Computer Science 420/594
Complex Systems and Self-Organization
Fall semester, 2002

Genetic Algorithms [1]

Motivation

The Genetic Algorithm (GA) is a search algorithm that is similar to depth-first search or to hill climbing, but it differs from the traditional search methods in that it is probabilistic and context insensitive. Only small changes are needed when using GA on different problems. For example, only the fitness function is modified when using a GA to solve a Traveling Salesman Problem (TSP) and to find a tic-tac-toe playing strategy.

Let s be a string. An important part of a GA is the fitness function $F(s)$. The input to the function is a binary string and the output is a number. $F(s)$ gives the GA feedback about its performance. For example, if the GA is asked to search for a date, $F(s)$ may pass back the difference between the actual date and the answer returned by the GA. The GA's answer is expected to improve as this process repeats.

Imagine the GA as a box and you are looking for a solution to some problem requiring search. As the box outputs an answer, you (acting as the fitness function) evaluate the merit of the solution by writing the merit value on paper and pass it back into the box. For many problems, even though the box does not have any contextual information about the problem it is working on, the box will produce solutions that are close to the optimal solution. One constraint when using GA is that you need to code possible answers as binary strings. For example, with a 16-city TSP problem, you could encode the solution as a string with 64 bits, with every 4 bits representing a city. Let C_j be city number j. A tour such as

$$[C_7, C_{12}, C_3, \ldots, C_8]$$

can be represented by

$$[011111000011\ldots1000]$$

The fitness function should decode the binary string and return the cost of the tour. For the date problem, you could encode the date as a 20-bit string where the first 4 bits are the month, the next 5 bits are the date, and the rest of bits are the year. There are many ways to encode GA solutions, and choice of coding is a factor that affects a GA's performance.

Description of Simple Genetic Algorithm (SGA)

Every genetic algorithm has a population of strings where each is a potential solution to the problem. There are 3 probabilistic operators on these strings: selection, crossover, and mutation. Let $\mathcal{P}_i$ be the population at ith generation, $S(\mathcal{P})$ be the selection operator, $C(s_i, s_j)$ be the crossover operator, $M(s_i)$ be the mutation operator, and $V(\mathcal{P})$ be the convergence operator. The algorithm for the SGA is

[1] Adapted by B. MacLennan from a handout prepared by D. Mutchler.

```
1    $\mathcal{P}_0 \leftarrow$ random population
1    For generation $i = 0$ do
1    Compute the next generation $\mathcal{P}_{i+1}$ from $\mathcal{P}_i$
2        For every $s \in \mathcal{P}_i$, calculate $F(s)$
2        While $\mathcal{P}_{i+1}$ is not full
3            $s_j \leftarrow S(\mathcal{P}_i)$
3            $s_k \leftarrow S(\mathcal{P}_i)$
3            $s_j, s_k \leftarrow C(s_j, s_k)$
3            $s_j \leftarrow M(s_j)$
3            $s_k \leftarrow M(s_k)$
3            Add $s_j$ and $s_k$ to $\mathcal{P}_{i+1}$
1    Until $V(\mathcal{P}_{i+1})$ is true
```

Below is an explanation of the pseudo code. Let ℓ be the string length, m be the mutation rate, and c be the crossover rate (the rates are real numbers in $[0, 1]$).

- $\mathcal{P}_0$: Form the initial population randomly. For example, if ℓ is 30 and $|\mathcal{P}|$ is 20, one can generate $\mathcal{P}_0$ by using $20 \times 30 = 600$ random binary values (perhaps from flipping a coin 600 times) and group the values into strings.

- $F(s)$: This is the fitness function.

- $S(\mathcal{P})$: The selection operator randomly picks a string from $\mathcal{P}_i$ with the probability of selecting s_i as
$$\frac{F(s_i)}{\sum_{j=1}^{|\mathcal{P}|} F(s_j)}$$
where the denominator is the total population fitness of $\mathcal{P}$.

- $M(s)$: For each bit of s, the mutation operator flips that bit with probability m.

- $C(s_i, s_j)$: With probability c, the crossover operator will select a random crossover point in $[1 \ldots \ell - 1]$ and exchange the tail portions of s_i and s_j. For example, if $s_i =$`11111111` and $s_j =$`00000000` are crossed at position 3, the result is $s_i =$`11100000` and $s_j =$`00011111`.

- Convergence: Let s^i be the ith bit of s (it can be either a 0 or a 1), τ be the convergence rate and
$$\gamma_i = \frac{1}{|\mathcal{P}|} \sum_{j=1}^{|\mathcal{P}|} s_j^i$$
be the sum of the ith bits of all the strings in the population. The population has converged if the convergence operator
$$V(\mathcal{P}) = \bigwedge_{i=1}^{\ell} \left(\gamma_i \geq \tau \bigvee \gamma_i \leq (1 - \tau) \right)$$
is `true`. The $\bigwedge$ and $\bigvee$ are logical `AND` and logical `OR` operators. In words, the population has converged when the ith-bit of all strings are either mostly 1s or mostly 0s, for all bit positions.