Lecture Note

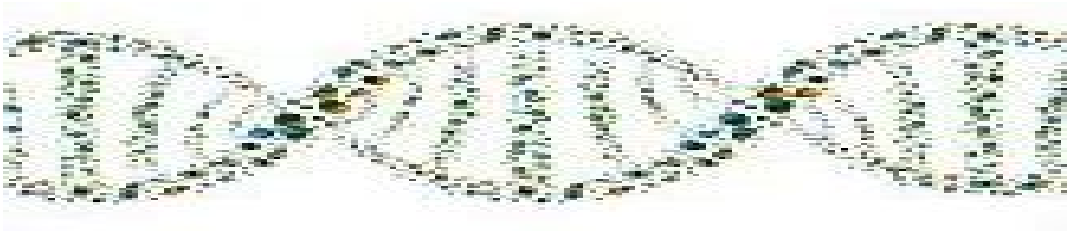
An Introduction to Bio-inspired Computation

Akira Imada
Brest State Technical University

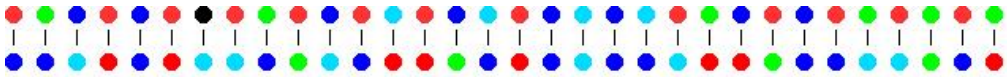
Last modified on 10 November 2016

I. Idea borrowed from biological evolution

An Image of DNA Spiral



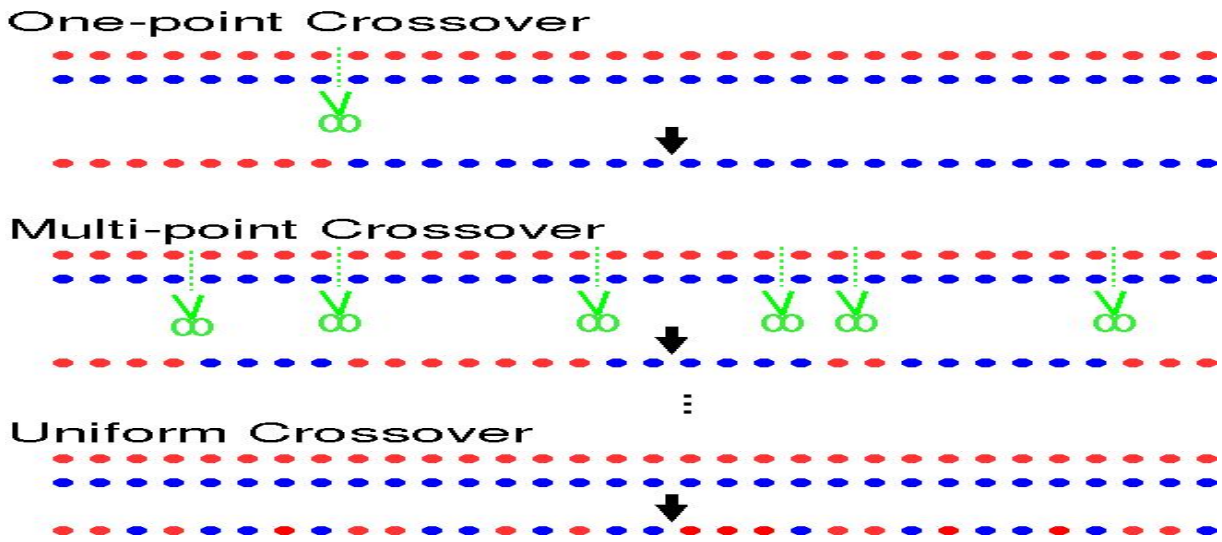
Simplification



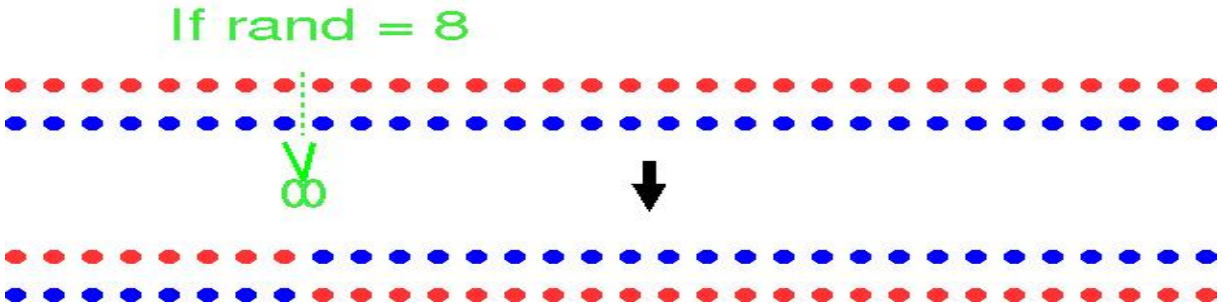
● ... adenine ● ... cytosine ● ... guanine ● ... thymine

Our virtual chromosome

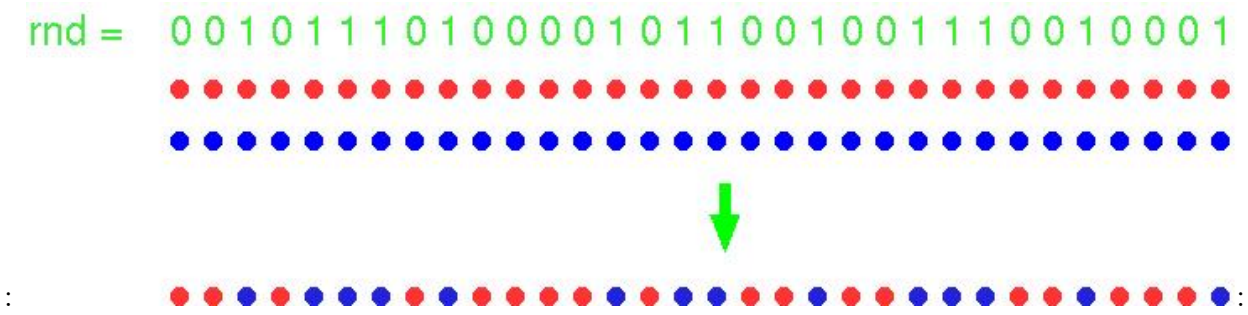
Crossover



One-point crossover with random number



Uniform crossover with random number



Mutation

Usually probability (mutation) = $\frac{1}{\text{number of genes}}$

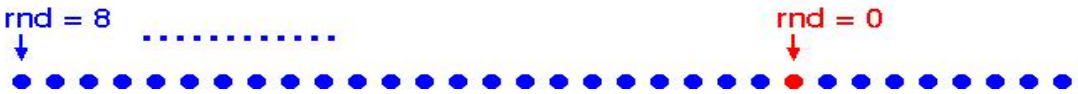
↓

Create random number from 0 to N - 1 each of the all genes

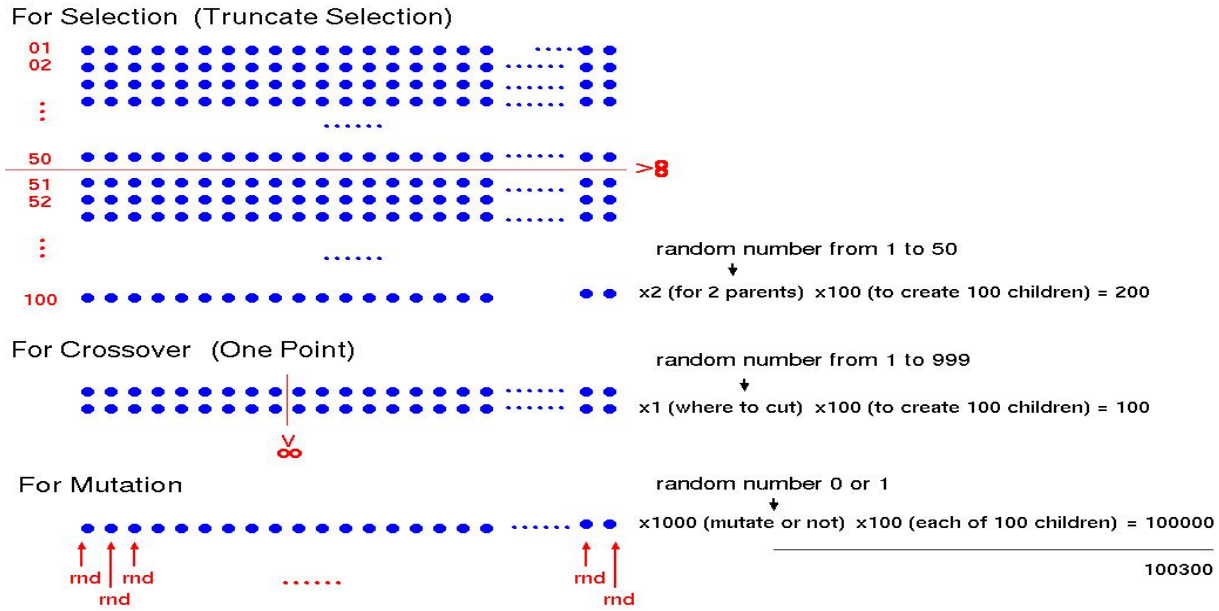
↓

If and only if random number = 0 then mutate

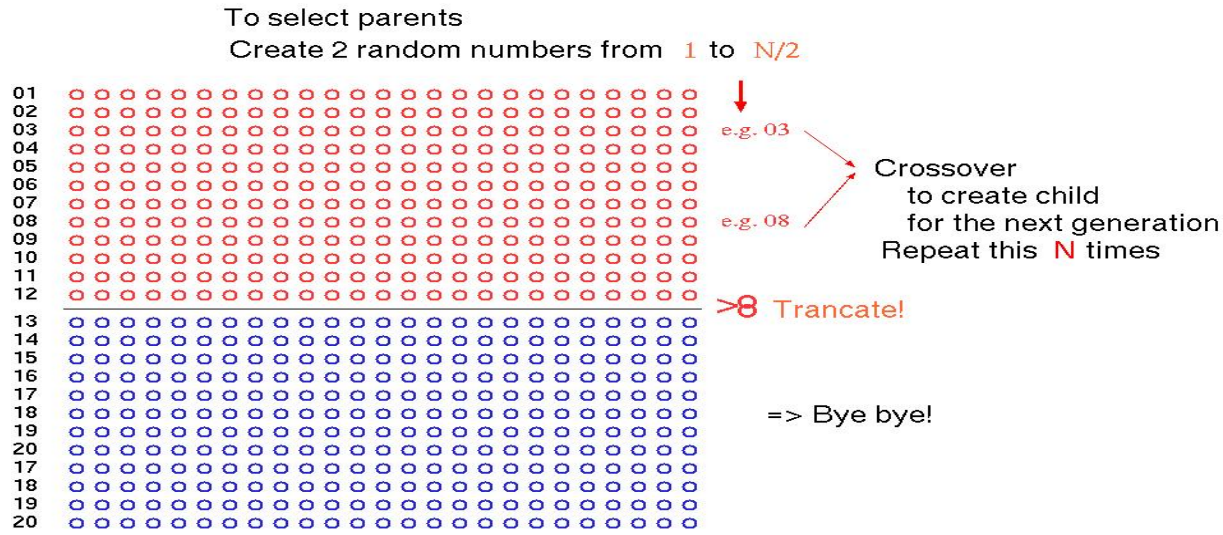
↓



How Many Mutations are necessary in one generation?

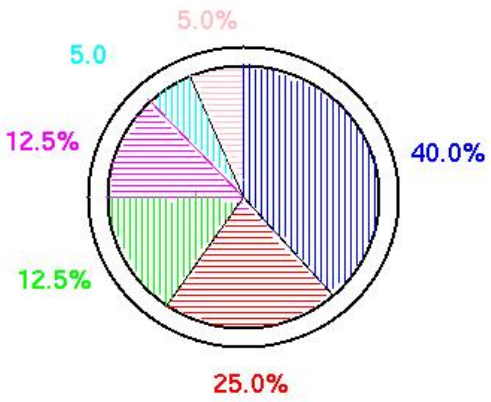


Truncate Selection



Fitness Proportionate Selection

	fitness	
#1	16	=>16/40 = 0.400
#2	10	=>10/40 = 0.250
#3	5	=> 5/40 = 0.125
#4	5	=> 5/40 = 0.125
#5	2	=> 2/40 = 0.500
#6	2	=> 2/40 = 0.500
	40	



a.k.a

Roullett Wheel Selection

II. Simplest version - Hill Climbing

Random Mutation Hill-climbing

(1) choose a string at random and call this current-hilltop

(2) choose a locus at random to flip. If the flip leads to an equal or higher fitness then set current-hilltop to the resulting string

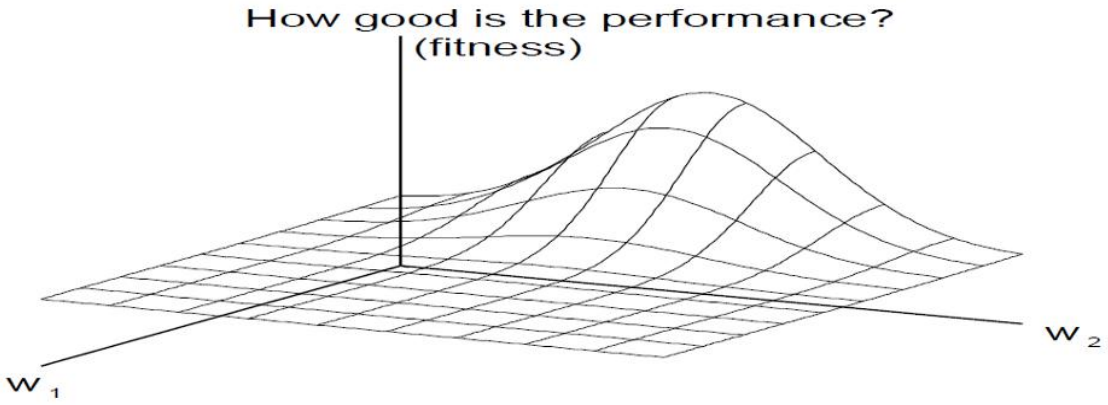
(3) goto step (2) until an optimum string has been found or until a maximum number of evaluations have been performed.

(4) return the current-hilltop

III. The 1st toy example
All one problem

IV. Fitness landscape

A conceptual plot of fitness value defined on a fictitious 2-D space



V. The 2nd toy example
A needle in a haystack problem

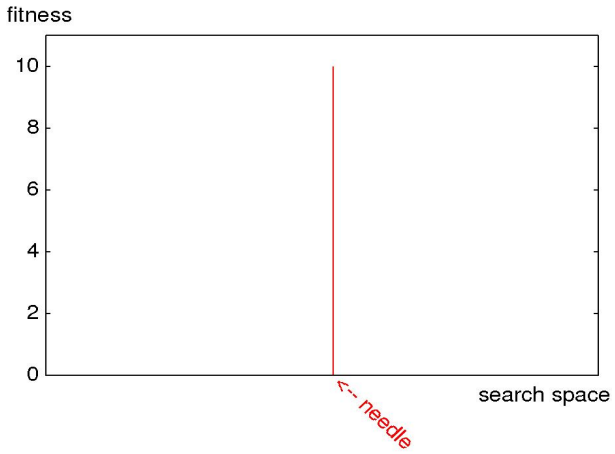
N-digit password matching problem

Assume N-digit integer password and Chromosome with N integer genes

Fitness is 1 if and only if chromosome match the password perfectly

otherwise fitness = 0.

Fitness landscape int this case

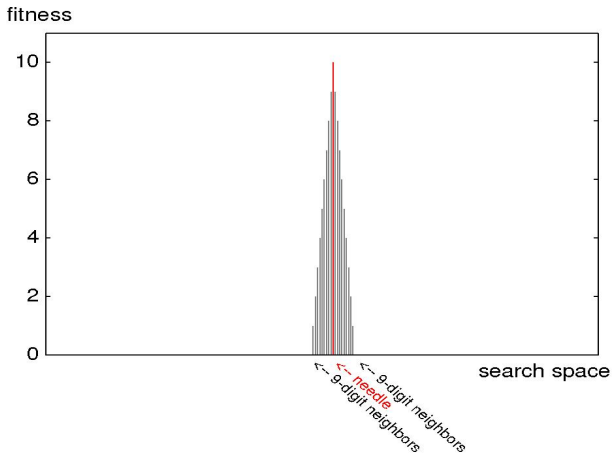


fitness	generation
0	5
1	986
2	3430
3	10152
4	25365
5	52982
6	95682
7	198253
8	402362
9	623023

(Right figure is from the student work by Bakun Anton in 2015)

An extended fitness evaluation

When we evaluate its fitness by "how many digits match?"



password	iteration	n
807	1473	3
3859	25237	4
03186	41489	5
723186	772246	6
9224186	3194682	7
35284196	16185762	8
874284695	78326820	9
7803204905	eror ! Very long time	10

(Right figure is from the student work by Bakun Anton in 2015)

VI. The 3rd toy example
Lucky dog

A dog lookling for a sausage

A dog in the gridworld (0,0)-(1000,1000) with sausage at (200,800)

Dog starts from (500,500) looking for the sausage.

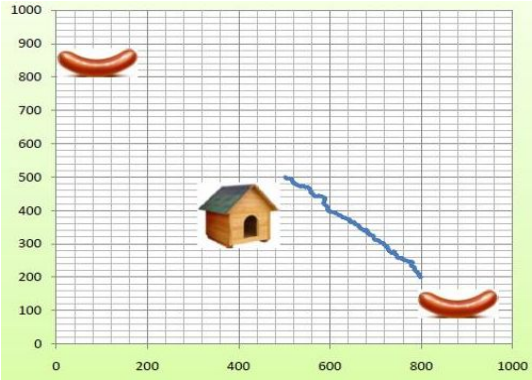
VII. What if more than one solution exsit?

A dog lookling for two sausages

A dog in the gridworld (0,0)-(1000,1000) with sausage at (200,800) and (800,200)

Dog starts from (500,500) looking for the sausages.

(From the student work by Belous Sophia in 2015)



In fact

All the dogs gather only one sausage out of the two!

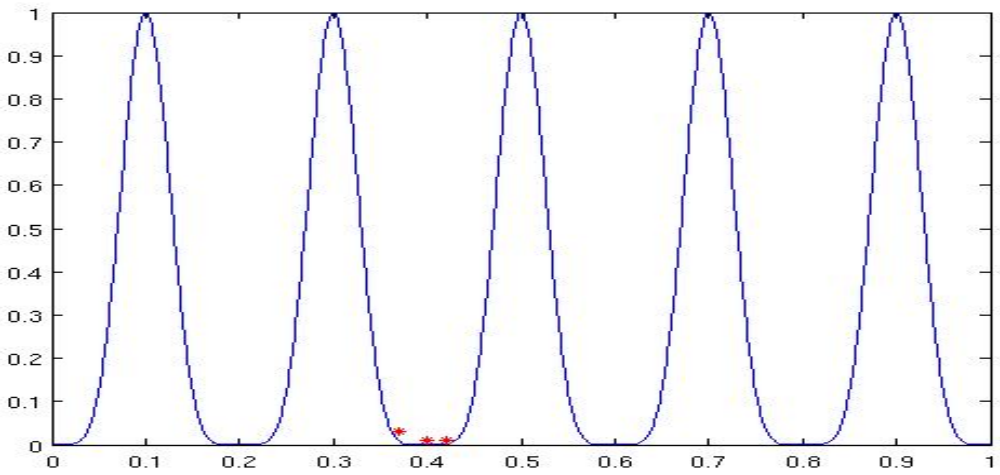
⇓

Can we implement so that

half of the dogs go to the one while the other half to the other?

Or, a 2-D function minimization

$$y = \sin^6(5\pi x)$$



Two algorithms to get all the solutions

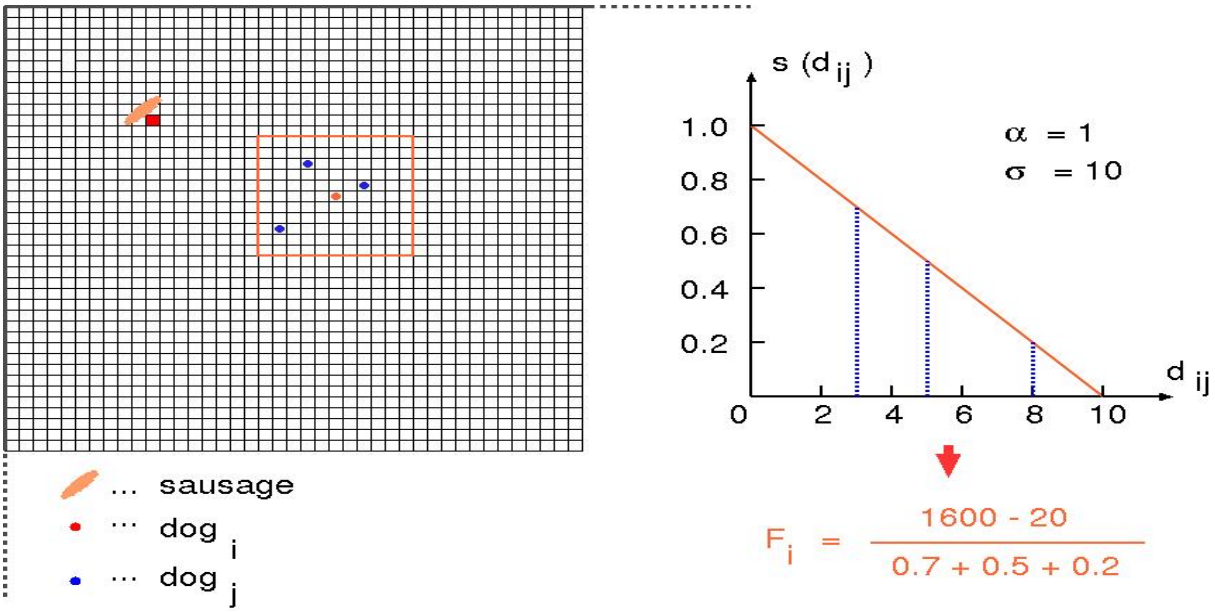
1. Fitness sharing

$$F_s(i) = \frac{F(i)}{\sum_{j=1}^{\mu} s(d_{ij})}$$

where

$$s(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{\text{share}})^{\alpha} & \text{if } d_{ij} < \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases}$$

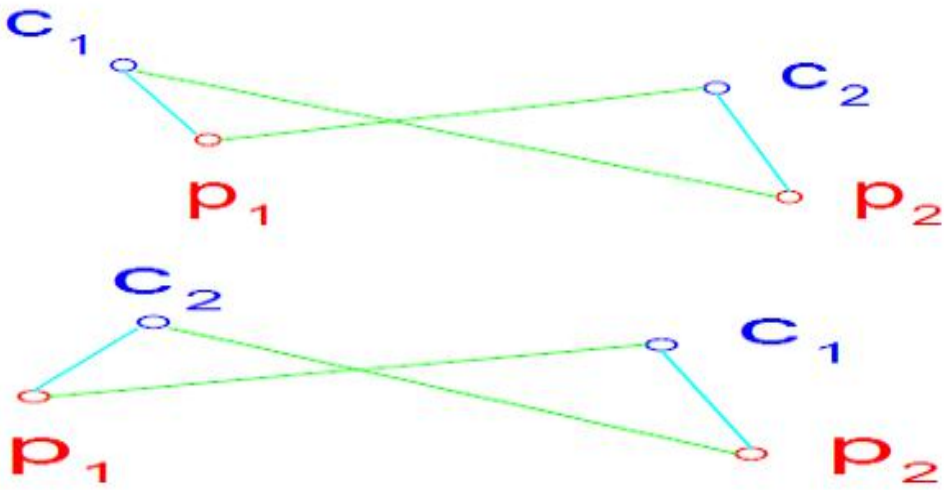
Fitness Share - Lucky Dog



2. Crowding Algorithm

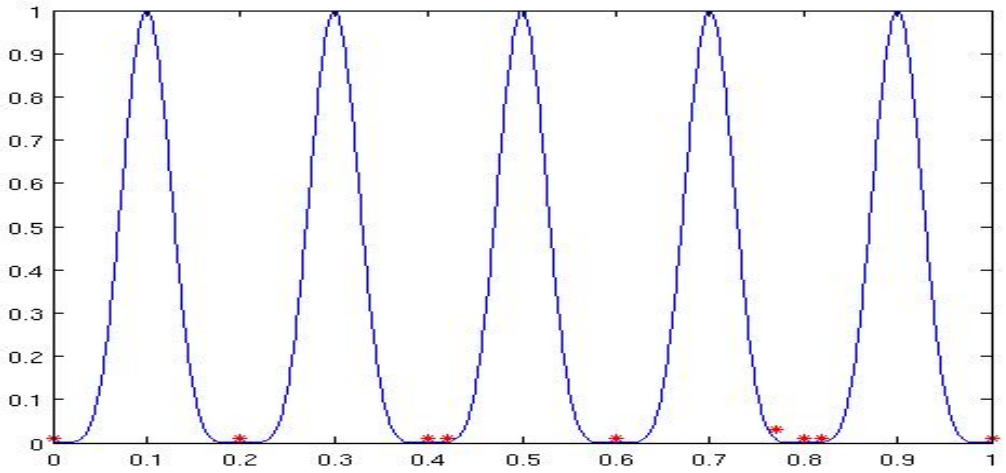
1. Choose two parents, p_1 and p_2, at random.
2. Produce two children, c_1' and c_2'.
3. Mutate the children yielding c_1 and c_2, with a crossover.
4. Replace parent with child as follows:
   - IF $d(p_1, c_1) + d(p_2, c_2) > d(p_1, c_2) + d(p_2, c_1)$
     * IF $f(c_1) > f(p_1)$ THEN replace p_1 with c_1
     * IF $f(c_2) > f(p_2)$ THEN replace p_2 with c_2
   - ELSE
     * IF $f(c_2) > f(p_1)$ THEN replace p_1 with c_2
     * IF $f(c_1) > f(p_2)$ THEN replace p_2 with c_1

Crowding - two cases of parents & children



2-D function minimization - What happens?

$$y = \sin^6(5\pi x)$$



VIII. Commonly used test function

1. Sphere model

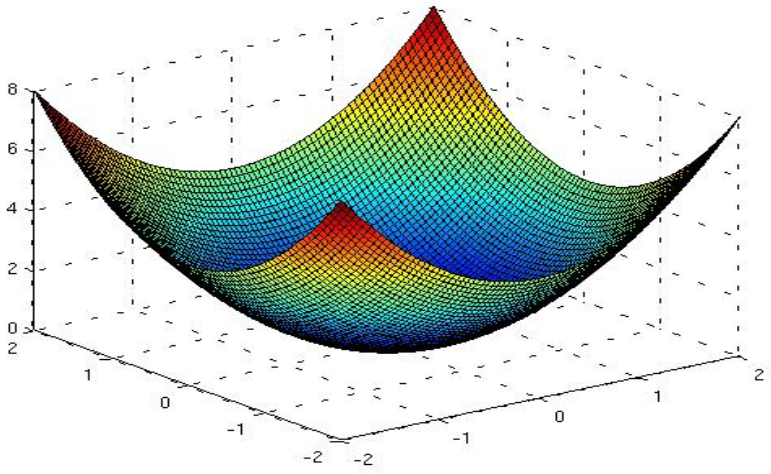
$$y = \sum_{i=1}^{20} x_i^2$$

That is,

$$y = x_1^2 + x_2^2 + x_3^2 + \cdots + x_{20}^2$$

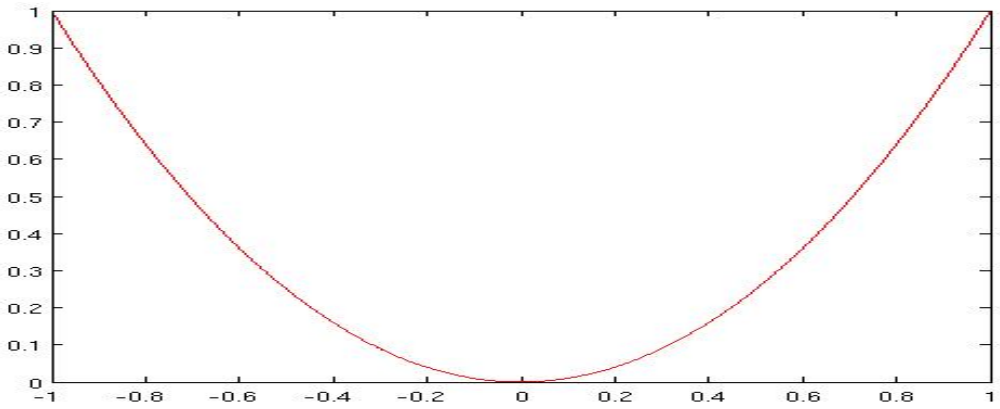
It's 3-D version

$$z = x^2 + y^2$$



It's 2-D version

$$y = x^2$$



2. Shcwefel function

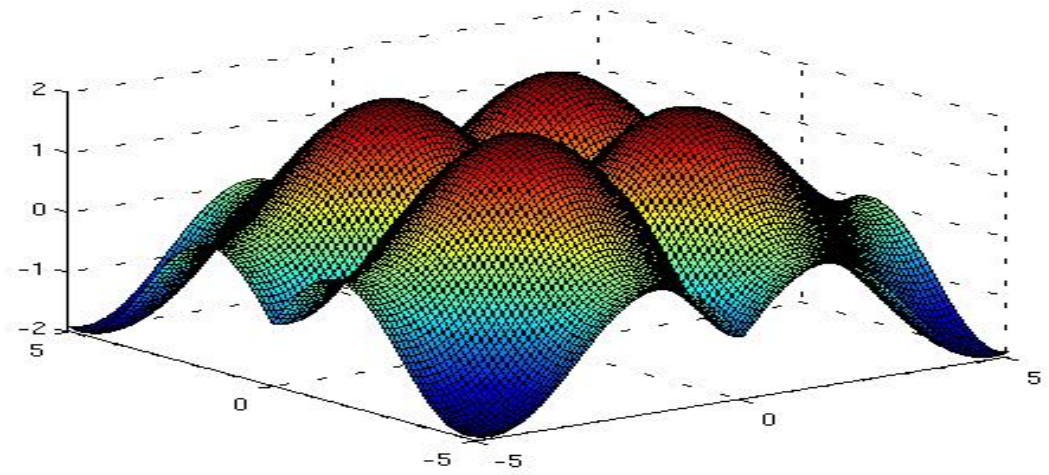
$$y = \sum_{i=1}^{n} (x_i \sin(|x_i|))$$

That is,

$$y = x_1 \sin(|x_1|) + x_2 \sin(|x_2|) + \cdots + x_{20} \sin(|x_{20}|)$$

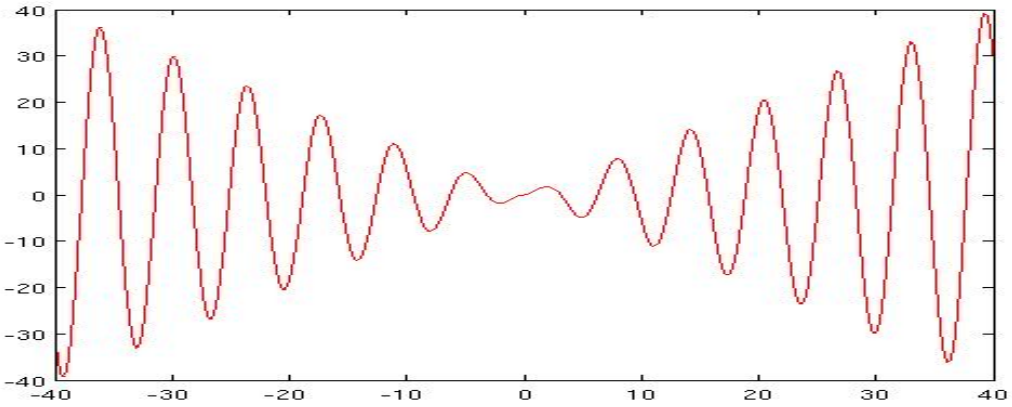
Shcwefel - It's 3-D version

$$z = x\sin(|x|) + y\sin(|y|)$$



Schwefel - It's 2-D version

$$y = x \sin(|x|)$$

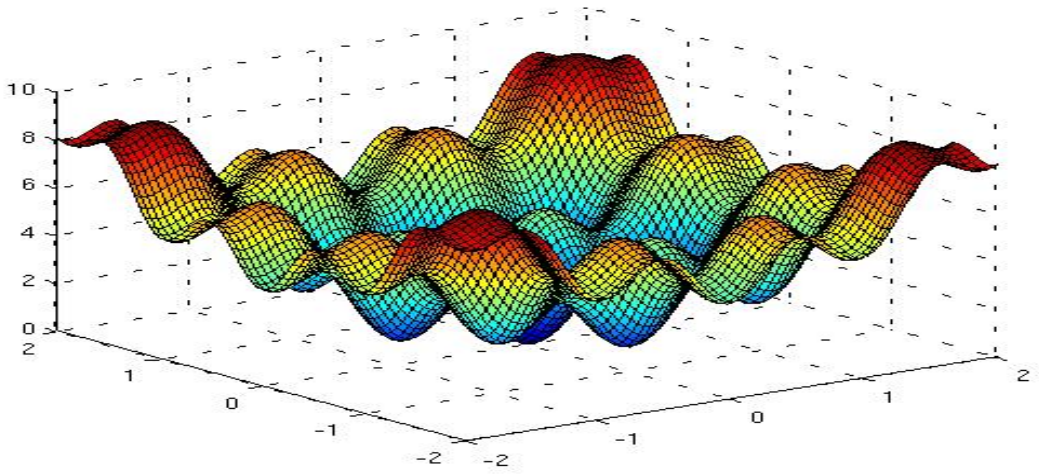


3. Rastrigin's function

$$y = nA + \sum_{i=1}^{n} (x_i^2 - A\cos(2\pi x_i))$$

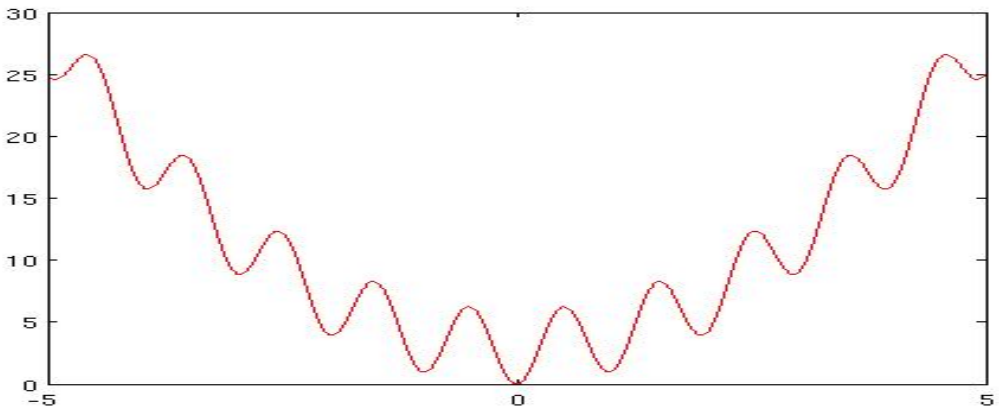
Rastrigin - It's 3-D version

$$z = 2 + x^2 - \cos(2\pi x) + y^2 - \cos(2\pi y)$$



Rastrigin - It's 2-D version

$$y = 3 + x^2 - 3\cos(2\pi x)$$

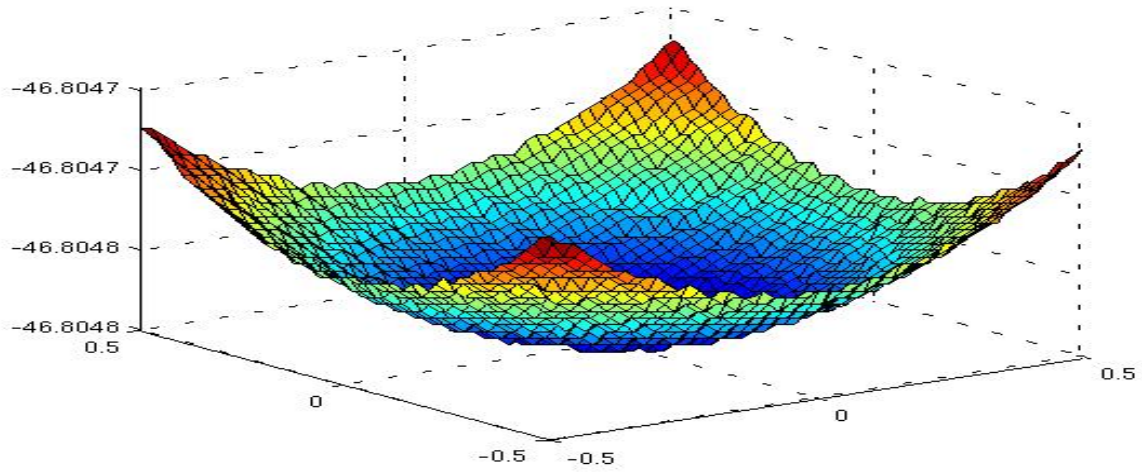


4. Griewangk's function

$$y = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$$

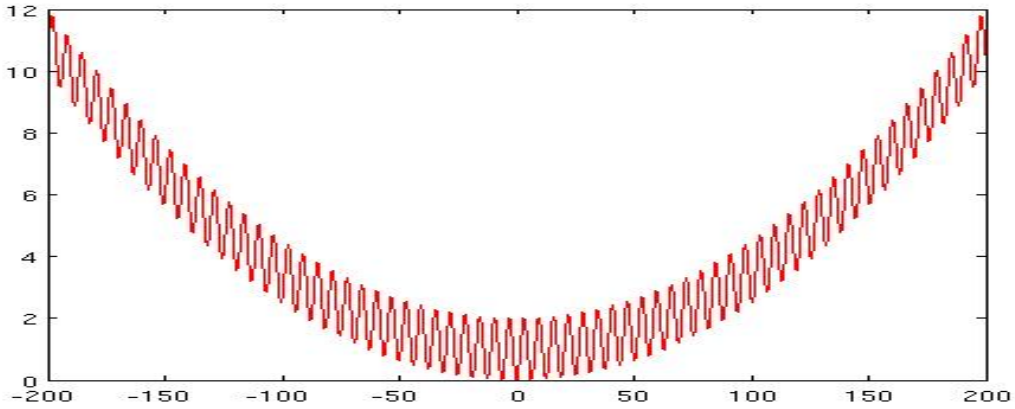
Griewangk - It's 3-D version

$$z = \frac{x^2 + y^2}{4000} - (\cos x)\cos(x/\sqrt{2}) + 1$$



Griewangk - It's 2-D version

$$y = \frac{x^2}{4000} - \cos x + 1$$

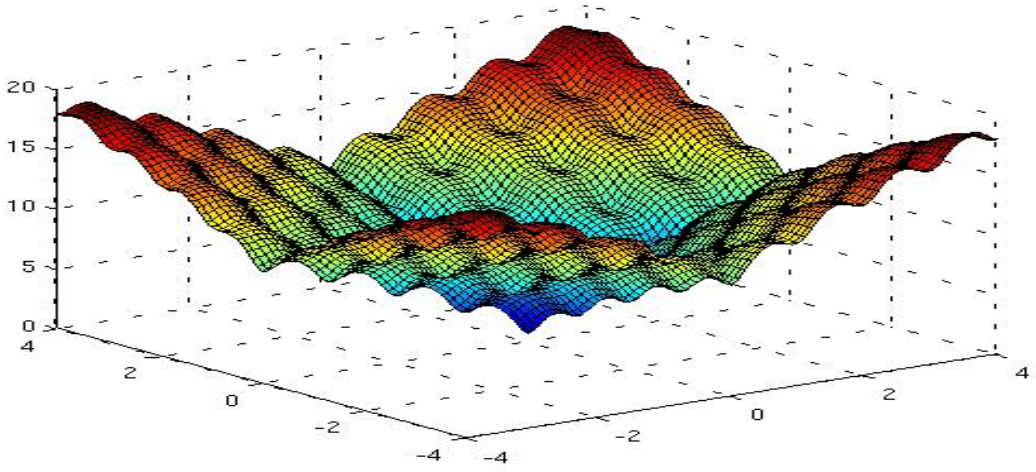


5. Ackley's function

$$y = -20 \sum_{i=1}^{n} \exp\left(-0.2\sqrt{x_i^2/n}\right) - \exp\left(\left(\sum_{i=1}^{n} \cos 2\pi x_i\right)/n\right) + 20 + e$$

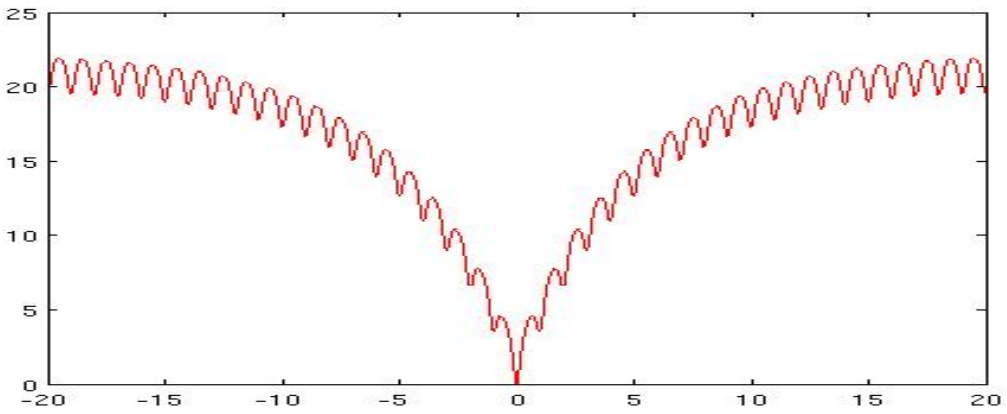
Ackley - It's 3-D version

$$z = -20\exp -0.2\sqrt{(x^2+y^2)/2} - exp(\cos 2\pi x + \cos 2\pi y)/2) + 20 + e$$



Ackley - It's 2-D version

$$y = -20\exp\left(-0.2\sqrt{x^2}\right) - \exp\left(\cos 2\pi x\right) + 20 + e$$



IX. What if more than one fitness function exsit?

Parete Optimal Solution

If X is better than Y for all the Fitnesses then it is said that X dominates Y.

When a solution is not dominated by any others it is called
Non Dominated Solution
or
Parete Optimal Solution

Rank - When we try evolution for Parete optimum

Order according to

the number of how many others in the population it dominates.

E.g. who dominates whom and how rank is counted.

	test-1	test-2	test-3	test-4	test-5	dominated by	dominates	rank
A	9	9	9	8	7	0	3	1
B	5	4	5	3	6	1	2	2
C	3	3	4	2	3	2	1	3
D	2	2	3	1	2	3	0	4
E	1	1	1	1	9	0	0	4

Algorithm

1. *Initialize the population.*
2. *Select individuals uniformly from population.*
3. *Perform crossover and mutation to create a child.*
4. *Calculate the rank of the new child.*
5. *Find the individual in the entire population that is most similar to the child. Replace that individual with the new child if the child's ranking is better, or if the child dominates it.*
6. *Update the ranking of the population if the child has been inserted.*
7. *Perform steps 2-6 according to the population size.*
8. *If the stop criterion is not met go to step 2 and start a new generation.*

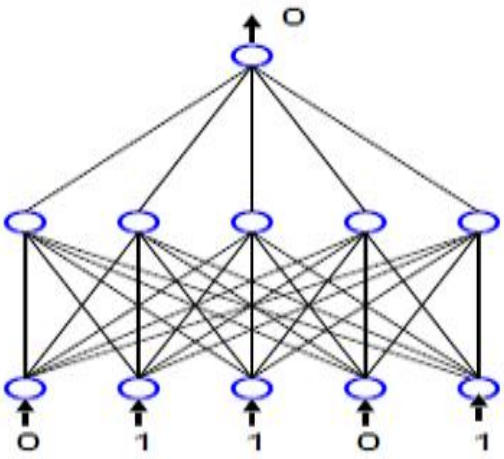
Excercise

Try it with two fitness functions $y_1 = (x-2)^2$ *and* $y_2 = (x-4)^2$ *as follows:*

1. *Create 20 10-bit binary chromosomes, assuming each chromosome represent x-cordinate ranges from 0 to 6 with* $(0000\cdots00)$ *and* $(1111\cdots11)$ *being corresponding to 0 and 6, respectively.*
2. *Calculate* y_1 *and* y_2 *for each of 20 x's represented by these 20 chromosomes.*
3. *Create a table with 5 columns: (i) chromosome, (ii) its x value, (iii) its* y_1 *value, (iv) its* y_2 *value, (v) how many these* (y_1, y_2) *dominates others (rank).*
4. *Plot these 20 points on each of two graphs (e.g., red and blue color).*
5. *Create next generation by applying the algorithm on these 20 chromosomes.*
6. *While 20 points are different from previous generation Do 2-4 Else stop.*

X. Neural Network
Weight configutation by evolution

5-even-parity by Neural Network



input	output
00000	1
00001	0
00010	0
00011	1
00100	0
00101	1
00110	1
00111	0
01000	0
...	...
11110	1
11111	0

Feedforward neural network

Out put y of one neuron with 5 inputs of $+1$ or -1 is calculated as:

$$y = sgn(\sum_{i=1}^{5} w_i x_i)$$

that is,

if $\sum_{i=1}^{5} w_i x_i$ is negative then $y = -1$, otherwise $y = 1$

Evolution of weight configuration

Chromosomes here are:

$$(w_1, w_2, w_3, \cdots, w_{30})$$

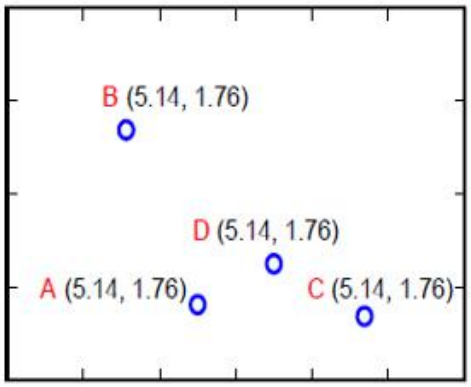
Fitness is:
how many outputs are collect

when each of all possible $2^5 = 32$ inputs is given one by one?

XI. Traveling Salesperson Problem (TSP)

E.g. TSP with 4 cities)

Map of the 4 cities



Distance Matrix

	A	B	C	D
A	0.00	4.13	4.34	1.95
B	4.13	0.00	7.35	5.00
C	4.34	7.35	0.00	2.51
D	1.95	5.00	2.51	0.00

All the possible routes

A-B-C-D-A => 4.13 + 7.35 + 2.51 + 1.95 = 15.94

A-B-D-C-A => 4.13 + 5.00 + 2.51 + 4.34 = 15.98

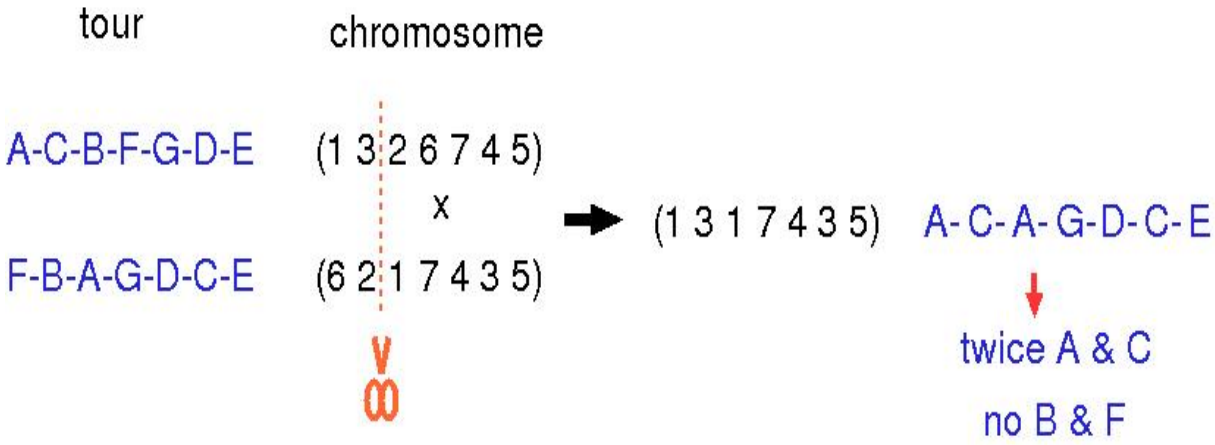
A-B-C-D-A => 4.34 + 7.35 + 5.00 + 1.95 = 18.64

E.g. A travel (A-C-B-F-C-D-E).

Can it be encoded to (1 3 2 6 3 4 5)? ⇓

The answer is no!

One-point crossover, e.g., wouldn't work



Then how to encode TSP into a chromosome?

Step-1. Set $i = 1$.

Step-2. If i-th gene is n then n-th city in the list is the city to be currently visited.

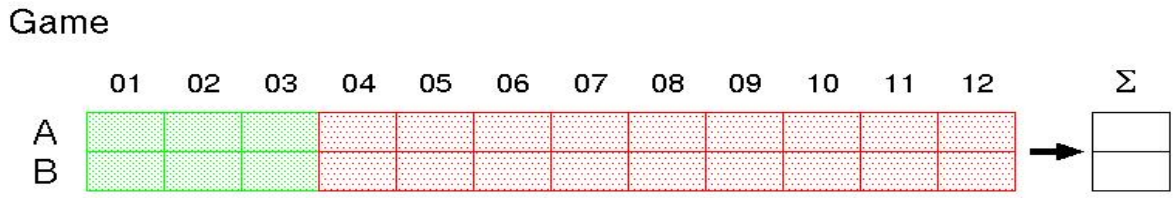
Step-3. Remove the city from the list.

Step-4. Set $i = i + 1$ and repeat Step-2 to Step-4 while $i \leq n$.

XII. Iterated Prisonner's Dilemma (IPD)

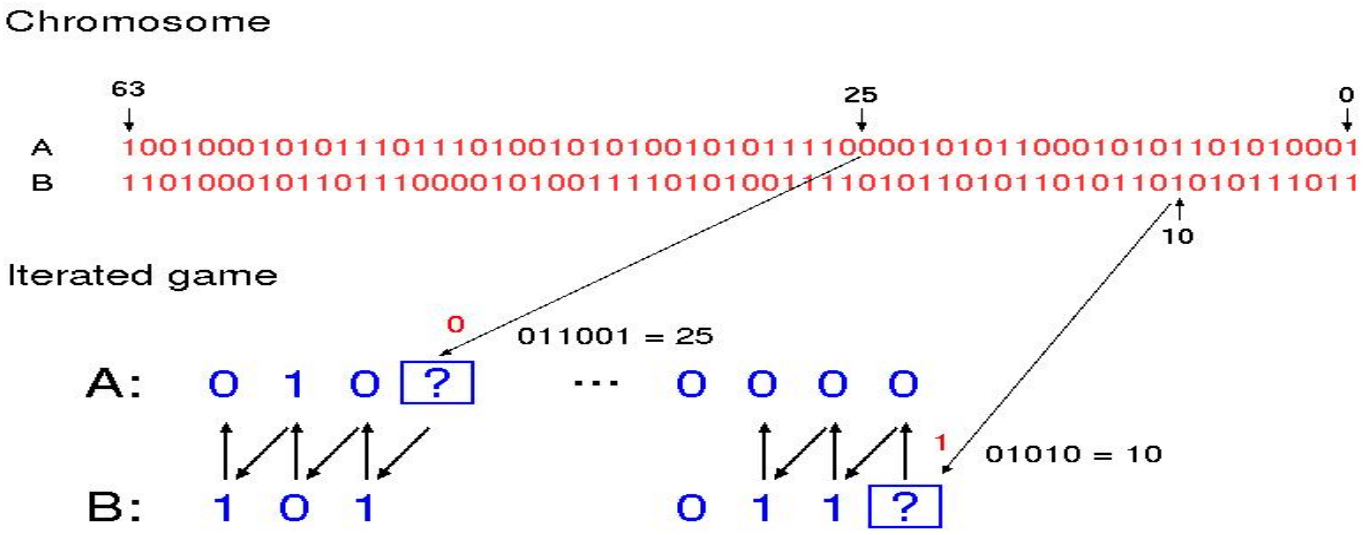
A two player's game like Paper Stone Scissors

An iterated game

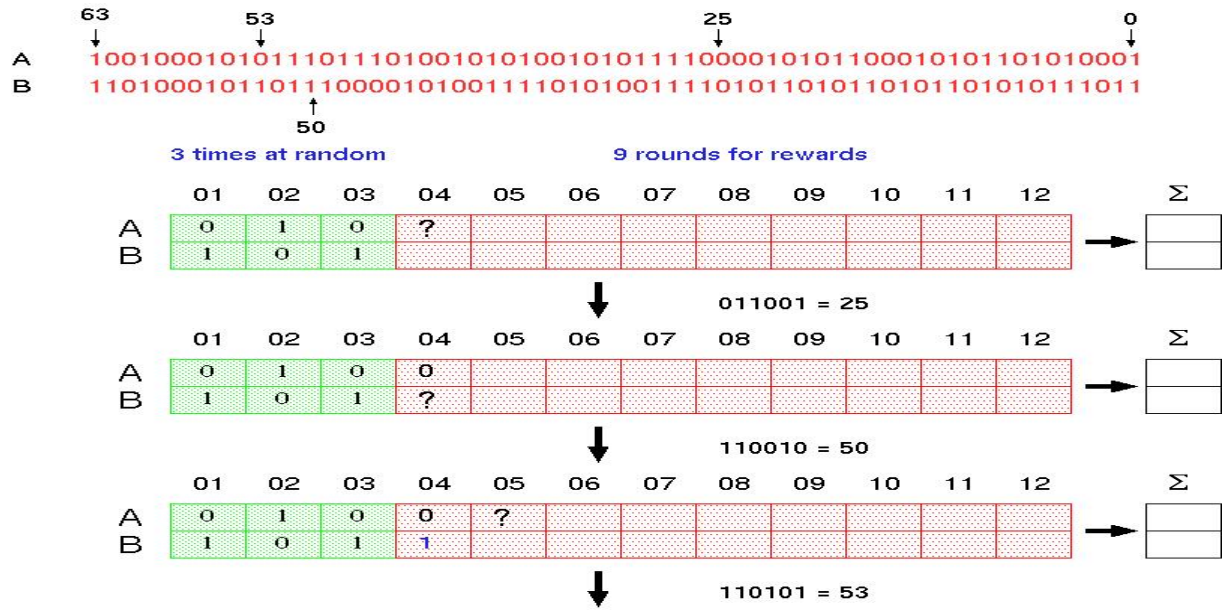


Try: Random vs Random and Best vs Random, Always-1, Always-0, Tit-for-tat

Chromosome as a game strategy



Next action dependent on chromosome



XIII. Dimension Reduction by Evolution

Samon Mapping

3 points in 3D plus Origin

A:	(0.328, 0.819, 0.118)
B:	(0.129, -0.256, 0.713)
C:	(0.277, -0.584, -0.354)
O:	(0.000, 0.000, 0.000)

Distance Matrix

	A	B	C	O
A	0.000	1.245	1.481	0.890
B		0.000	1.126	0.769
C			0.000	0.737
O				0.000

Normalized

	A	B	C	O
A	0.000	0.846	1.000	0.601
B		0.000	0.760	0.519
C			0.000	0.498
O				0.000

random 5 points in 2D plus Origin

X:	(0.514, -0.223)
Y:	(-0.861, 0.979)
Z:	(-0.113, -0.142)
O:	(0.000, 0.000)

	X	Y	Z	O
X	0.000	1.826	0.632	0.560
Y		0.000	1.348	1.304
Z			0.000	0.182
O				0.000

	X	Y	Z	O
X	0.000	1.000	0.346	0.307
Y		0.000	0.738	0.714
Z			0.000	0.273
O				0.000

chromosome:

(0.514 -0.223 -0.861 0.979 -0.113 -0.142 0.000 0.000)

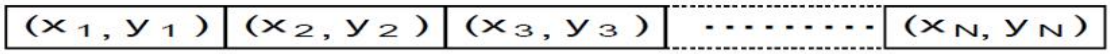
fitness

(0.846-1.000)+(1.000-0.346)+(0.601-0.307)+(0.760-0.738)+(0.519-0.714)+(0.498-0.273)

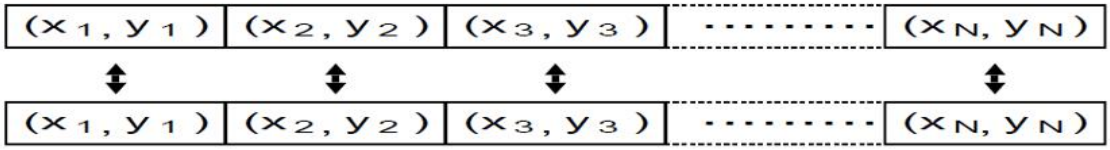
= 0.846

Chromosome for Samon Mapping

Chromosome:



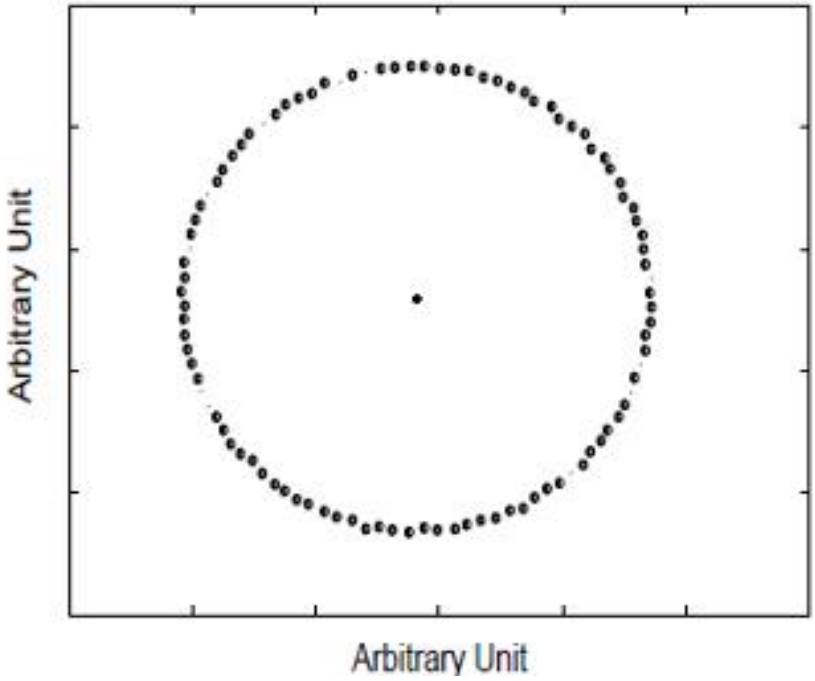
Recombination with Uniform Crossover:



Algorithm

1. *Assume N points are given in the n-D space.*
2. *Calculate distance matrix R ($N \times N$) whose i-j element is the Euclidean distance between the i-th and j-th point.*
3. *Also think of a tentative N points in the 2-D space that are located at random at the beginning.*
4. *The distance matrix Q is calculated in the same way as R.*
5. *Then the error matrix $P = R - Q$ is defined.*
6. *Search for the locations of N points in the 2-D space that minimizes the sum of element P.*

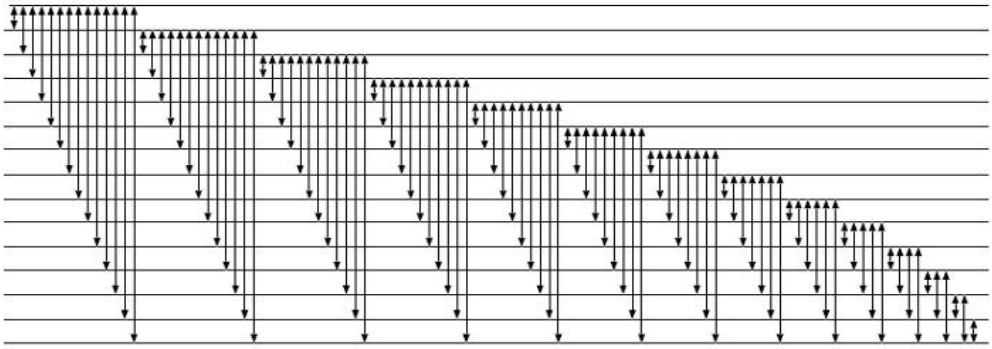
Example of mapped circle from surface of a hypersphere



XIV. Sorting Network

1. What is a sorting network?

E.g. Bubble sort - 16 items



The number of comparison of the items in this case is:

$$15 + 14 + 13 + \cdots + 2 + 1 = 120$$

Then what would be a minimal number of comparison?

⋆ 65 comparisons Bose and Nelson (1962).

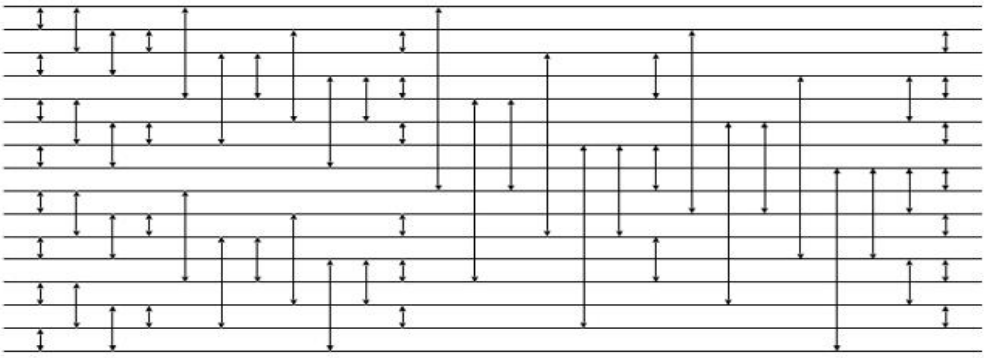
⋆ 63 by Batcher and by Floyd and Knuth (1964).

⋆ 62 by Shapiro (1969)

⋆ 60 by Green (1969)

Ex. Sorting algorithm by Knuth et. al (1964)

63 Comparisons



2. A possible sorting algorithm by evolution

E.g. An integer chromosome to sort 16 items
with 140 genes each takes an integer from 1 to 16 permitting overlaps

(12 01 05 04 16 12 04 14 01 02 06 ..... 07 15 08 10)

Which compares

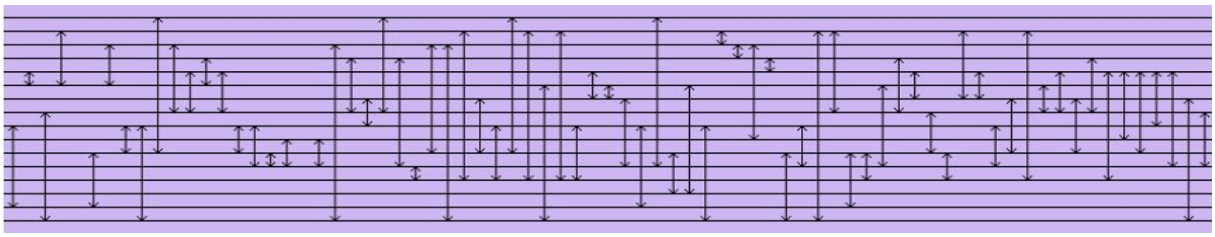
12 <=> 01
05 <=> 04
16 <=> 12
.....
08 <=> 10

A simple implementation

(1) Create one chromosome
to see how it will sort 16 integers from 1 to 16 with a random order
Also draw its graphic representation

E.g.

chromosome: [08 14 04 05 15 07 05 01 07 07 10 14 02 05 08 10 08 15 10 00 02 07 07 04 03 05 04 07 08 10 08 11
11 10 11 09 03 03 09 11 15 02 07 03 06 08 07 00 11 03 12 11 02 10 02 15 12 01 06 10 08 12 00 10 01 12 15 05 01
12 12 08 06 04 06 05 06 11 14 08 00 11 13 10 05 13 08 15 01 02 03 02 02 09 03 04 15 10 08 11 01 15 01 07 10 14
10 12 05 11 03 07 04 06 10 07 10 12 01 06 06 04 11 08 06 10 12 01 07 05 04 07 10 06 03 07 12 04 04 09 10 04 04
08 11 04 15 06 07 11]

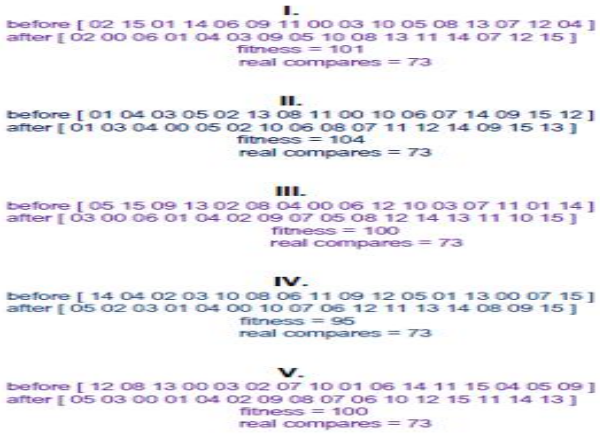


(From the student work by Supruniuk Darya in 2015)

(continued)

(2) Apply your chromosome to integer from 1 to 16 with a random order

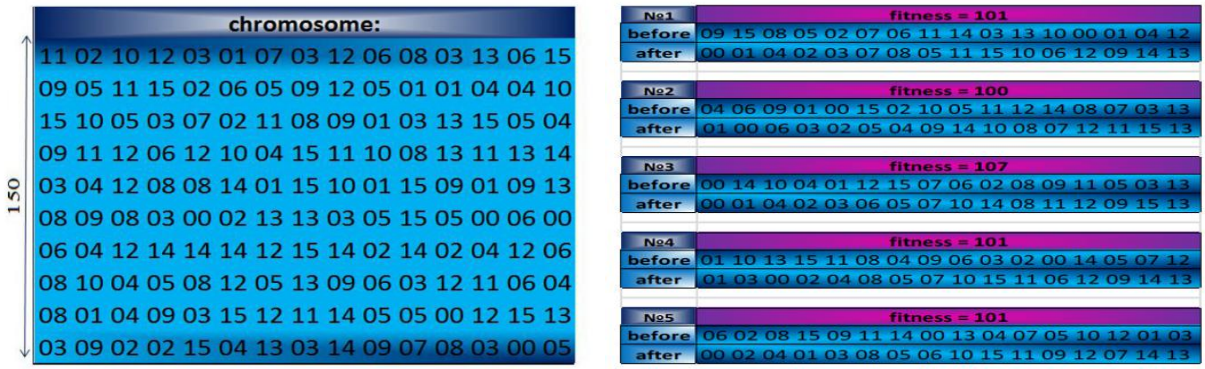
such as



(From the student work by Supruniuk Darya, too, in 2015)

(continued)

Or, see another example



(From the student work by Radchuk Aliona in 2015)

(continued)

(3) Fitness is evaluated as

$fitness = 0$
FOR i=1 TO n
FOR j=1 TO n
IF $x(i) > x(j)$ THEN replace $x(i)$ with $x(j)$ and $fitness{+}{+}$
ELSE do nothing

For example, assuming $n = 5$

$$01\ 03\ 02\ 05\ 04 \Rightarrow \text{fitness} = 4 + 2 + 0 + 0 = 6$$
$$04\ 02\ 05\ 01\ 03 \Rightarrow \text{fitness} = 1 + 2 + 0 + 1 = 4$$
$$05\ 01\ 03\ 02\ 04 \Rightarrow \text{fitness} = 0 + 3 + 1 + 1 = 5$$
$$01\ 02\ 03\ 05\ 04 \Rightarrow \text{fitness} = 4 + 3 + 2 + 0 = 9$$

(continued)

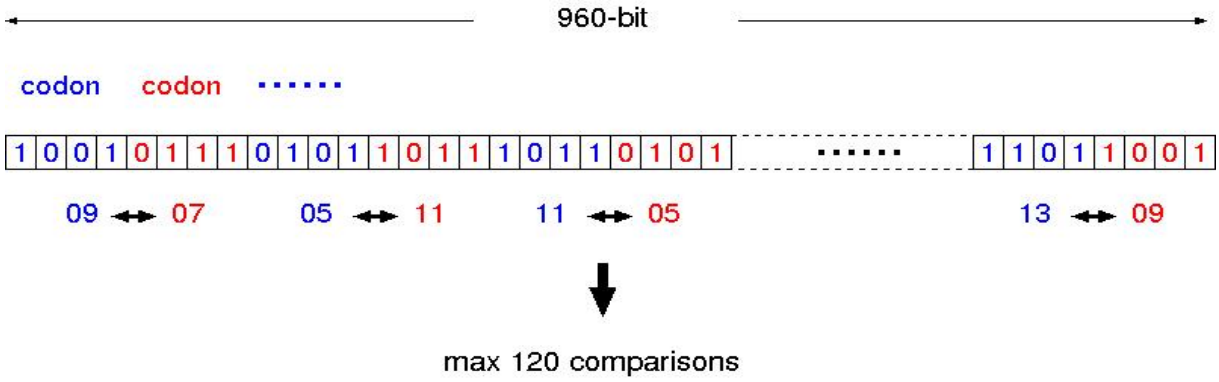
An algorithm

1. Create N - a set of random order integers from 1 to 16.
2. Create a population of, say, 20 chromosomes at random.
3. Evaluate fitness of each chromosome by applying it to N.
4. Create the next generation by selection, crossover and mutation.
5. Repeat from 3. to 4. untill maximum fitness saturates.

Show (i) the best chromosome in the final generation; (ii) The result of applying it to N with the original order (before & after); and (iii) its diagram by ommiting those identical comparisons.

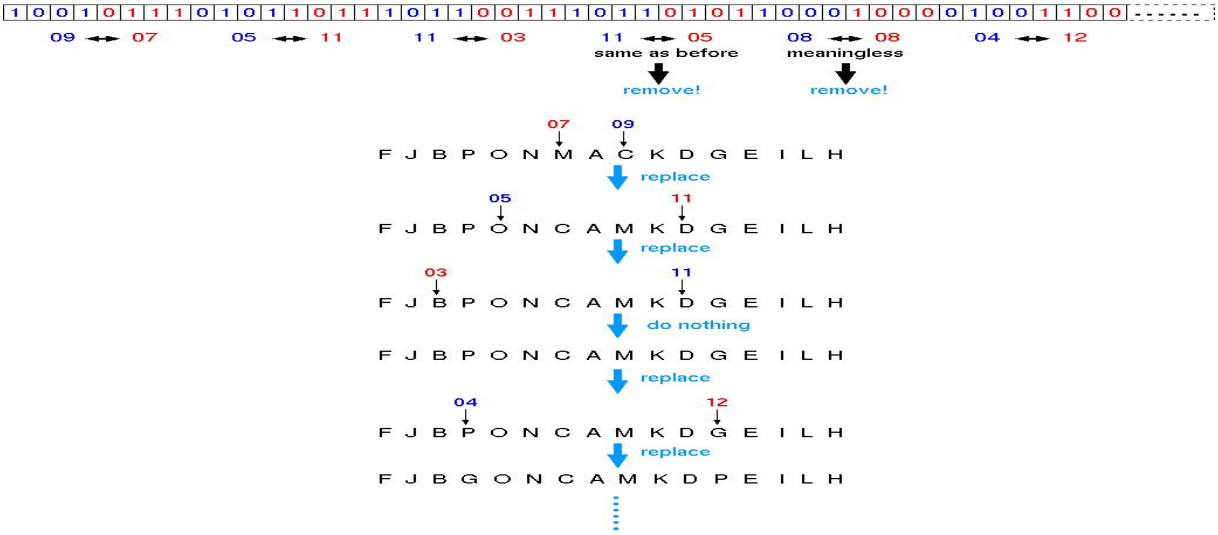
3. Let's use binary chromosomes

Binary chromosome made up codon works as



(continued)

How this chromosome sorts 16 letters to alphabetical order?



4. A more biological diploydy chromosomes

Hillis' Diploydy chromosome (1992)

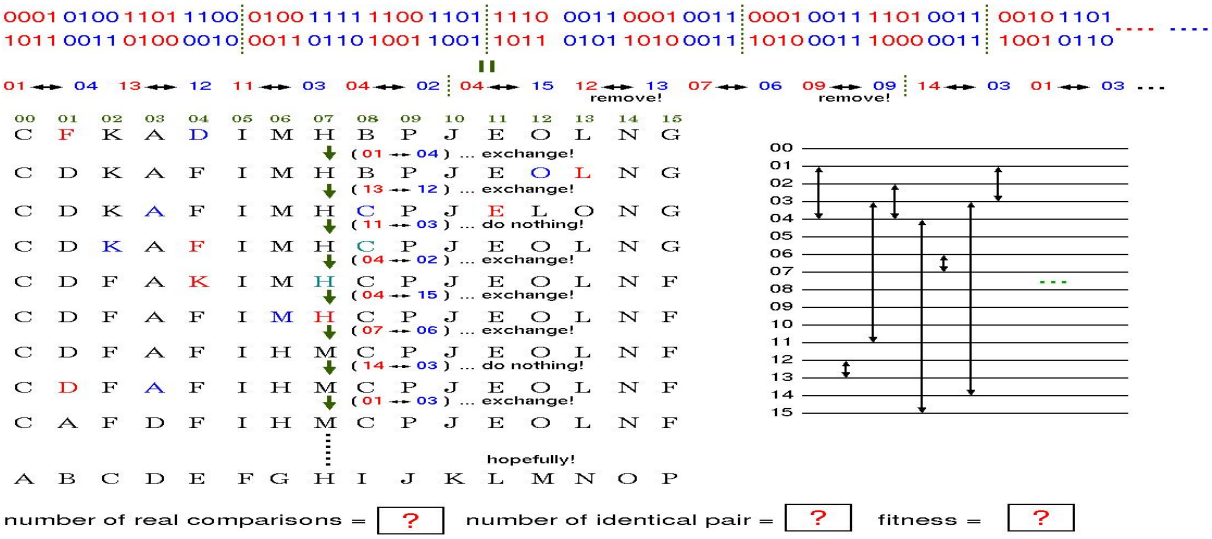
(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)
(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)
(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)
(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)
(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)	(1001 1000 1010 1101 1110 0100 1110 0011) (1001 1000 1010 1101 1110 0100 1110 0011)

15 pairs of 32-bit chromosomes
Each chromosome consists of eight 4-bit strings called codons
Each codon is an integer between 0 and 15
$\Downarrow$
60 comparisons: world record, so far!

(no proof this is minimum, though)

(continued)

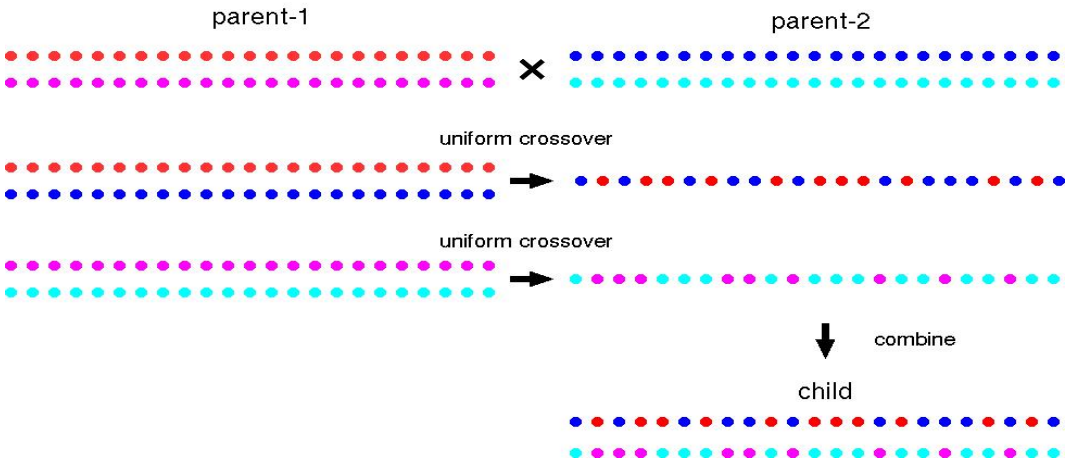
How a pair of chromosomes each with 480-bit sorts 16 items?



number of real comparisons = ? number of identical pair = ? fitness = ?

(continued)

Diploydy chromosome & its crossover



(continued)

Excercise - Evolution of diploydy chromosomes

(1) Create your target random order string of 16 alphabets from A to P
(2) Observe evolution from a poulation of randome 40 diploydy chromosomes untill maximum fitness does not change

(continued)

What should be shown are:

(1) Original string with 16 alphabets in random order;
(2) The result of string after sort by the final best chromosome;
(3) The best chromosome in the final generation;
(4) The diagram of comparisons of the best chromosome;
after removing multiple comparisons and self-comparisons;
(5) (i) What is its fitness value? And (ii) how many real comparisons it includes?

(6) Three graphs during evolution

The best chromosome can sort other random string?

Can the best chromosome, whose fitness was how the original one target string of 16 letters, will be able to sort another new random string correctly?

(continued)

Excercise - Evolution of diploydy chromosomes to sort multiple strings

(1) Create two target random order 16 alphabets from A to P.
· Fitness this time is the sum of how it sorts these two strings.
(2) Observe evolution from a poulation of randome 40 diploydy chromosomes untill maximum fitness does not change.
(3) Apply the best chromosome to 10 random strings including the original two!

⇓

Show a set of 10 strings with its BEFORE & AFTER!

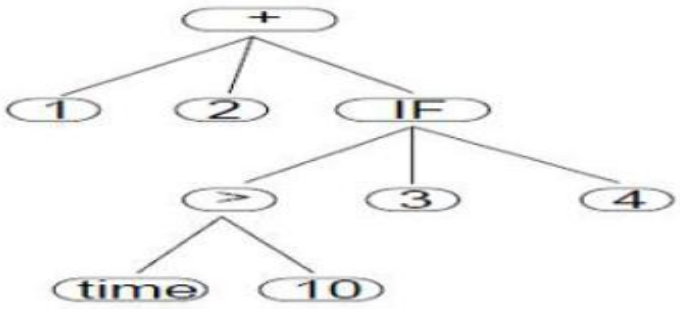
XV. Genetic Programming

Programming by evolution

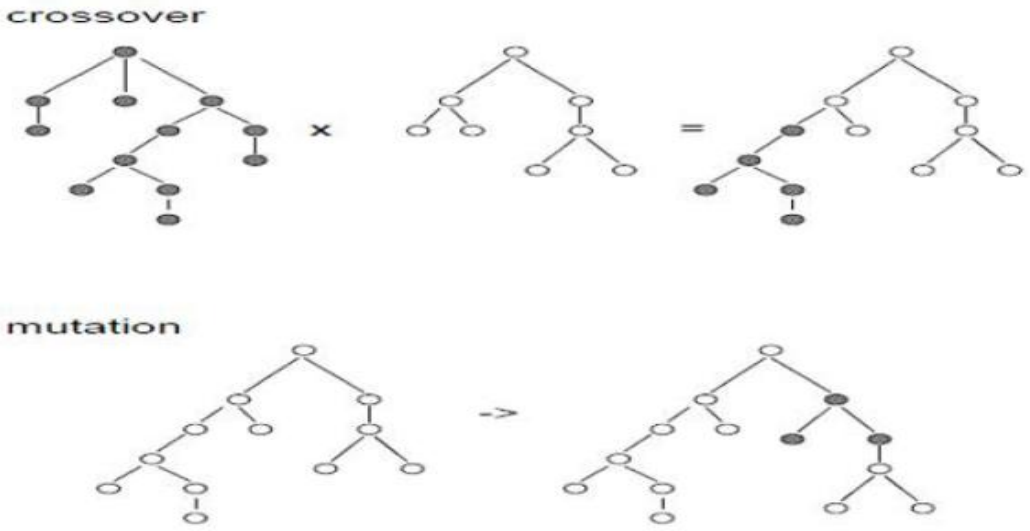
Prgramming in LISP which can be represented by tree

such as

```
(+ 1 2 (IF (> time 10) 3 4))
```



Its crossover & mutation



XVI. To evolve structure to grow

XVII. Boldwin Effect

A needle in a heystack problem

Assume now a neural network with 20 connections whose weight is either 1 or 0. Also assume only one configuration of 20 connections out of 2^{20} possibilities is correct.

How difficulty of search increases as connections increase?

Hinton and Nowlan's Simulation of Life Time Learning

Now let's represent each neural network by a chromosome whose gene is either "1" or "0" or "?". All genes that is "?" of each chromosome are replaced with either "1" or "0" at random from one gene to the next. Each connection that is given "?" knows what it should be. If connection feels the replacement is good, then it will be fixed and no changed thereafter, otherwise maintain to be "?", which is called "learning." Each learns a maximum of 1000 times during its lifetime. N = the number of repetion till all "?" genes are fixed , that is, minimum 0 when it learns at once and maximum 1000 when it never learned) 1000 matings happens each of which creates one child with one-point crossover. Provability to be selected is proportionate to 1 + 19N/1000, which means the more quick chromosomes learn the more likely to be selected.

An algorithm

1. Create a population of 1000 chromosomes with 20 genes either of 1, 0, ? with probability being 0.25, 0.25, 0.5, respectively.
2. Make each chromosome a maximum of 1000 learnings by assigning "1" or "0" to each of "?"
   * each gene feels happy or unhappy and if happy the gene is fixed and otherwise remain "?"
   * if all "?" are happy tnen $N = 1000 - so.far.trials$ else keep learning till $N = 0$.
3. Select two parents at random with the probability of $1 - 19N/1000$.
4. Create next generation by repeating 3. 1000 times
5. Repeat from 2. to 4. untill solution found