same tournament)
• created two children
• entered the children in the population

A common alternative is to use a breeding pool.
The Breeding Pool Algorithm

Start with an empty breeding pool with capacity $POP\_SIZE$.

Until the breeding pool is full, do
   Copy individuals from current pop into the breeding pool
   (use some decent selection mechanism)

Initialize the new population to empty.

Until the breeding pool is empty, do
   Remove two parents from the pool & create two children
   Add the children to the new population
Mutation: Preserve Genetic Diversity

Mutation is a minor GA tool.

Without mutation we may get premature convergence to a population of identical clones.

A Genetic Algorithm must combine

- Exploration (e.g., random search)
- Exploitation (e.g., hill climbing)
Mutate Strings & Permutations

• Bit strings (or small alphabets)
  – Flip some bits
  – Reverse a substring (nature does this)

• Permutations
  – Transpose some pairs
  – Reverse a substring

• Trees . . .
Uses of Randomness in the GA

1. Initializing the population members at start-up.

2. Selecting individuals to compete for parenthood.

3. Selecting crossover points.

4. Mutation.

What difference can the choice of random number make?
Different Random Number Generators

1. `drand48()` in the standard C library.

2. From an old calculator handbook: \( x_n \leftarrow \{(x_{n-1} + \pi)^5\} \).

3. Some uniform, subrandom numbers: \( x_n \leftarrow \{x_{n-1} + (1 + \sqrt{5})/2\} \).

4. Linear congruential ("Fibonacci"): \( x_n \leftarrow \{x_{n-1} + x_{n-2}\} \).

\( \{x\} \) denotes the fractional part of \( x \). \( \{x\} = x - [x] \).
1000 values of \texttt{drand48()}

![Graph showing 1000 values of \texttt{drand48()}.]
1000 values of $x_n \leftarrow \{(x_{n-1} + \pi)^5\}$
1000 values of \( x_n \leftarrow \{x_{n-1} + (1 + \sqrt{5})/2\} \)
1000 values of $x_n \leftarrow \{x_{n-1} + x_{n-2}\}$
Comparing Three Randoms: How Long Does It Take?

 Doesn’t it matter???
Random Experiments Used The CA Synch Problem

POP_SIZE  50
tournament_size  2
two_point  1
FIT_EVALS  10000
N  10
CA_length  300
no_left_nhb  3
no_right_nhb  3
pbm_lines  600
seed  0
random_type  0
MUT_RATE  0.001000