An Introduction to Evolutionary Computations

(14 April 2003 — one day in winter 2004)

Akira Imada

Brest State Technical University
e-mail: akira@bstu.by

last modified on:

November 4, 2003

Bibliography

The followings are the most popular, frequently cited and hopefully still available books in the field of Evolutionary Computations (ECs).

(1) Goldberg D.E. (1989) "Genetic Algorithm in Search, Optimization and Machine Learning", Addison-Wesley.

· Though it covers only Genetic Algorithm (GA), we might say it was this book that made Evolutionary Computations so popular nowadays.

(2) Fogel D.B. (1995) "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence", IEEE Press.

· This book covers all the main Evolutionary Computations, that is, Genetic Algorithm, Evolution Strategy (ES) and Evolutionary Programming (EP) but emphasis is on EP.

(3) Back T (1996) "Evolutionary Algorithms in Theory and Practice", Oxford University Press.

· This also covers GA, ES, and EP but emphasis is on ES. Description is a little more mathematical than the previous two.

(4) Michalevwicz Z. (1994) "Genetic Algorithms + Data Structures = Evolutionary Programs", Springer.

· This again covers only GA, but from a different aspect than the first one. Also this used to be a very popular book among the community.

(5) Koza J.R. "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press.

· This book explains only Genetic Programming (GP), that is evolution of tree structures, while previous four treat binary or real number of string structures. This book includes a lot of examples of its application as well as source codes in LISP language.

□ Contents:

The 1st day:

Monday, 14 April 2003
(13:20 – 14:40)

Today's Keywords:

chromosome, gene, selection, fitness, recombination, crossover, mutation, population, generation

□ What are Evolutionary Computations?

— Then what for, and how?

(What?)

- Algorithms inspired by Biological Evolution, i.e.,

  'Survival of the Fittest.'

(What for?)

- To obtain (Near-optimum) solution(s) to
  - · not-solvable-analytically

    and/or
  - · very difficult problems

(But how?)

- All what we need to design are?
  - · How we represent the problem by chromosomes?
  - · How do we evaluate fitness?

□ How we evolve candidate solutions?

— From stupid solutions to an excellent one!

Design the form of *chromosome*[1] *(genotypes/individuals)* so that it respresents a candidate solution (*phenotype*).

⇓

Create a *population* of random chromosomes to construct the 1st *generation*.

⇓

Evaluate *fitness* (how good they are?) of each of these individuals.

⇓

Select two chromosomes according to the fitness such that "the higher the fitness the more likely to be selected."

⇓

Recombine the two chromosomes to produce one child chromosome by *crossover* and *mutation*.

⇓

[1] The name is in the sense that our chromosome give instruction or is interpreted on how to build a feasible phenotype like biological chromosome when phenotype is created.

(cont'd)

⇓

Create the next generation by repeating selection & recombination.

⇓

Thus, we expect better individuals from generation to generation.

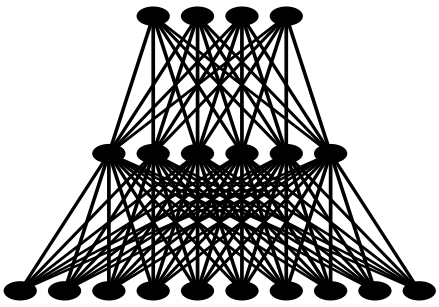
⇓

The cycle of reconstructing the new population with better fitness individuals and restarting the search is repeated until one of the global optima is found or a set maximum number of generation has been reached.

□ A Simple Example

— Evolution of Weights of Feed-forward Neural Networks.

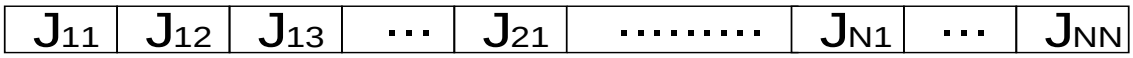
- NN as a black box.
  - ⋆ E.g. for Pattern Recognition.



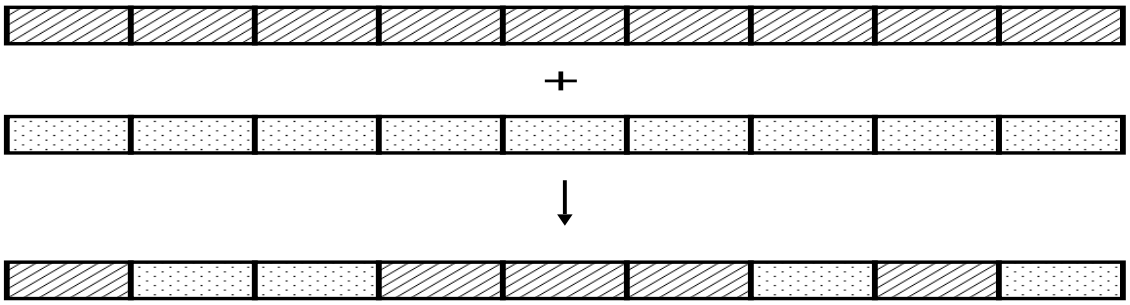
- But how we determine its weights?
  - ⋆ A proposal is by using ECs.

□ J_{ij} by Genetic Algorithm?

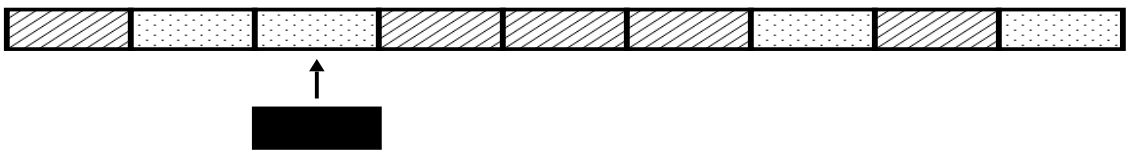
1. Represent a series of J_{ij} as a *population* of strings (*chromosome*).



2. Define a fitness evaluation.
   (How good is each individual?)
3. Generate an initial *population* at random.
4. Evolve them with
   - *Selection*
   - *Crossover*



   - *Mutation* [2]



5. Better Solutions from *generation* to *generation.*

[2] A commonly used mutation rate is one over the length of chromosome.

The 2nd day:

Monday, 21 April 2003
(13:20 – 14:40)

Today's Keywords:

AND, OR, XOR, neuron, synapse, weight, transfer function, threshold, neural networks, input/hidden/output layer

□ Probably a Simplest Examples.

- Output Y of the neuron which receives weighted-sum of the signals X_i from other N neurons is:

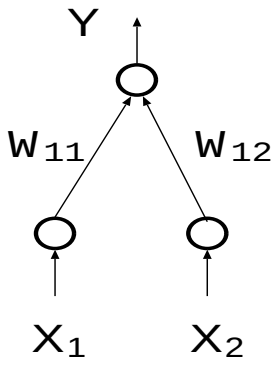
$$Y = \text{sgn}(\sum_{i=1}^{N} w_i X_i - \theta)$$

where $\text{sgn}(x) = 1$ if $x \geq 0$ and 0 otherwise, and w_i and θ are called *weight* and *threshold*, respectively.

- NN to solve AND & OR.

AND

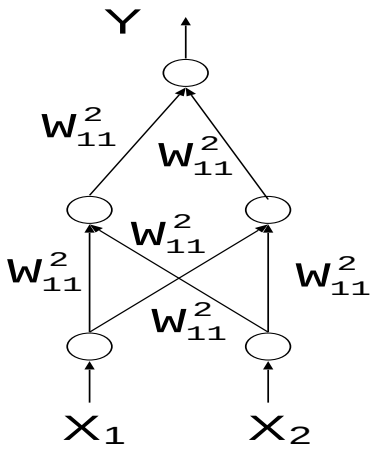
X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1



· Well, how about NN to solve XOR?

XOR

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



Exercise 1 *Obtain six weights values so that the above NN function as XOR.*

Exercise 2 *Create Pseudo code for EC to obtain the six weights above.*

The 3rd day:

Monday, 28 April 2003
(13:20 – 14:40)

Today's Keywords:

Trancate/Roulette-Wheel(Fitness-proportionate)/Tornament Selection
one-point/two-point/multipoint Crossover, uniform-crossover

The 4th day:

Monday, 19 May 2003
(13:20 – 14:40)

Today's Keywords:

Combinatorial Optimization Problem
Polynomial time, Combinatorial explosion, NP-complete,
Knap-sack-problem

□ Combinatorial Optimization Problem.

When the problem is of size N [3]

$\Downarrow$

the number of possible candidate solutions is typically [4]

$N!$, N^N, e^N, $\cdots$ etc

(called *combinatorial explosion*)
(Imagine how huge is the solution space for a large N.)
(See also the graph of next page.)

$\Downarrow$

If we want the one that minimizes the cost function

$\Downarrow$

we call these problems

Combinatorial Optimization Problems.

[3] In Traveling Salesperson Problem (TSP) the size is the number of city to which a salesperson must visit in a tour.
[4] In TSP the number of possible candidate solutions is $(N-1)!$.

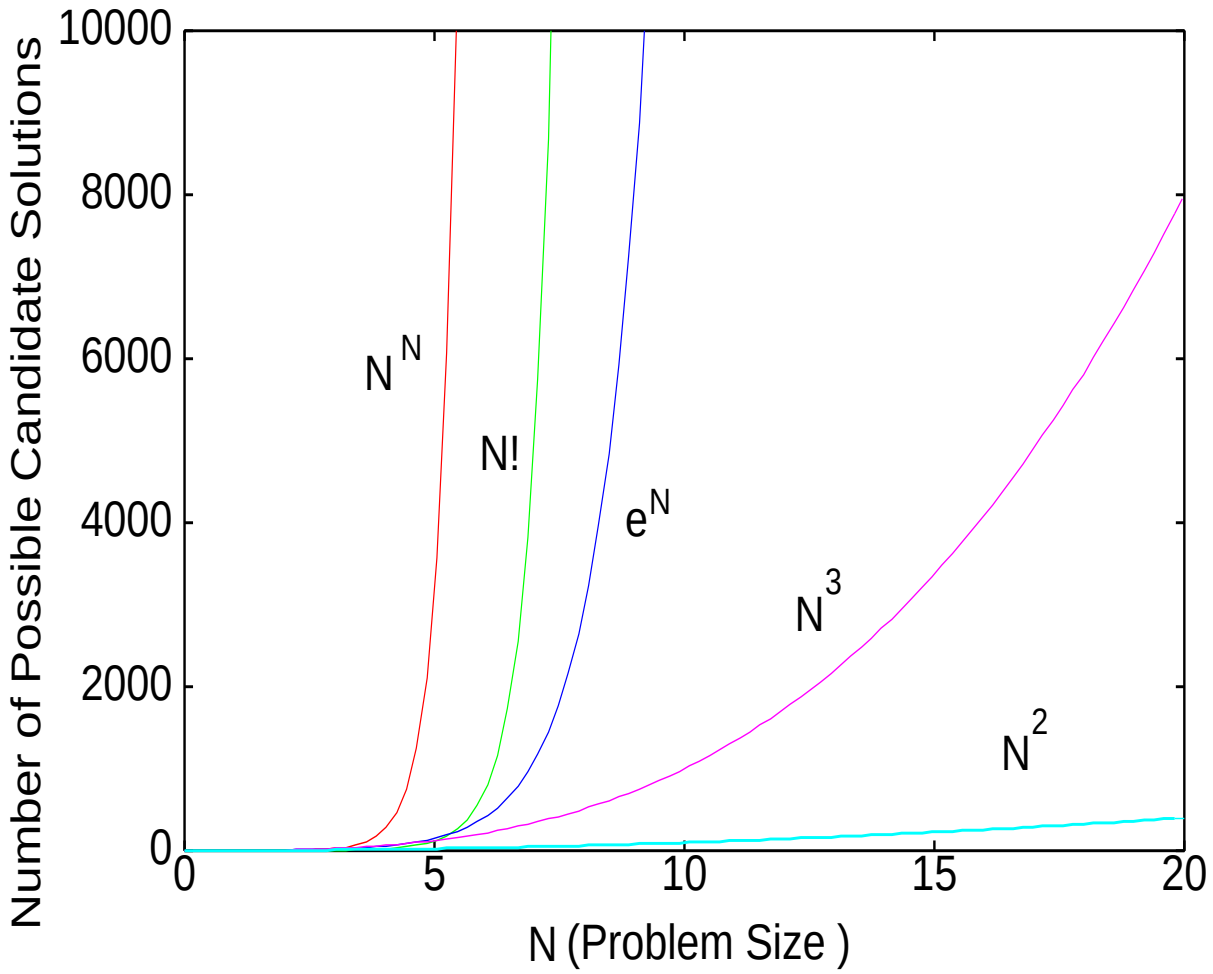
□ NP-Complete.

If there exists an algoithm that solves the problem in a time that grows only polynomially (or slower) w.r.t. N

$$\Downarrow$$

then it is said to be polynomial class: P.

- See examples of how the number grows w.r.t. N.



(cont'd)

If one can verify in polynomial time
whether any guess of the solution is right or not

$\Downarrow$

we call it non-deterministic polynomial class: *NP*.

If it's impossible to verify in polynomial time

$\Downarrow$

we call it *NP-hard*.

Then

$\Downarrow$

$$NP + NP\text{-}hard = NP\text{-}complete$$ [5]

[5] If one could find a deterministic algorithm that solves one NP-complete problem in polynomial time then all other NP problems could be solved in polynomial time.

(cont'd)

- Examples:

  ⋆ 0-1 Knapsack Problem

  · When n items whose i-th item has weight w_i and profit p_i, search for a binary vector

  $$\mathbf{x} = (x_1, x_2, \cdots, x_n)$$

  that maximizes

  $$\sum_{i=1}^{n} x_i p_i.$$

  such that

  $$\sum_{i=1}^{n} x_i w_i \leq C$$

  where C is capacity of the knapsack.

  ⋆ Assume $\mathbf{x}$ be chromosome of our EC.

  ⋆ We sort all items in decreasing order of p_i/w_i then we take i-th item iff i-th gene of chromosome $\mathbf{x}$ is 1 until knapsack is full or there are no items left.

The 5th day:

Monday, 26 May 2003
(13:20 – 14:40)

Today's Keywords:

Traveling Salesperson Problem (TSP)

(cont'd)

⋆ Traveling Salesman Problem

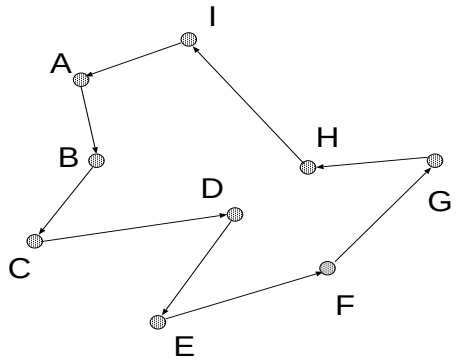
· A salesman must visit a number of cities, and then return home. In which order should the cities be visited to minimize the distance traveled?

· If we represent a candidate solution with a list of cities to be visited in order, the results of crossover and mutation are feasible?

· In order for the result of crossover and mutation to be feasible what representations are possible?

(cont'd)

An example:



tour:
A-B-C-D-E-F-G-H-I-A

Chromosome is to be designed so that it allows us to form a tour by following the procedure:

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 $i \leq n$.

For example, when the list of cities is

$$\{\text{A, B, C, D, E, F, G, H, I }\}$$

chromosome: (112141311) is the tour:

A-B-D-C-H-E-I-F-G.

Exercise 3 *Try some one-point crossover on two parents* (112141311) *and* (515553321)

(cont'd)

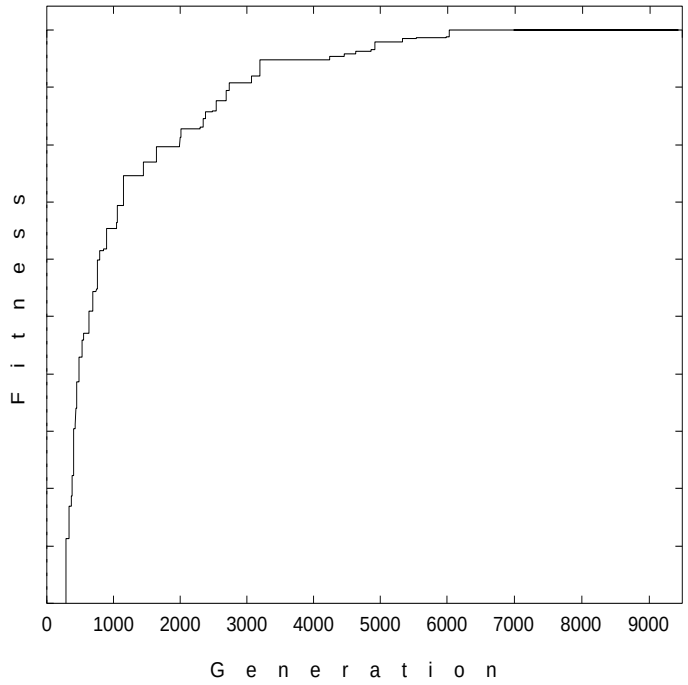
⋆ Then how we design mutation?

· How about specifying two points at random and reverse the gene order?

⋆ When to stop a run?

· If we don't know the optimum, then observe the evolution of fitness.

- We can expect the run converges to the optimum or near-optimum.
- See an example bellow.



Let's open the course again!

The 1st day of this fall semester
(6th day since the last spring semester):

Tuesday, 03 November 2003
(10:00 – 11:20)

Today's Keywords:

Commonly used testfunctions, Sphere Model, Ackley's Function

□ Let's try EC on a testfunction.

- Commonly Used Test Functions
  — to learn "How an EC works?"

Sphere Model F_1:

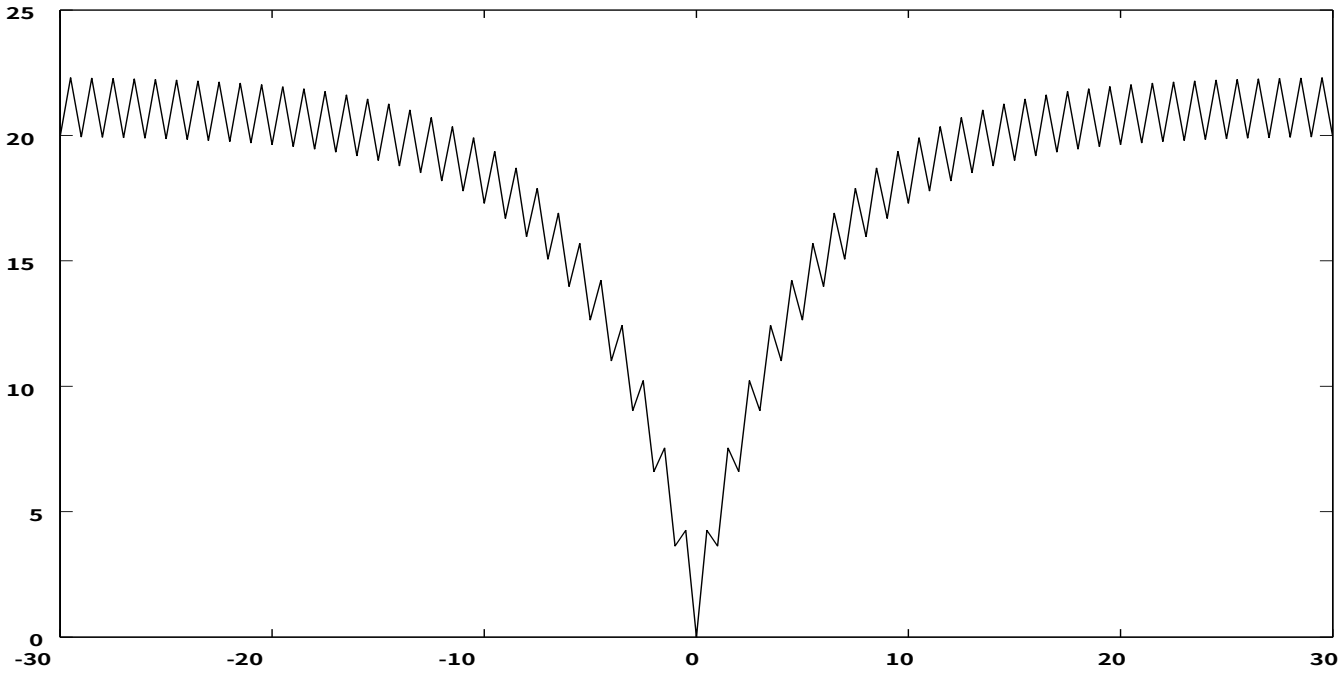
$$y = \sum_{i=1}^{n} x_i^2, \quad x_i \in [-5.12, 5.12].$$

Ackley's Function F_9:

$$y = -20 \sum_{i=1}^{n} \exp\left(-0.2\sqrt{x_i^2/n}\right) - \exp\left(\left(\sum_{i=1}^{n} \cos 2\pi x_i\right)/n\right) + 20,$$
$$x_i \in [-30, 30].$$

· A two dimensional example:

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



(cont'd)

- Other Test Functions

Rastrigin's Function F_6**:**

$$y = nA + \sum_{i=1}^{n} (x_i^2 - A\cos(2\pi x_i)), \quad x_i \in [-5.12 - 5.12].$$

Schwefel's Function F_7**:**

$$y = \sum_{i=1}^{n} (x_i \sin(|x_i|)), \quad x_i \in [-500, 500].$$

Griewangk's Function F_8**:**

$$y = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1, \quad x_i \in [-600, 600].$$

Excersise 1 *Choose one out of four functions above, and explore the function. That is, draw surface of its 3-D version, check the location of the global minimum, learn distribution of local minima, etc...*

Excersise 2 *Apply Evolutionary Computation to locate the global minimum. Try various parameters and/or selection and crossover system, for example, try with* $N = 100, p_m = 0.05$*, tornament selection and one-point crossover, etc.*