Evolutionary Computation

- Computational procedures patterned after biological evolution.
- Search procedure that probabilistically applies search operators to set of points in the search space.

- Lamarck and others:
  Species transmute over time
- Darwin and Wallace:
  Consistent, heritable variation among individuals in population
  Natural selection of the fittest

- Mendel and genetics:
  A mechanism for inheriting traits
  genotype -> phenotype mapping

- Types of evolutionary computation (EC):

  . Genetic Algorithm (GA)
  . Genetic Programming (GP)
  . Evolutionary Strategy (ES)
  . Evolutionary Programming (EP)

- Foundations of genetic algorithms (GAs): some major questions

(1) What are the law describing the behavior of schemas in GAs?
(2) How can we characterize the types of fitness landscapes on which the GA is likely to perform well?
(3) What does it mean for a GA to perform well? That is, what is the GA good at doing?
(4) How can we characterize the types of fitness landscapes on which the GA outperforms other search methods, eg. hill-climbing?

- Some implementation issues for genetic algorithms

. Representation:
How to best encode the problem to be solved.
. Fitness scaling:
How to scale fitness to maintain constant population and to achieve the best rate of evolution.
. Genetic operators:
Which operators to use; how often to apply various operators.
. Population size
. Maintaining diversity in the population
(to prevent the premature convergence)

- Basic genetic algorithm

```
begin
  initialize the generation number: $t \leftarrow 0$
  initialize the population at $t$: $P(t)$
  while (not termination-condition) do
  begin
    $t \leftarrow t+1$
    select $P(t)$ from $P(t-1)$
    recombine $P(t)$
    evaluate $P(t)$
  end
end
```

- Example of genetic algorithm

. maximize a function of k variables:

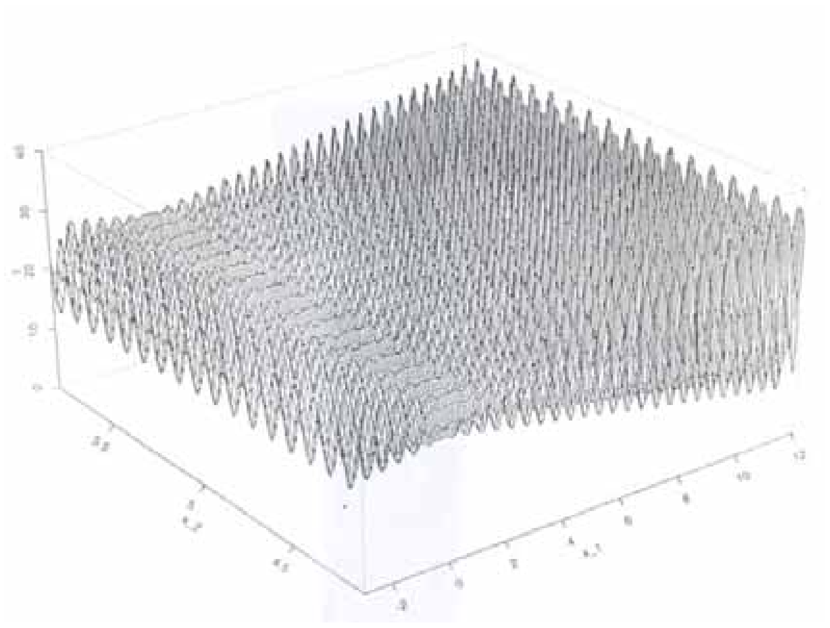
$$f(x_1, \cdots, x_k): R^k \rightarrow R$$

where $x_i \in [a_i, b_i]$.

eg. $f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$

where $-3.0 \leqq x_1 \leqq 12.1$ and $4.1 \leqq x_2 \leqq 5.8$.

. The plot of $f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$:



. Encoding of x_i

We will consider the bit string for each x_i.
Let the required precision be 6 decimal places.
Then,

$$(b_i-a_i)\cdot 10^6 \leqq 2^{m_i}-1$$

where m_i represents the smallest integer for binary representation.

Each bit string is interpreted by

$$x_i=a_i+deciaml(m_i\ bits)\cdot\frac{b_i-a_i}{2^{m_i}-1}.$$

For the given function $f(x_1, x_2)$, the input range is given by

$-3.0 \leqq x_1 \leqq 12.1$ and $4.1 \leqq x_2 \leqq 5.8$.

Let us consider 4 decimal digit precision for x_1 and x_2, that is,

$2^{17} < 15.1 \cdot 10^4 \leqq 2^{18}$ and $2^{14} < 1.7 \cdot 10^4 \leqq 2^{15}$.

In this case, we need $m = 18 + 15 = 33$ bits to encode a (x_1, x_2) pair, that is, each chromosome has 33 bits.

An example of chromosome:

$(010001001011010000111110010100010)$

The first 18 bits 010001001011010000 represents

$$x_1 = -3.0 + decimal(010001001011010000_2) \cdot \frac{12.1 - (-3.0)}{2^{18} - 1} = 1.052426.$$

The next 15 bits 111110010100010 represents

$$x_2 = 4.1 + decimal(111110010100010_2) \cdot \frac{5.8 - 4.1}{2^{15} - 1} = 5.755330.$$

So the chromosome

$(010001001011010000111110010100010)$

corresponds to $(x_1, x_2) = (1.052426, 5.755330)$.

The fitness value for this chromosome is

$f(1.052426, 5.755330) = 20.252640$.

. Evaluation

The fitness of the population is measured by

$$F = \sum_{i=1}^{N} eval(v_i)$$

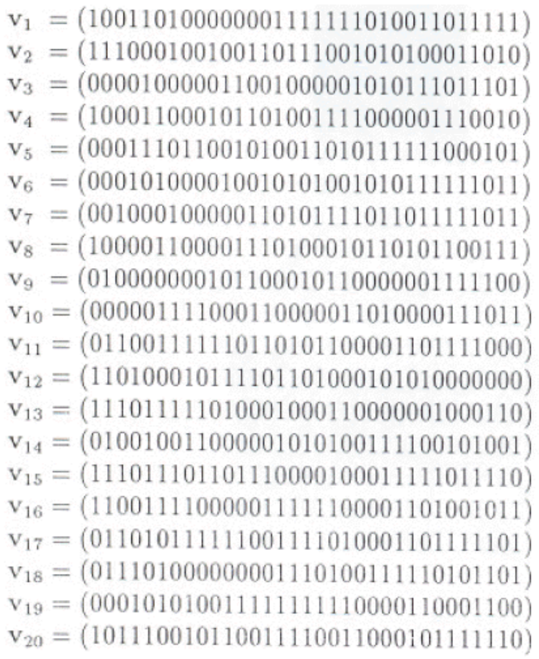
where N represents the population size and $eval(v_i)$ represents the fitness value for each chromosome v_i.

eg. $v_i = (m_1\ bit\ string\ \text{for}\ x_1, m_2\ bit\ string\ \text{for}\ x_2)$

$v_i \to (x_1, x_2)$ (decoding)

$eval(v_i) = f(x_1, x_2)$

. initial population:



$v_1 = (100110100000001111111010011011111)$
$v_2 = (111000100100110111001010100011010)$
$v_3 = (000010000011001000001010111011101)$
$v_4 = (100011000101101001111000001110010)$
$v_5 = (000111011001010011010111111000101)$
$v_6 = (000101000010010101001010111111011)$
$v_7 = (001000100000110101111011011111011)$
$v_8 = (100001100001110100010110101100111)$
$v_9 = (010000000101100010110000001111100)$
$v_{10} = (000001111000110000011010000111011)$
$v_{11} = (011001111110110101100001101111000)$
$v_{12} = (110100010111101101000101010000000)$
$v_{13} = (111011111010001000110000001000110)$
$v_{14} = (010010011000001010100111100101001)$
$v_{15} = (111011101101110000100011111011110)$
$v_{16} = (110011110000011111100001101001011)$
$v_{17} = (011010111111001111010001101111101)$
$v_{18} = (011101000000001110100111110101101)$
$v_{19} = (000101010011111111110000110001100)$
$v_{20} = (101110010110011110011000101111110)$

. evaluation of the fitness function values:

$$\begin{aligned}
eval(\mathbf{v}_1) &= f(6.084492, 5.652242) = 26.019600\\
eval(\mathbf{v}_2) &= f(10.348434, 4.380264) = 7.580015\\
eval(\mathbf{v}_3) &= f(-2.516603, 4.390381) = 19.526329\\
eval(\mathbf{v}_4) &= f(5.278638, 5.593460) = 17.406725\\
eval(\mathbf{v}_5) &= f(-1.255173, 4.734458) = 25.341160\\
eval(\mathbf{v}_6) &= f(-1.811725, 4.391937) = 18.100417\\
eval(\mathbf{v}_7) &= f(-0.991471, 5.680258) = 16.020812\\
eval(\mathbf{v}_8) &= f(4.910618, 4.703018) = 17.959701\\
eval(\mathbf{v}_9) &= f(0.795406, 5.381472) = 16.127799\\
eval(\mathbf{v}_{10}) &= f(-2.554851, 4.793707) = 21.278435\\
eval(\mathbf{v}_{11}) &= f(3.130078, 4.996097) = 23.410669\\
eval(\mathbf{v}_{12}) &= f(9.356179, 4.239457) = 15.011619\\
eval(\mathbf{v}_{13}) &= f(11.134646, 5.378671) = 27.316702\\
eval(\mathbf{v}_{14}) &= f(1.335944, 5.151378) = 19.876294\\
eval(\mathbf{v}_{15}) &= f(11.089025, 5.054515) = 30.060205\\
eval(\mathbf{v}_{16}) &= f(9.211598, 4.993762) = 23.867227\\
eval(\mathbf{v}_{17}) &= f(3.367514, 4.571343) = 13.696165\\
eval(\mathbf{v}_{18}) &= f(3.843020, 5.158226) = 15.414128\\
eval(\mathbf{v}_{19}) &= f(-1.746635, 5.395584) = 20.095903\\
eval(\mathbf{v}_{20}) &= f(7.935998, 4.757338) = 13.666916
\end{aligned}$$

. selection (roulette wheel selection)

The probability of selection p_i for each v_i is given by

$$p_i = eval(v_i)/F.$$

Then, the cumulative probability q_i for each v_i is given by

$$q_i = \sum_{j=1}^{i} p_j.$$

selection process:

(1) Generate a random number $r \in [0,1]$.

(2) If $r < q_1$, select v_1.

Otherwise, select v_i such that $q_{i-1} < r \leqq q_i$.

As a result, we construct a new population.

. the probability of a selection p_i for each chromosome v_i:

$$
\begin{array}{rr}
p_1 = eval(\mathbf{v}_1)/F = 0.067099 & p_2 = eval(\mathbf{v}_2)/F = 0.019547 \\
p_3 = eval(\mathbf{v}_3)/F = 0.050355 & p_4 = eval(\mathbf{v}_4)/F = 0.044889 \\
p_5 = eval(\mathbf{v}_5)/F = 0.065350 & p_6 = eval(\mathbf{v}_6)/F = 0.046677 \\
p_7 = eval(\mathbf{v}_7)/F = 0.041315 & p_8 = eval(\mathbf{v}_8)/F = 0.046315 \\
p_9 = eval(\mathbf{v}_9)/F = 0.041590 & p_{10} = eval(\mathbf{v}_{10})/F = 0.054873 \\
p_{11} = eval(\mathbf{v}_{11})/F = 0.060372 & p_{12} = eval(\mathbf{v}_{12})/F = 0.038712 \\
p_{13} = eval(\mathbf{v}_{13})/F = 0.070444 & p_{14} = eval(\mathbf{v}_{14})/F = 0.051257 \\
p_{15} = eval(\mathbf{v}_{15})/F = 0.077519 & p_{16} = eval(\mathbf{v}_{16})/F = 0.061549 \\
p_{17} = eval(\mathbf{v}_{17})/F = 0.035320 & p_{18} = eval(\mathbf{v}_{18})/F = 0.039750 \\
p_{19} = eval(\mathbf{v}_{19})/F = 0.051823 & p_{20} = eval(\mathbf{v}_{20})/F = 0.035244
\end{array}
$$

. recombination process:

(1) cross-over

the probability of cross-over: p_c

for each chromosome in the (new) population,

(a) generate a random number $r \in [0,1]$.

(b) if $r < p_c$, select the given chromosome for cross-over.

mate the selected chromosomes randomly.

for each pair of coupled chromosomes, generate

a random integer $\in [1, \cdots, m-1]$. (one-point cross-over) eg. $(b_1, b_2, \cdots, b_{pos}, b_{pos+1}, \cdots, b_m)$ ---> $(b_1, b_2, \cdots, b_{pos}, c_{pos+1}, \cdots, c_m)$

$(c_1, c_2, \cdots, c_{pos}, c_{pos+1}, \cdots, c_m)$ ---> $(c_1, c_2, \cdots, c_{pos}, b_{pos+1}, \cdots, b_m)$

(2) mutation

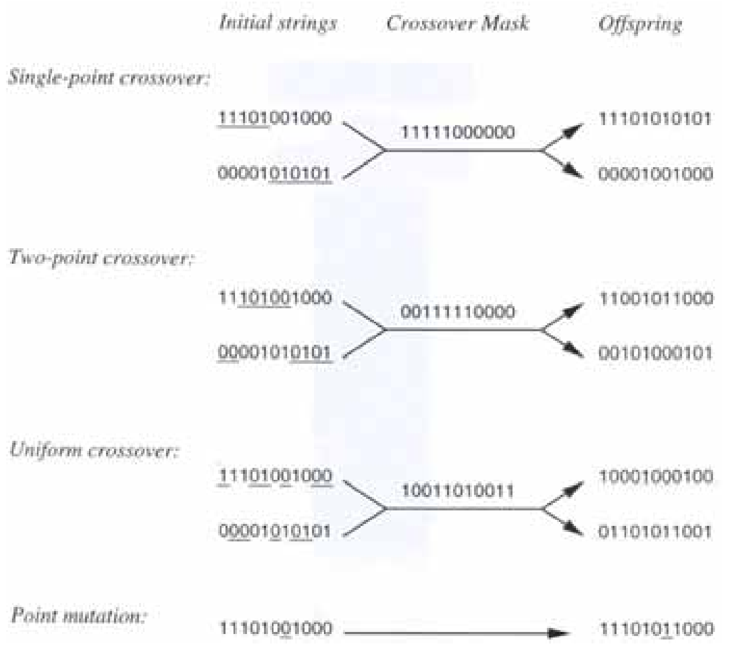
the probability of mutation: p_m

for each bit within the chromosome,

(a) generate a random number $r \in [0,1]$.

(b) if $r < p_m$, mutate the bit $(0 \rightarrow 1, 1 \rightarrow 0)$.

. cross-over and mutation operators



- Schema theorem (Holland, 1975)

. Schema: any string composed of 0s, 1s, and $\ast$s.

. Schema theorem characterizes the evolution of population by the number of instances representing each possible schema. Let $m(s,t)$ be the number of instances of schema s in population at time t.
Then, what is $E[m(s,t+1)]$ in terms of $m(s,t)$?
Here, we consider
$f(h)$ = the fitness of the individual bit string h,
$\bar{f}(t)$ = the average fitness of the population at time t,
n = the total number of individuals in the population,

$h \in s \cap P_t$, that is, the individual h is both a representative of schema s and a member of the population at time t, and
$\hat{u}(s,t)$ = the average fitness of instances of schema s in the population at time t.

. The probability of selecting h in one selection step:

$$\Pr\{h\} = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} = \frac{f(h)}{n\bar{f}(t)}$$

. The probability of selecting an instance of s:

$$\Pr\{h \in s\} = \sum_{h \in s \cap P_t} \frac{f(h)}{n\bar{f}(t)} = \frac{\hat{u}(s,t)m(s,t)}{n\bar{f}(t)}$$

. The expected number of instances of s resulting from the n independent selection steps that create the entire new generation is given by

$$E[m(s,t+1)] = n \cdot \Pr\{h \in s\} = \frac{\hat{u}(s,t)m(s,t)}{\bar{f}(t)}. \quad \ldots \quad (1)$$

. recombination process: cross-over

Let $d(s)$ be the distance between the leftmost and rightmost defined bits in s and l be the length of the individual bit strings. Then, the probability of destruction of a schema s is given by

$$p_d(s) = \frac{d(s)}{l-1} \quad \text{and}$$

the probability of schema survival is given by

$$p_s(s) = 1 - p_d(s) = 1 - \frac{d(s)}{l-1}.$$

Let p_c be the probability of cross-over. Then,

$$p_s(s) \geqq 1 - p_c \frac{d(s)}{l-1}. \quad \ldots \quad (2)$$

. recombination process: mutation

Let p_m be the probability of mutation. Then, the probability of single bit survival is $1-p_m$ and the probability of schema survival is given by

$$p_s(s) = (1-p_m)^{o(s)}$$

where $o(s)$ represents the number of bits (non $\ast$s) in s.

Since $p_m \ll 1$ (usually set as $1/l$),

$p_m \approx 1 - o(s) \cdot p_m$. ... (3)

Therefore, from (1), (2), and (3),

$$E[m(s,t+1)] \geqq \frac{\hat{u}(s,t)m(s,t)}{\bar{f}(t)}\left(1 - p_c\frac{d(s)}{l-1}\right)(1-p_m)^{o(s)}$$
$$\geqq \frac{\hat{u}(s,t)m(s,t)}{\bar{f}(t)}\left(1 - p_c\frac{d(s)}{l-1} - o(s)p_m\right)$$

This implies that the above-average schema with short defining length $d(s)$ and low-order (small $o(s)$) would still be sampled at exponentially increased rate.

. premature convergence (or crowding)

As the number of generations is increased, the chromosomes with high fitness score tend to dominated in the population. This premature convergence is not desirable from the view point of the diversity of the population.

. sharing

To prevent the premature convergence the fitness function is modified by

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} s(d(x_i, x_j))}$$

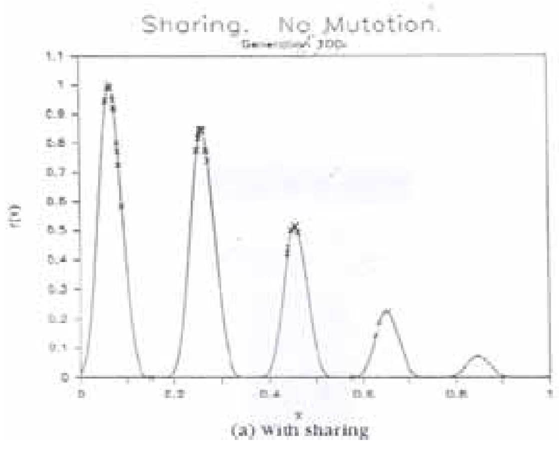
where x_i represents the ith chromosome,

$d(x_i, x_j)$ represents the distance between x_i and x_j, and $s(d)$ represents the sharing function.

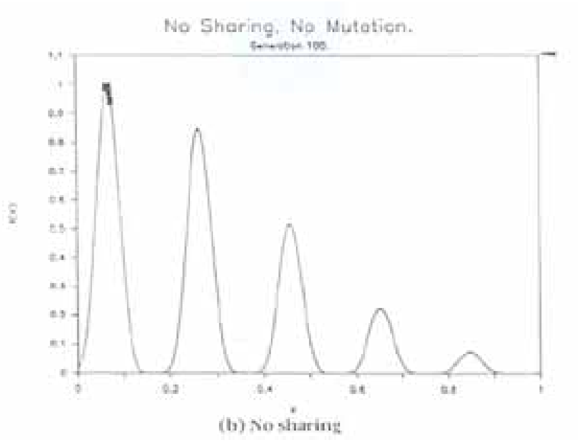
eg. $s(d) = \frac{-x}{d_{\max}} + 1$

This sharing function gives more spreading distribution of chromosomes in the solution space.

. Example of sharing



(a) With sharing



(b) No sharing

- Convergence of genetic algorithm (Rudolph, 1994)

. GA can be analysed by a finite Markov chain describing a probability trajectory over a finite state space S of cardinality $|S| = N$.
eg. Let l be the number of genes and n be the population size.
Then, $N = 2^{ln}$.

. Let us consider the transition matrix $P = [p_{ij}]$ in which

$$p_{ij} \in [0,1] \quad \text{and} \quad \sum_{j=1}^{|S|} p_{ij} = 1 \quad \forall i \in S.$$

Then, the distribution of the chain after the tth step is

$$p^t = p^0 P^t$$

where p_0 represents the initial distribution as a row vector.
Here, the homogeneous finite Markov chain
(P is not changing over time) is completely described by a pair (p^0, P).

. Definitions:
(1) A is positive if $a_{ij} > 0 \quad \forall i,j \in \{1, \cdots, n\}$.
(2) A is primitive if $\exists k \in N$ (integer) such that A^k is positive.

(3) A is reducible if A can be brought into the form

$$\begin{pmatrix} C & O \\ R & T \end{pmatrix}$$

(with square matrices C and T) by applying the same permutations to rows and columns.

(4) A is stochastic if $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i \in \{1, \cdots, n\}$.

(5) A is stable if it has identical rows.

. Theorem: Let P be a *primitive stochastic matrix*. Then, p^k converges as $k \to \infty$ to a positive stable stochastic matrix p^∞ where $p^\infty = p^0 \lim_{k \to \infty} P^k = p^0 P^\infty$ has non-zero entries and

is unique regardless of the initial distribution.

. The GA procedure can be described by

$P = CMS$

where $C, M,$ and S describe the intermediate transition caused by cross-over, mutation, and selection.

. Lemma: P is primitive

(proof)

Since C is stochastic, there exists at least one positive entry in each row of C. The matrix M is positive and S has at least one positive entry in each column. Thus, $A = CM$ is positive. Then, $P = AS$ is positive. Therefore, P is primitive.

. Theorem: The GA maintaining the best solution found over time before selection converges globally optimal.

(proof)

Since P is primitive, there exists p^{∞} which has unique non-zero entries. This implies that $p_i > 0 \quad \forall i \in S$.

Since GA keeps the best solution, the globally optimal solution can be found with the probability of 1.

- Issues on Genetic Algorithms

. How do they work?

efficient genetic operators

implicit parallelism

. What are they good for?

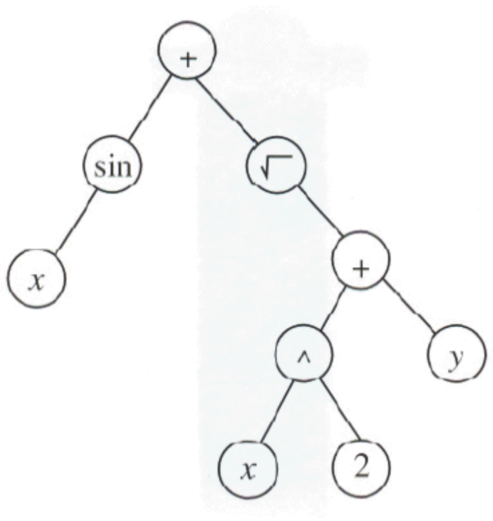
How to characterize class of problems on which they will work well?

- Some applications of genetic algorithms

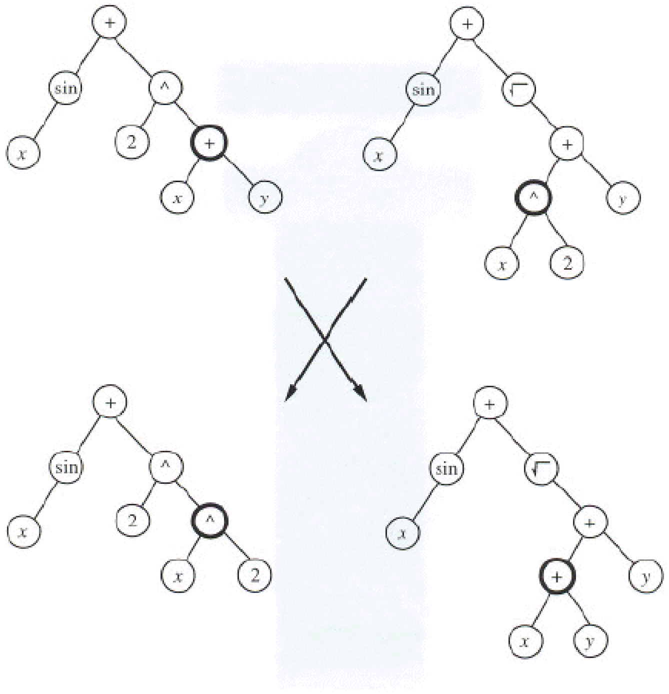
. optimization
  eg. numerical optimization, circuit design, factory scheduling
. automatic programming
  eg. evolving optimal sorting algorithms, evolving Lisp programs
. machine and robot learning
  eg. robot navigation, evolving artificial neural networks
. complex data analysis and time-series prediction
  eg. weather prediction, financial market prediction,
    protein structure prediction
. scientific models of adaptive complex systems
  eg. economics, immunology, ecology, population genetics

- Genetic Programming (GP)

. Programs in population are represented by trees
  eg. $\sin(x) + \sqrt{x^2 + y}$



. cross-over in GP



- Evolutionary Computation Conclusions

. EC defines a class of search methods modeled after natural evolution.
. Performs a randomized beam search over a hypothesis space.
. EC methods can search any hypothesis space.
. Global, multi-point search rather than local, single point search.
. Do not follow gradients: if gradient information can be efficiently calculated and effective to search the solution, EC method may not be the best search method.
. Very easy to apply to a wide range of problems.

- References

. Goldberg D., "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.
. Davis L. (Ed.), "Handbook of Genetic Algorithms," New York, Van Nostrand Reinhold, 1991.
. Michalewicz, Z., "Genetic Algorithms + Data Structures = Evolution Programs," New York, Springer-Verlag, 1992.
. Whitley, D. (Ed.), "Foundations of Genetic Algorithms 2," San Mateo, Morgan Kaufmann, 1993.
. Baek T. (Ed.), "Handbook of Evolutionary Computation," IOS Press, 1997.

. Banzhaf W. et al., "Genetic Programming: An Introduction," Morgan Kaufmann, 1999.
. Yao X. (Ed.), "Evolutionary Computation: Theory and Applications," World Scientific, 1999.