Genetic Algorithm

Kuei-Yuan Chan, chanky@ntu.edu.tw
Professor of Mechanical Engineering,
National Taiwan University.

Genetic algorithm is an optimization technique that draws its analogy from nature. The process of nature evolution intrigued John Holland of the University of Michigan in the mid-1960's. He developed computational techniques which simulated the evolution process and applied to mathematical programming. The genetic algorithm revolves around the genetic reproduction process and "*survival of the fittest*" strategy.

In the most commonly used GA, each variable is represented as a binary number of n bits, genome. This is conveniently carried out by dividing the feasible interval of variable x_i into $2^n - 1$ intervals. For $n = 6$, the number of intervals will be 63. Then each variable x_i can be represented by any of the discrete representations.

$$000000, 000001, 000010, \cdots, 111111 \tag{1}$$

A standard GA involves (1)creation of initial population; (2) evaluation of the 'fitting' function (objective) of each genome; (3) creation of a mating pool by replacing the weaker members; (4) reproducing offsprings by crossover; (5) performing random mutation operation to change the expected fitness; (6) evaluating the final populations and iterate.

Genetic-Algorithm (f', π, P_0)

$\triangleright \pi : C \mapsto S$ is a decoding function
$P \leftarrow P_0$ $\triangleright$ typically P_0 is random
for each $s \in P$ **do**
    $fits[s] \leftarrow f'(\pi(s))$ $\triangleright$ evaluate fitness
end for
while termination-condition()=FALSE **do**
    $Q \leftarrow \emptyset$
    while $|Q| < |P|$ $\triangleright$ assuming $|P|$ is evening **do**
        $s_1 \leftarrow select(P)$
        $s_2 \leftarrow select(P - \{s_1\})$
        $(s_1, s_2) \leftarrow crossover(s_1, s_2)$
        $s_1 \leftarrow mutate(s_1)$
        $s_2 \leftarrow mutate(s_2)$
        $fit[s_1] \leftarrow f'(\pi(s_1))$ $\triangleright$ evaluate fitness
        $fit[s_2] \leftarrow f'(\pi(s_2))$
        $Q \leftarrow Q \bigcup \{s_1, s_2\}$ $\triangleright$ duplication allowed
    end while

$P \leftarrow Q$
end while
return $\mathbf{x} \leftarrow \pi(s)$ where $fit[s] = \max_{t \in P} fit[t]$

Select (P)

$\triangleright$ returns a string $s \in P$ with probability $\frac{fit[s]}{\sum_{t \in P} fit[t]}$
first compute "total sum" $q = \sum_{t \in P} fit[t]$
$q \leftarrow 0$
for each $s \in P$ **do**
  $q \leftarrow q + fit[s]$
  $\triangleright$ linear search : $O(|P|)$ - return as soon as
  "partial sum" p exceeds $r \times q$
end for
$r \leftarrow random(0, 1) \triangleright r \in (0, 1]$
$p \leftarrow 0$
for each $s \in P$ **do**
  $p \leftarrow p + fit[s]$
  if $r \leq \frac{p}{q}$ **then**
    then return s
  end if
end for

crossover(s_1, s_2)

$\triangleright$ with probability P_{cross}, cross strings s_1 and s_2
$\triangleright$ with length l at a random location
$r \leftarrow random(0, 1) \triangleright r \in (0, 1]$
if $r \leq P_{\text{cross}}$ **then**
  $q \leftarrow random - integer(1, l - 1)$
  for $i \leftarrow 1$ to q $\triangleright$ copy first half **do**
    $t_1[i] \leftarrow s_1[i]$
    $t_2[i] \leftarrow s_2[i]$
    for $i \leftarrow q + 1$ to l $\triangleright$ copy second half **do**
      $t_1[i] \leftarrow s_1[i]$
      $t_2[i] \leftarrow s_2[i]$
    end for
  end for
else
  $t_1 \leftarrow s_1$ $\triangleright$ no crossover : just copy
  $t_2 \leftarrow s_2$

end if
return (t_1, t_2)

mutate(s)

$\triangleright$ flip each bit in s with low probability P_{mutate}
for $i \leftarrow 1$ to l **do**
    $r \leftarrow random(0, 1), \triangleright r \in (0, 1]$
    if $r \leq P_{\text{mutate}}$ **then**
        $t[i] \leftarrow \text{not}(s[i])$ $\triangleright$ flip a bit
    else
        $t[i] \leftarrow s[i]$ $\triangleright$ just copy
    end if
end for
return t