

## CS 555/495: Evolutionary computation, selection methods

Lecture notes for October 9, 2002

Alden H. Wright

Department of Computer Science, University of Montana, Missoula, MT

A very good reference (written by Adam Prügel Bennet) on the web:

[http://www.isis.ecs.soton.ac.uk/isystems/evolutionary/notes/evol/Genetic\\_Algorithms.html](http://www.isis.ecs.soton.ac.uk/isystems/evolutionary/notes/evol/Genetic_Algorithms.html)

### 1 Evolutionary computation as compared to local search

The local search methods (hill-climbing, simulated annealing, tabu search, and local search) all use a single “current individual”. Evolutionary computation methods use a population of individuals. This means that the local search methods can be considered as special cases of evolutionary computation. (My use of the term “local search methods” here is not a standard terminology.)

One of the possible advantages of evolutionary computation over local search methods is that it enables the use of crossover. Why could this be an advantage? We might look at one of the hypothesized advantages of sexual reproduction in biology. This argument is due to H. J. Muller (1932, Amer. Natur. 8:118-138). Suppose that there are two mutations that would be advantageous for a population. Without sexual reproduction, one of these mutations would have to arise first, and then the second mutation would have to arise in a descendant of the individual that had the first instance of the first mutation. With sex (and crossover) the two mutations could arise in different individuals, the descendants of these two individuals could mate, and children of this mating can have both mutations.

### 2 Generational and steady-state algorithms

There are two basic templates for an evolutionary computation algorithm, generational and steady-state. In a generational algorithm, all or most individuals are replaced in each major step. In a steady-state algorithm, only one or a small number of individuals is replaced in every step.

Here is the template for a generational algorithm.

```

1  Initialize a random population of individuals (the current population)
2  repeat
3    for  $i \leftarrow 1$  to  $popSize$  do
4      if  $random() < crossoverRate$  then
5        Use selection based on fitness to choose 2 individuals from the current population
6        Use crossover to combine these individuals to create a child
7      else
8        Use selection to choose 1 individual (the child) based on fitness from the current population
9      end if
10     Apply mutation to the child
11     Add the child to the new population
12   end for
13   Replace the current population by the new population
14 until termination test

```

In this algorithm,  $random()$  returns a random number between 0 and 1. The underlined words stand for procedures that are supplied by the designer of the evolutionary computation algorithm.

Here is a template for the steady-state algorithm.

```

1  Initialize a random population of individuals
2  repeat
3      if  $random() < crossoverRate$  then
4          Use selection based on fitness to choose 2 individuals
5          Use crossover to combine these individuals to create a child
6      else
7          Use selection to choose 1 individual (the child) based on fitness
8      end if
9      Apply mutation to the child
10     Use deletion based on fitness to remove an individual from the population
11 until termination test

```

Note that there are two steps that are based on fitness rather than one. A common implementation for the deletion procedure is to delete the worst element of the population. If both selection and deletion are really based on fitness, then the algorithm has stronger selection than the generational algorithm. In other words, the less-fit individuals are eliminated from the population more quickly.

### 3 Selection methods

Note that selection is representation-independent.

There are a number of methods for implementing selection.

#### 3.1 Proportional selection

This method is also called *roulette wheel selection*.

The idea of the roulette wheel analogy is that we divide a roulette wheel up into sectors, with one sector for each individual in the current population. The size of the sector is proportional to the corresponding individual's fitness. (See Figure 6.1 of the text.) Now when the roulette wheel is spun, an individual is chosen with a probability proportional to its fitness.

Thus, the probability of selecting individual  $x$  is given by

$$p(x) = \frac{f(x)}{\sum_y f(y)}$$

Here  $f(x)$  is the fitness of individual  $x$ , and the sum in the denominator is done over all individuals in the population.

Let us do an example. Suppose that the population is of size 4, and the four individuals have fitnesses of  $\{7, 1, 3, 5\}$ . Then we can construct the following table:

Individual	fitness	scaled fitness
1	7	7/15
2	1	1/15
3	3	3/15
4	5	5/15
Total	15	1

Since the scaled fitnesses add up to 1 by definition, they can be used as probabilities. Thus, the probability of selecting individual 1 is 7/15.

Thought problem: Suppose that your population is very large. How do you implement this method efficiently?

In many situations, proportional selection is too weak for optimization. Consider what can happen as we get close to the optimum. The fitness of the optimum individual might be 100, and the fitness of nearby individuals might be 99. Then the optimum individual has only about a 1% selective advantage over nearby individuals. If there are many such nearby individuals, then it may be unlikely to see the optimum individual.

This problem can be reduced by using *scaled selection* where the fitness of the worst individual in the population is subtracted from the fitness of all individuals. However, tournament and ranking selection are more commonly used. These will be described later.

Proportional selection is also difficult to parallelize in the model where different individuals or groups of individuals are on different processors.

### 3.2 Boltzmann Selection

Boltzmann selection is essentially an exponentiation rescaling of proportional selection. One does proportional selection on  $e^{\beta f(x)}$  where  $\beta$  is a parameter which might be called the inverse temperature. The probability of selecting individual  $x$  is given by

$$p(x) = \frac{e^{\beta f(x)}}{\sum_y e^{\beta f(y)}}$$

The strength of selection is controlled by the  $\beta$  parameter. A high value of  $\beta$  (low temperature) gives a strong selection (similar to hill-climbing), and a low value of  $\beta$  gives a weak selection.

A big advantage of Boltzmann selection is this parametric control over the strength of selection. It also has some nice theoretical properties in some models of genetic algorithms.

### 3.3 Ranking selection

Under ranking selection, selection is done on the reverse ranks of the population under fitness. The population is ranked according to fitness with the highest fitness individual having the largest rank. Let us redo our example of population of size 4.

Individual	fitness	rank	scaled rank
1	7	4	4/10
2	1	1	1/10
3	3	2	2/10
4	5	3	3/10
Total	10	1	

The advantage of ranking selection is that it gives more selective

pressure towards the optimum when the fitnesses of the population are similar. This often may happen at

the end of a run.

### 3.4 Tournament selection

First we describe binary tournament selection. Two individuals are selected randomly from the population, and the better of these is the result of the selection.

In  $k$ -ary tournament selection,  $k$  individuals are selected randomly from the population, and the best of these is selected.

Tournament selection is a rank-based method. The probability that an individual will be selected is based only on the rank of that individual in the population ordered by fitness, and not on the size of the fitness. Thus, it provides for more selective pressure than proportional selection when the fitnesses of the population are similar.

An advantage of tournament selection is that it is very easy to implement, and it works very well in a parallel implementation where different individuals are on different processors.

### 3.5 Truncation selection

In  $k$ -ary truncation selection, the best  $k$  individuals are selected. One might select the best half of the population, or the best one-fifth.

This is another rank-based selection method.

Truncation selection is a very strong selection method. (See below.)

### 3.6 Deletion methods

In a steady-state algorithm, a method is needed to delete individuals from the population. This can be thought of as selection in reverse—you want to have a higher probability of selecting individuals of low fitness.

Thus any of the selection methods can be used in reverse. One could use the negative fitness plus a constant with proportional selection, or one could use the ranks rather than the reverse ranks in ranking selection. One could also use random deletion where a random element of the population is deleted.

A common method is to delete the worst element of the population. This is a form of truncation selection.

### 3.7 Selection strength

The selection strength measures how strongly the fitter individuals are selected over the less fit individuals. One measure would be

$$\textit{selection\_strength} = \frac{\textit{increase\_in\_average\_fitness}}{\textit{standard\_deviation\_of\_population}}$$

If selection is applied repeatedly to a finite population with no other operators, the result will eventually be a uniform population (a population where all of the individuals are the same). Another measure of selection strength would be the average length of time that it would take to reduce a random population of size  $N$  to a uniform population.

Selection strength is an important parameter of an evolutionary computation algorithm.

## 4 Representation

The choice of representation is one of the most crucial in the designing of an evolutionary algorithm.

A biological way of looking at representation is the genotype-phenotype map. The genotype of an organism is its genome, or DNA (or possibly RNA) complement. The phenotype of the organism is the organism as it interacts with its environment. The process of using the genotype to produce the phenotype is very complex. It involves transcription of the DNA by messenger RNA, and the construction of the corresponding protein by the ribosome. In a complex organism, the proteins are produced in just the right amounts at just the right times. This complex process is called the genotype-phenotype mapping.

In evolutionary computation, the genotype is a data structure that can be mapped into the phenotype, which is a potential solution to the problem. The variation producing operators of mutation and crossover operate on the genotype. However, fitness is determined from the phenotype. In many cases, we can say that it is the interaction of the phenotype with its environment that determines the fitness of the phenotype.

In some cases, the genotype-phenotype mapping is very simple and straightforward. For example, in the binary string representation for the SAT problem, the mapping is straightforward. In other examples like the prisoner's dilemma, the mapping is more complex.

### 4.1 Fixed-length string representations

In this case the representation data structure is a fixed length string. The most common alphabet is the binary alphabet, but this is not necessary. In fact, it would be possible to have a different alphabet at each string position.

Note that a string is equivalent to an array where the values of the array are taken from an alphabet (normally a finite alphabet).

Biological terminology is often used for this representation. A **locus** is a string position, and an **allele** is the symbol in that string position.

Usually, the symbol at each string position has a fixed meaning. For example, in the SAT problem representation, string position  $i$  corresponds to the value of the variable  $x_i$ .

It is not always the case that each string position has a fixed meaning. It is also possible to have a representation as a sequence of genes, where each gene has a meaning independent of its position on the string.

For example, in biology, a gene which codes for a protein has the same meaning wherever it occurs on the chromosome.

In evolutionary computation, a gene might be an ordered pair, where the first element of the pair determines the function of the gene, and the second element determined the value or allele of the gene.

For example, one might construct a SAT representation using ordered pairs for genes. The first element of the pair would be the variable number, and the second would be the value of that variable. For example, the gene (5, 1) would specify that the SAT variable  $x_5$  should have value 1.

### 4.2 Vector of real numbers representation

For real-parameter problems, the most common representation is a fixed-length vector of real numbers. This is also the representation used by classical real-parameter optimization.

The textbook uses a non-linear programming (NLP) problem as an example of this representation.

### 4.3 Permutation representations

For many scheduling problems, a permutation is the natural representation. However, there are several ways to represent permutations. These include the adjacency representation, the ordinal representation, the path representation, and the network random keys representation. (See section 8.1 of the text.) We will return to the details of these when we discuss crossover methods for permutations.

### 4.4 Finite state machines

A finite state machine (in particular, a Mealy machine) is a way to map a string of input symbols into a string of output symbols. At each time step, the machine is in one of a finite number of states. When an input symbol comes into the machine, the machine goes into a new state, and it outputs a symbol from its output alphabet. Thus, the machine is described by

1. A finite set  $P$  of internal states,
2. A finite set of input symbols  $\Sigma$  called the input alphabet,
3. A next-state function  $\delta : P \times \Sigma \rightarrow P$
4. An output alphabet  $\Phi$ ,
5. An output-symbol function  $\tau : P \times \Sigma \rightarrow \Phi$ ,
6. A start state  $p \in P$ .

The functions  $\delta$  and  $\tau$  can be described as matrices  $S$  and  $Q$  respectively. In particular,  $\delta(i, j) = S_{i,j}$  and  $\tau(i, k) = Q_{i,k}$ .

Any matrix can be represented as a string. Thus, if the set of states, alphabets, and start state are all fixed, then the representation can be string representations of the matrices  $S$  and  $Q$ . These then are fixed-length string representations.