An Introduction to Evolutionary Computation

(September 2004 —)

Akira Imada

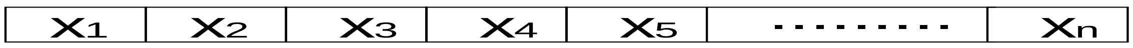
(last modified on)

November 18, 2004

1 What does GA look like?

We now assume to solve a problem which includes n variables. That is, our task is to determine an optimal set of n variables. Then we design GA as:

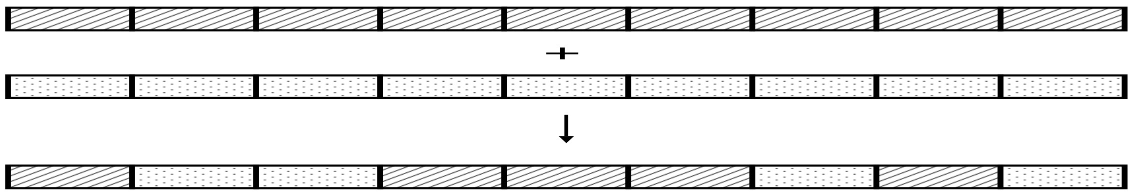
- Represent a series of x_i as a *population* of strings. Each of these strings is refered to as *chromosome* or sometimes called *individual.*



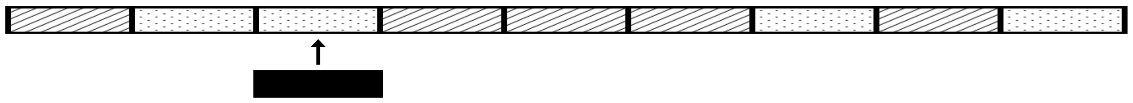
- Define a *fitness* evaluation by "How good is each individual?"

Then we start an evolution as follows, expecting better solutions from generation to generation.

1. **(Initialization)** Generate an initial *population* of p individuals at random.
2. **(Fitness Evaluation)** Evaluate fitness of each chromosome and sort the chromosomes according to its fitness from the best to the worst.
3. **(Selection)** Select two chromosomes
   - Here, from the best half of the population at random, which is called a *Truncate Selection.*
4. **(Reproduction)** Produce a child by the following two operations:
   - *Uniform Crossover*, for example



   - *Mutation* [1]



5. Create the next generation by repeating the steps from 3 to 4 n times.
6. Repeat the steps from 2 to 5 until (near) optimal solution is obtained.

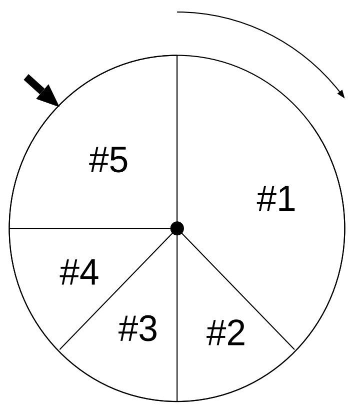
[1] A commonly used mutation rate is one over the length of chromosome.

2 How we select parents?

Hopefully, *the better the fitness the more likely to be selected.*

- Threee different versions of Selection.
  - ⋆ Truncation Selection (Simplest of the three):
    - · Select parents from the best some-percentage of the population.
  - ⋆ Roulette Wheel Selection (or Fitness Proportionate Selection):
    - · Select such that the probability to be selected is proportional to the fitness value.
      For example,

	fitness
individual #1	0.375
individual #2	0.125
individual #3	0.125
individual #4	0.125
individual #5	0.250

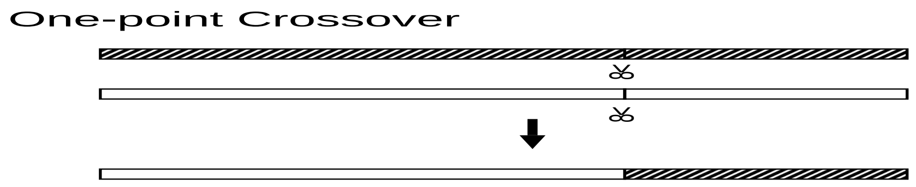


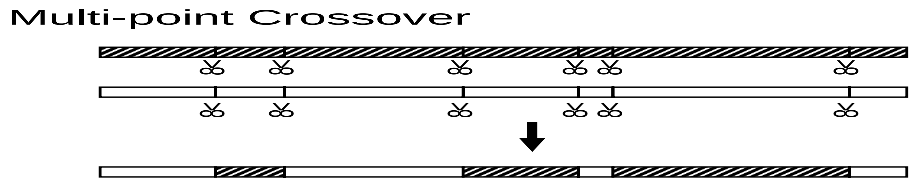
  - ⋆ Tournament Selection
    - · Assume we have the original μ parents and their μ children. The fitness value of each of the 2μ individuals are compared to those of q individuals which are chosen randomly from the whole 2μ points at every time of the comparison Then the 2μ individuals are ranked according to the number of wins, and the best μ individuals survive (q-tournament selection).

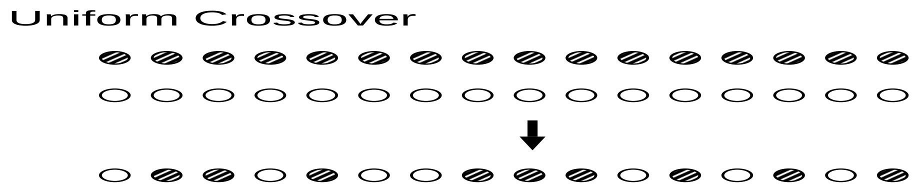
Note that even the worst fitness individual have a chance to be selected under Roulette Selection however few it might be, while under Trancate Selection worse individual have no chance to be selected. Tornament Selection could also select a worse individual except for the worst q individuals. We can control the probability of selecting worse individual by changing q.

3 How parents produce a child?

So-called *crossover* is exproited for the reproduce children. Here, we see three different versions of crossover bellow.







4 Let's challange an already known solution by a GA

Bellow we study two multi-dimensional test-functions whose location of the global minimum was already known. It is, therefore, useful to explore this test-function to learn how evolution works inside the PC.

4.1 Sphere model

Probably the simplest one is

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

which is difined, for example, on each $x_i \in [-5.12, 5.12] \quad (i = 1, 2, \cdots, n)$. This function is an extention of well known $y = x^2$ to its n-dimensional version. The unique global minimum is located on the Origin and no local minimum. Hence, this might be a good start to try a GA.

4.2 Shcwefel function

The function bellow is called Schwefel function and enormous amount of local minimum and the only unique global minimum at the Origin. Search for the global minimum when the function is difined on, say, $x_i \in [-500, 500]$.

$$y = \sum_{i=1}^{n} (x_i \sin(|x_i|)) \tag{2}$$

You might try to explore this hyper-surface by setting n to 20, for example. Then the surface is defiened on 20-dimensional Eucledean space. If, however, you want to know the image of the hyper-surface, see a two dimensional version of this function. The graph of

$$y = x \sin(|x|) \tag{3}$$

is shown in Figure 1.

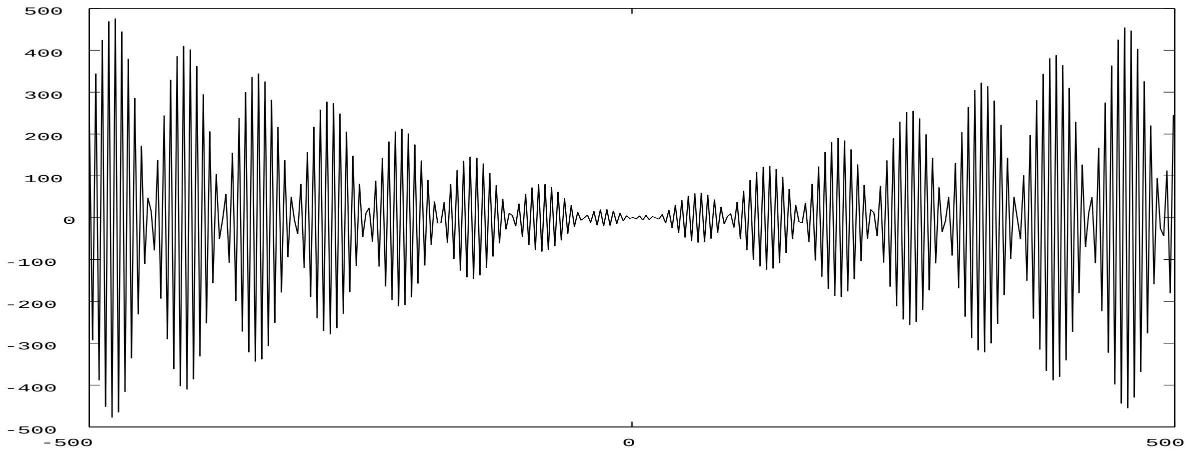


Figure 1: A two-dimensional version of the Schwefel's Function's graph.

4.3 Yet another Testfunction — A 2-D function but multi peaks

The test-function of the previous subsection is the one which evolutionary computations are especially good at, since we can treat whatever large dimension simply by setting the number of genes in a chromosome to the dimensionality.

Then what should we do, if we are interested in a 2-D function? For example,

$$y = \sin^6(5\pi x) \tag{4}$$

or

$$y = -2((x - 0.2)/0.8)^2 \sin^6(5\pi x) \tag{5}$$

are interesting functions in order for us to observe how randomly created chromosomes in the 1st generation evolve to find peaks. See Figure 2.

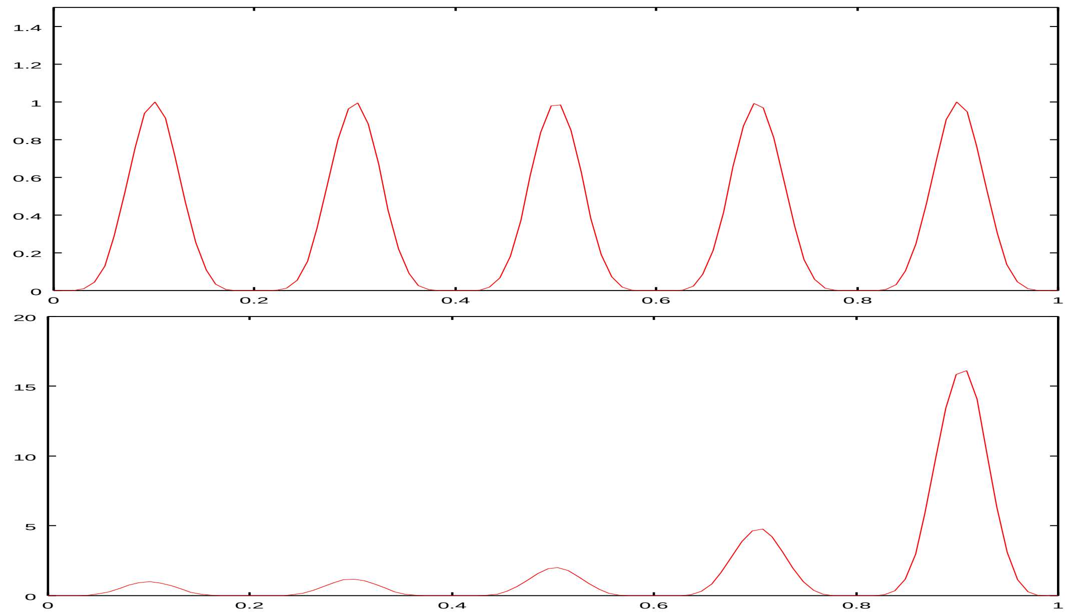


Figure 2: A multi-peak 2-D function and its variation

The question is how we design our chromosomes. In the multi-dimensional function $y = f(x_1, x_2, \cdots, x_n)$ our genes might be continuos value each of which corresponds to the independent variable x_i $(i = 1, 2, \cdots n)$, that is, our chromosomes are made up of n genes. On the other hand how should we design our chromosomes when the number of independent variable is only one. A chromosome with only one gene? How we crossover two parents?

O.K. we usually use binary chromosome in this situation. Any (decimal) real-valued variable $x_i \in [a, b]$ could be encoded by n-bit binary strings where a and b is represented by $(00 \cdots 0)$ and $(11 \cdots 1)$, respectivley, and therfore accuracy (or granularity) is $(b - a)/(2^n - 1)$. For example, if our concern is the above $x \in [0, 1]$ then 10 bit of binary strings from 0000000000 to 1111111111 are expresses decimals with the precision of 1/1024. Or you might use and comapre *Gray Code* where gray-code $a_1 a_2 \cdots a_n$ is tlanslated from binary number $b_1 b_2 \cdots b_n$ as

$$a_i = \begin{cases} b_i & \text{if} \quad i = 1 \\ b_{i-1} \oplus p_i & \text{otherwise} \end{cases} \tag{6}$$

where $\oplus$ is addition modulo 2. In gray code a pair of adjusent decimals are different only with Hamming distance 1, while in the standard binary encode this does not hold.

5 Jeep Problem — Also we know the solution, but...

Assume we have a Jeep at the base which locates at the edge of a desert. The Jeep has a tank which can be filled with a maximum of one-unit of gasoline. With one unit of gasoline, the jeep can move one unit of distance. The Jeep can only fill gasoline at the base.The jeep can carry containers to put its gasoline on the desert for a future use, Assume the tour should be on the strait line in the desert.

The question is "How far the jeep can penetrate in to the desert on the straight road when n unit of gasoline is available at the base.

For example when $n = 2$, the best strategy is to start the base with one unit of gasoline in its tank and go 1/3 unit distance (it has spent 1/3 unit of gasoline to reach the point, then put 1/3 unit of gasoline in the container there (now jeep has 1/3 unit of gasoline in the tank) and go back to the base. Exactly when the jeep arrive to base all gasoline filled at the start was spent. Then geep fill the 2nd unit of gasoline given, go 1/3 unit fill the gasoline he had put before and the tank is again full, then go forward until all the gasoline in the tank will be spent. Therefore the maximum distance the jeep can go is 4/3 unit of distance.

Guess the maximum distance when $n = 2$. We already know the maximum distance with n unit ofgasoline is D_n is expressed as the recursive relation $D_n = D_{n-1} + 1/(2n - 1)$.

Our interest is on whether an evolutionary computation can find a almost best strategy, say, $n = 5$ in which maximum distance is 1323/945=1.4. (If my calcuration is correct. Try it by yourself).

6 Knapsack Problem

This is one of the most popular *NP-complete Combinatorial Optimization* problems.

We now assume n items whose i-th item has weight w_i and profit p_i, then we pick up x_i of the i-th item $i = 1, 2, \cdots, n$ and x_i is non-negative integer. The goal is to maximizes

$$\sum_{i=1}^{n} x_i p_i. \tag{7}$$

such that

$$\sum_{i=1}^{n} x_i w_i \leq C \tag{8}$$

where C is the capacity of the knapsack.

GA implementation is quite simple. Our chromosomes are in the form

$$(x_1 x_2 x_3 \cdots x_n) \tag{9}$$

7 Traveling Salse-person Problem

Asuuming N cities all of whose cordinate are given. Traveling Salse-person Problem (TSP) is a problem in which a sales-person should visit all of these cities once but only once and objective is to look for the shortest tour.

I found that 13,509 real cities in USA are given with their cordinates in the web-page http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95. Why don't we try this very challenging problem by ourselves. The plotted these cities are shown in Figure 3.

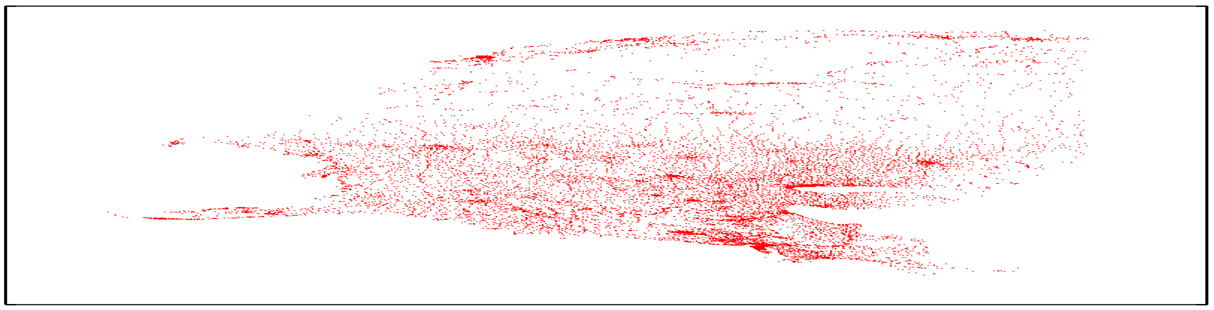


Figure 3: An example of 13509 real cities' location in US. Ploted with the data taken from http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

The question is how we design our chromosome representing a tour? If we represent a candidate solution with a list of cities to be visited in order, such as the chromosome of the tour A-C-F-D-G-E-B is

$$(ACFDGEB) \tag{10}$$

Are the the results of crossover and mutation feasible? The answer is No! For example, the possible child of two parent $(ACFDGEB)$ and $(AGBFCDE)$ by *one-point-crossover* could be $(ACFFCDE)$ and this is not a feasible tour because C and F are visited twice and B is not visited. Or, if we give a standard mutation to $(ACFDGEB)$, for example, by replacing 4th gene with other randomly chosen city, such as $(ACFAGEB)$, which is not a feasible tour either.

Then, in order for the result of crossover and mutation to be feasible what representations are possible?

One idea is

```
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 $i \le n$.
```

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.

Try some one-point crossover on two parents (112141311) and (515553321).

Then next, how we design mutation? How about, for example, specifying two points at random and reverse the gene order?

Finally, when, on earth, we stop a run? We don't know the optimum. The answer is let's observe the evolution of fitness. We can expect the run converges to the optimum or near-optimum.

Here we explore the TSP by an GA. But there seems to be more direct way of computation called Ant Colony Optimization (ACO). ACO is an optimization technique we borrowed the metaphore of an intelligence of ant society as their collective behaviour. Ants are good at seeking a shortest path from their nest to a food when one of their colleagues finds it he communicates with others by using a chemical called pheromon. If we have a time we will study ACO later.

8 **Evolving Strategy** — Iterated Prisonner's Dilemma

In the community of *Game Theory* we have the problem called Prisoner's Dilemma[2] which is formulated as follows.

Problem (Prisoner's Dilemma) *Two newly arrested prisoners A and B are offered a deal:*

- *If A confesses and B does not, A will be forbidden and B will get 5 years in jail, and vice versa.*
- *If both confess, then both will get 4 years in jail.*
- *If both do not they will each get 2 years.*

So, the reward that A/B will receive is summarized as:

$B\backslash A$	Cooperate	Defect
Cooperate	3/3	0/5
Defect	5/0	1/1

Defection increases one's own reward at the expense of the opponent, while *Cooperation* increases the reward for both players, fewer though. In any case it would be better to defect!

[2] Proposed by Merrill Flood and Melvin Dresher in the 1950's

When we assume this *award matrix* in more general form, that is,

$B\backslash A$	Cooperate	Defect
Cooperate	γ_1/γ_1	γ_2/γ_3
Defect	γ_3/γ_2	γ_4/γ_4

situation in which a dilemma is given rise to is when (Papopurt, 1966)

$$2\gamma_1 > \gamma_2 + \gamma_3 \tag{11}$$

$$\gamma_3 > \gamma_1 > \gamma_4 > \gamma_2 \tag{12}$$

Iterated Prisoner's Dilemma

The next question then is how about if the game is repeated? This is called the *Iterated Prisoner's Dilemma (IPD).* In this case *strategy* to get a higher award as a result arises: What would be the optimal next action? For example, *Always Defect* strategy, or *Tit-for-Tat* strategy where the player cooperates on the first play, and afterward the same action as the opponent in the previous game.

Here, strategy determins the next action based on three previous moves of the two players in a raw. Number of all possible previous three games is $2^6 = 64$ — 64 combination of Cooperate and Defect. Namely, all those possible combinations of 6 previous moves can be represented with a 64 bit binary chromosome. For example, if a history of 6 previous actions of opponent and the player is C-d-D-d-C-d we express the history by the binary number 100010 where "C" and "c" are expressed by 1 and "D" and "d" are expressed by 0 and uppercase "C" and "D" are opponent's and lowercase "c" and "d" is playere's Cooporation and Defection, respectively.

Then the next action when the hisory was (000000) is put on the 1st bit with 0 being defection and 1 being cooperation. The same is repeated, that is, the next action when the history was (000001) is put on the 2nd bit and so on. No need to repeat but, the 3rd bit is the action after the history of (000010) and the 64-th bit, the last bit, is when the history was (111111) Thus we can represent a strategy with a 64-bit binary chromosome.

where fitness is assigned with each individual playing against with each of some other individuals (tournament selection).

9 Visualization of high-dimensional space

— Sammon Mapping by GA

We, human, couldn't imagine the world of more than 3-dimensional space. In many scientic field, however, it is crucially important to grasp an image in high dimensional space. Hence visualization of high-dimensional space, or dimension reduction technique is very important. So far many such techniques has been proposed, among which Kohonen's Self Organizing Map is very popular above all.

Here we learn about Sammon Mapping. Sammon Mapping is a mapping a set of points a in high-dimensional space to the 2-dimensional space with the distance relation being preserved as much as possible, or equivalently, the distances in the n-dimensional space are approximated by distances in the 2-dimensional distance with a minimal error.

This method was proposed in 1980's as an optimization problem to which they approached by Operations Research technique suchas *Steepest Descend*, which is not so simple. Here, on the other hand, we employ Evolutionary Computatins which is quite simple. Let's see now what is the original Sammon Mapping look like.

Algorithm (Sammon Mapping)

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.*

This is an optimization problem which we now can solve quite simply by using EC. That is, *by creating N points in 2-D space each of which corresponding N points in the n-D space with the distance relation being preserved as much as possible, or equivalently, such that the n-D distances are approximated by 2-D distances with a minimal error.*

In an actual GA implementation of Sammon Mapping, chromosomes might be made up of n genes each of which corrisonds to $x-y$ coordinate of a candidate solution of n optimally distributed points in 2-dimensional space. Uniform crossover is employed and from time to time mutation is given by replacing one gene with other random $x-y$ coordinate. See the Figure 2. See also the Figure bellow.
Examples in $49^2 = 2401$ dimensional space:

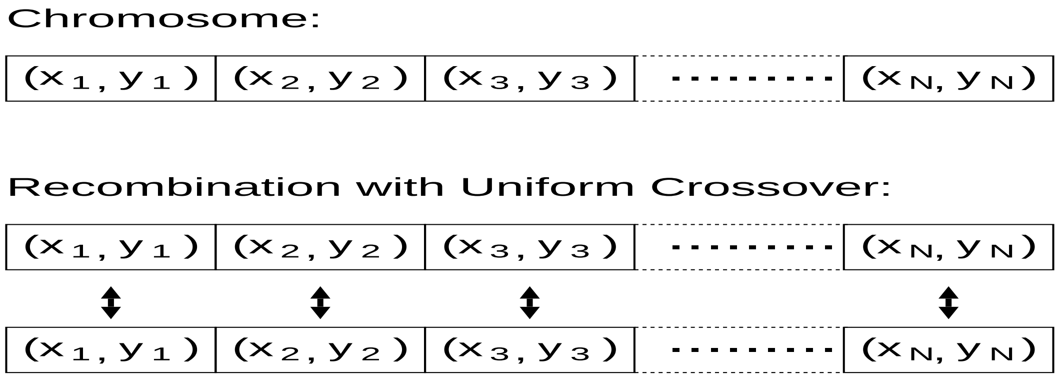

Chromosome:
(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_N, y_N)
Recombination with Uniform Crossover:
(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_N, y_N)
(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_N, y_N)

Figure 4: A chromosome representation and uniform crossover

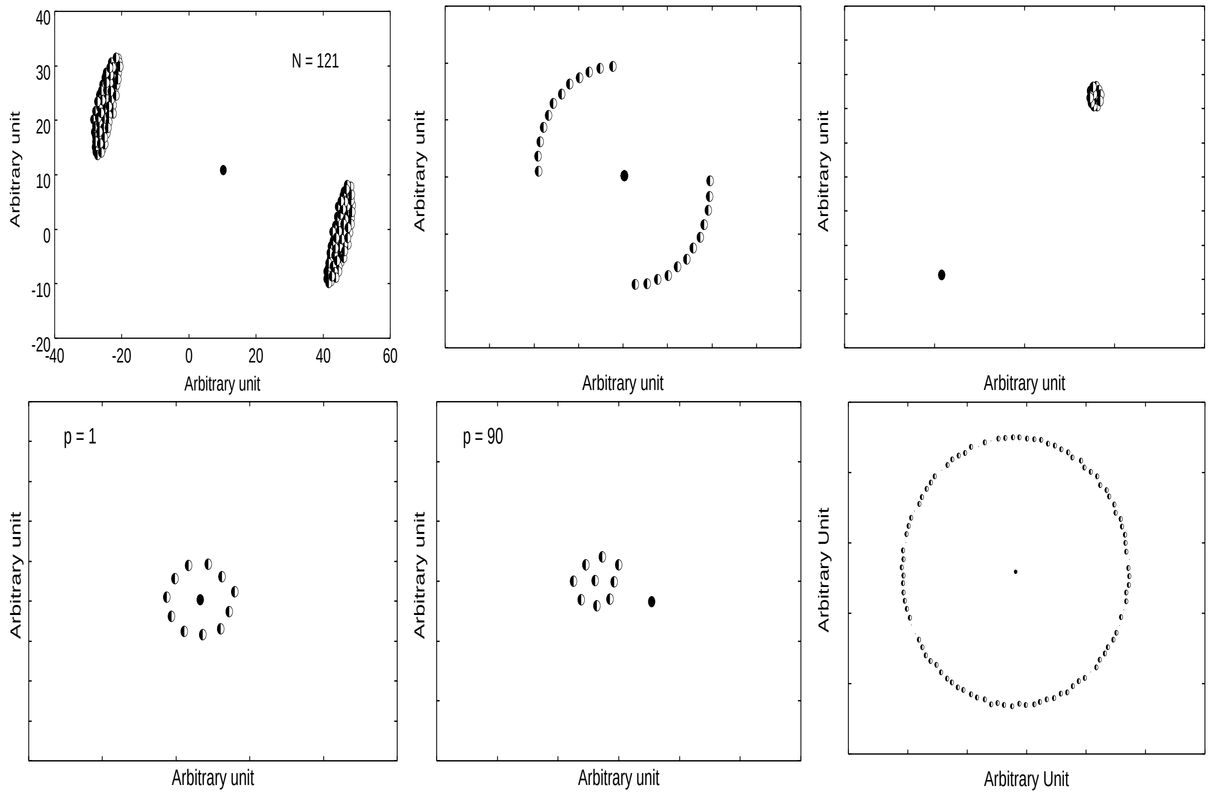


Figure 5: Six Examples of Mapping from 2401-dimensional space to the 2-dimensional space. Further explanations are shown in the text.

10 Let's be more biological: Part I

— Exploitation of Diploidy Chromosomes

We we write an algorithm, we often come accross a necessity to sort a set of items in the order of some criteria. For example how we create codes for sorting 16 integer inputs in ascending order. We pick up 2 items from one item to the next, compare, and swap them if the order is not the preferable one Then problem is what is the minimum number of comparisons with which any arbitrary set of 16 inputs are correctly sorted. That is,

Problem (Sorting Network) *The task is to sort n items. For the purpose, the i-th and j-th element are compared and swap if necessary. and the goal is to find an algorithm which correctly sorts all n items with the minimum number of comparisons.*
It might be interesting to overview a little history of this topic. In 1960's, in a comunity of computer algorithms, there was a competion of what is the minimum number of comparisons when, say, $(n = 16)$? The result was

- ⋆ 65 comparisons Bose and Nelson (1962).
- ⋆ 63 by Batcher and by Floyd and Knuth (1964).
- ⋆ 62 by Shapiro (1969)
- ⋆ 60 by Green (1969)

See the Figure bellow.

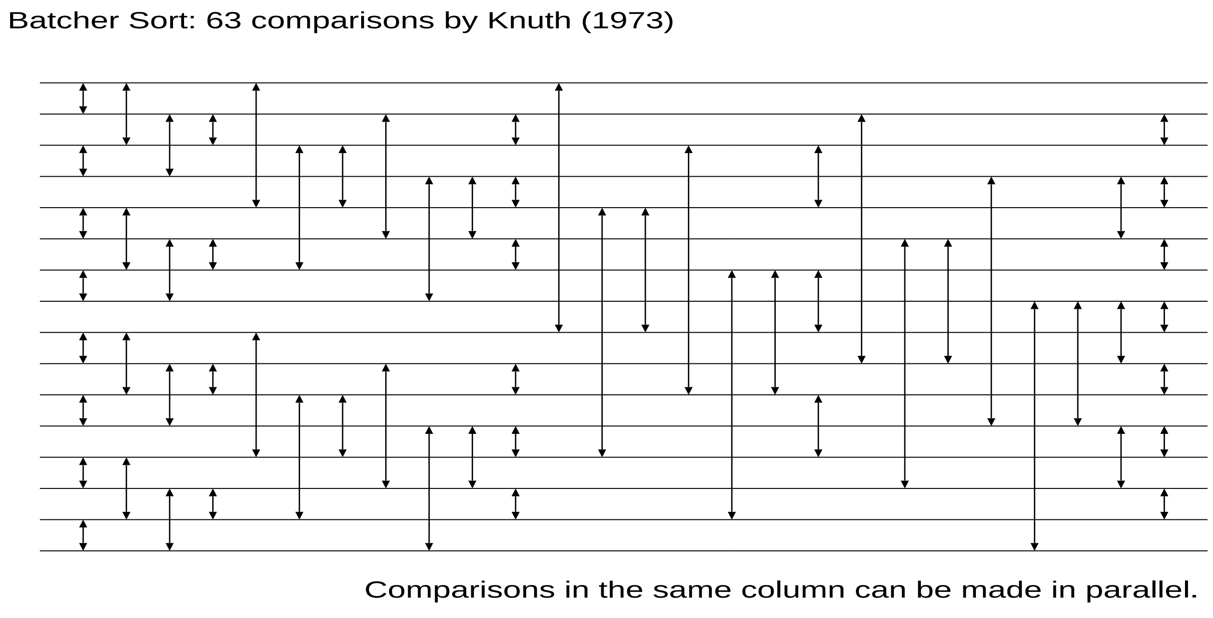

Batcher Sort: 63 comparisons by Knuth (1973)
Comparisons in the same column can be made in parallel.

Figure 6: Batcher's proposition of sorting network with 63 comprisons (1964)

Up to now, however, we have had no proof for this optimality. Then let's make an Evolutionary Computation surch for this nimimum number. Would it work better than by human? Hillis (1992) challenged this. Hillis's innovations was that he employed *Diploidy Chromosome* as follows.

(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)

Figure 7: An example of Hillis's set of diploidy chromosomes.

- Each individual consists of 15 pairs of 32-bit chromosomes.
- Each chromosome consists of eight 4-bit string (called *codons*).

(0001 0010 0101 1000 0000 0100 1111 1001)
(0011 0100 0101 1000 1101 1100 1111 1001)

- Each codon represents an integer between 0 and 15 indicating which item is to be compared out of 16, e.g. in the above example.

(01 02 05 08 00 04 15 09)
(03 04 05 08 13 12 15 09)

- Each adjacent pair of codons in a chromosome specifies a comparison between two elements. Thus each chromosome encodes four comparisons, e.g.,

(09 08 10 13 14 04 14 03)

indicates the four comparisons bellow.

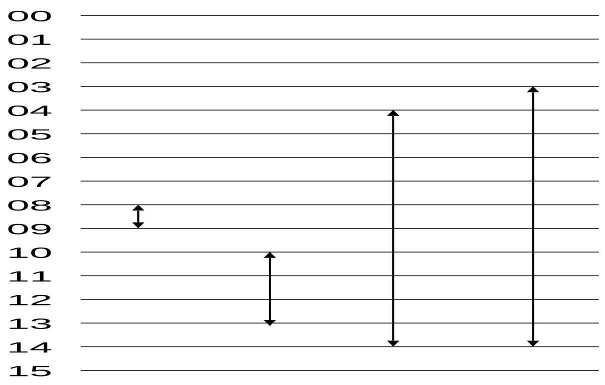


Figure 8: Four comparisons specified by the chromosome (09 08 10 13 14 04 14 03).

- The chromosome pairs is read off from left to right.
- If these adjacent two codons are same at the same corrisponding two positions of the chromosome pair (called *homozygous*) then only one pair of numbers is inserted in the phenotype. If it encodes different pairs of numbers (*heterozygous*) then both pairs are inserted in the phenotype. So in the previous example:

  (01 02 05 08 00 04 15 09)
  (03 04 05 08 13 12 15 09)

  means the following six comparisons:

  01⇔02, 03⇔04, 05⇔08, 00⇔04, 13⇔12, 15⇔09

- Thus 15 pairs of chromosomes produce a phenotype with 60-120 comparisons. The more homozygous positions, the fewer comparisons.
- When two individuals are selected, one-point-crossover takes place within each chromosome pair inside each individual.
- For each of the 15 chromosome pairs, a crossover point is chosen at random and a single chromosome (called *gamete*) is formed.
- Thus 15 gametes from each parent are created.
- Each of the 15 gametes from the first parent is then paired with the gamete of the corrisponding position from the second parent to form one offspring.

11 Fitness Landscape

In Figure reffig:fitness-l a Concept of Fitness Landscape is plotted on also fictiteous domain of 2-dimensional space. For all the possible points in search space, fitness value is calculated and plotted. If the domain is more than 3-dimensional, we could not visualize it, but we could imagin it as a hyper-surface. This (hyper-)surface is called a fitness landscape, usually include peaks and the top of the highest peak corresponds to the global solution, and top regeon of other lower peaks correspond to local optima.

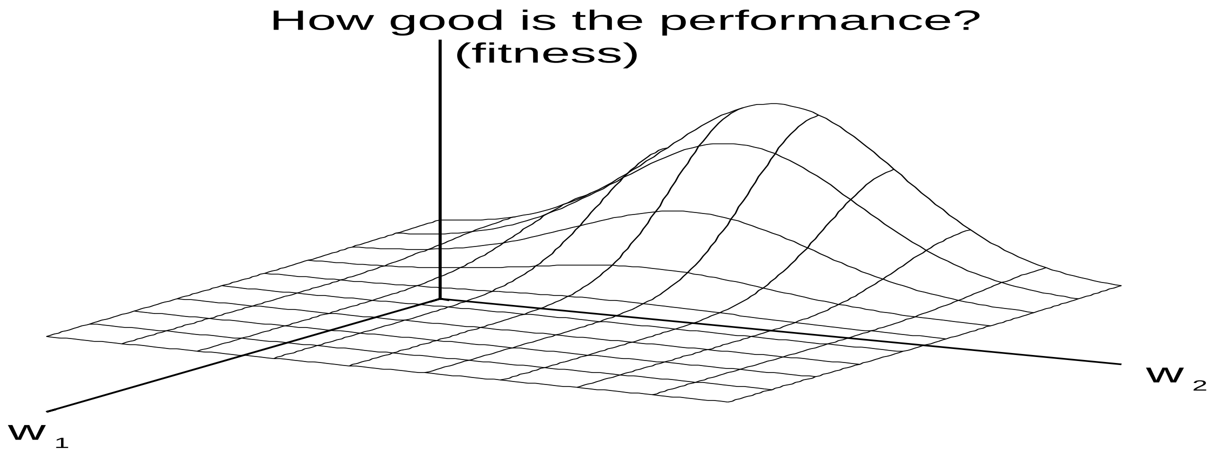


Figure 9: A conceptual plot of fitness value defined on a fictitious 2-dimensional space.

However, what if a landscape doesn't have such gradiant information? That is to say, if the peak is like a mesa which has a flat reagion, extremely steep sidewall, and anywhereelse is totally flat land of fitness zero — we (personally) call it *"A-Tiny-Flat-Island-in-a-Huge-Lake"*. Or, in extreme case, what if only one point in the search space has the fitness one and all other points have fitness zero — This is called a *"A-Needle-in-a Haystack"*, like in Figure .

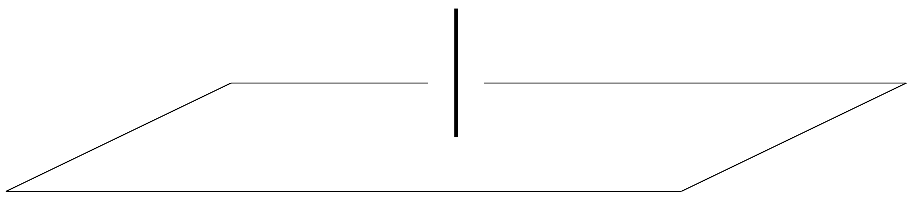


Figure 10: Conceptual figure of a needle in a haystack.

12 Why Giraffe has a long neck?

— Lamarckian Inheritance & Baldwin Effect

A Needle in a Haystack — The most difficult problem.

Once Hinton & Nowlan proposed a very interesting experiment to search for A-needle-in-a-haystack. Their needle and haystack is as follows.

- A-needle $\Rightarrow$ Only one configuration of 20 bits of binary string.
- Haystack $\Rightarrow$ $2^{20} - 1$ search points.

For example, (11111111110000000000) is assigned fitness one, while others are fitness zero. Conceptually, we could assume that given a black box to detect a needle like and our task

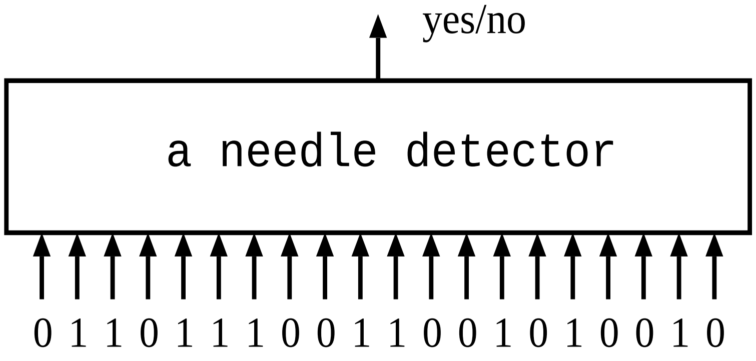


Figure 11: A conceptual black-box to detect a needle.

is to search for the 20-bit binary input which return "YES". We have only one such string out of all the possible 2^{20} inputs.

What Hinton & Nowlan proposed is a learning of each individual during its lifetime. In a biological evolution, if the result affects the next generation, this is called a *Baldwin Effect*. In the Hinton & Nowlan's proposal this is as follows:

- **lifetime learning of each individual (Baldwin Effect).**
  - *about 25% are "1", 25% are "0", and the rest of the 50% are "?" .*
  - *They are evaluated with all the "?" position being assigned "1" or "0" at random $\Rightarrow$ learning.*
  - *Each individual repeats the learning up to 1000 times $\Rightarrow$ lifetime learning.*
  - *If it reaches the point of fitness one at the n-th trial, then the degree to which learning succeeded is calculated as*

$$1 + 19 \cdot (1000 - n)/1000.$$

13 Multi-modal Optimization.

— To obtain multiple solutions at a time in a run.

Sometimes we have multiple solutions. But EC usually converges only one solution out of them. Hence, to get those multiple solution we run the algorithm multiple time. Here we learn the technique in which individual construct niches and each species found a different solution at a run.

For the purpose, multiple *species* will be created and maintained in a population. These species independently search for a peak (hopefully an optimum solution), construct their *niche* and stay around the peak during a run.

In comparatively early days, essentially the following three methods were proposed. So-far proposed methods are

- Fitness sharing (Goldberg & Richardson, 1987)
  - · Similar individuals share fitness with each other.
- Crowding (De Jong, 1975)
  - · Similar individuals are replaced with random individuals
- Species Method.
  - · Mating is restricted to among similar individuals.

These days, IMHO, the following two are popular among others.

- Detterministic Crwoding (Mahfoud, 1992)
- Sequential Niching

Let's see some of the aspects more in detail.

Fitness Sharing Fitness of each individual is derated by an amount related to the number of similar individuals in the population. That is, shared fitness $F_s(i)$ of the individual i is

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

where $F(i)$ is fitness of individual i; d_{ij} is distance between individual i and j; Typically d_{ij} is *Hamming distance* if in *genotypic space Euclidean distance* if in *phenotypic space* and $s(\cdot)$ is called *sharing function* and defined as:

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

where σ_{share} is interpreted as size of niche, and α determines the shape of the function. The denominator is called *niche count*. You see shape dependency of $s(d_{ij})$ on α in Figure 13.

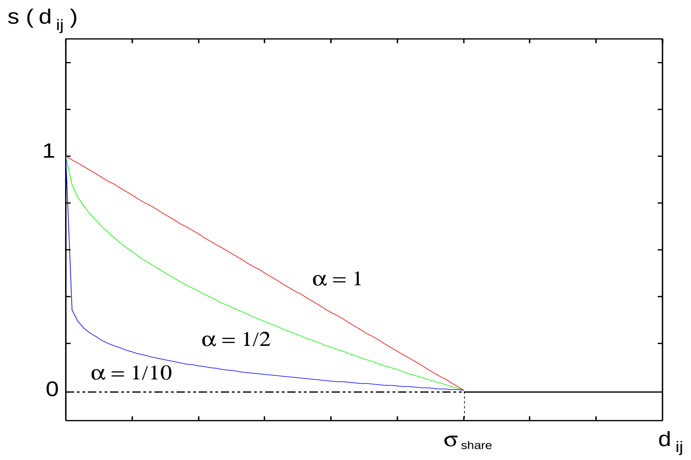


Figure 12: A shape dependency of $s(d_{ij})$ on α.

To be short (not so short though): Similar individual should share fitness. The number of individuals that can stay around any one of peaks (niche) is limited.
The number of individuals stay near any peak will theoretically be proportional to the *hight* of the peak

Deterministic Crowding: If the parents will be replaced or not with their childeren will be determined under a criteria of the distance between parents and children,

p

Algorithm Assuming crossover, mutation and fitness function are already defiened

1. *Choose two parents, p_1 and p_2, at random, with no parent being chosen more than once.*
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

where $d(\zeta_1, \zeta_2)$ *is the Hamming distance between two points* (ζ_1, ζ_2) *in pattern configuration space. The process of producing child is repeated until all the population have taken part in the process. Then the cycle of reconstructing a new population and restarting the search is repeated until all the global optima are found or a set maximum number of generation has been reached.*

Sequential Niche: Single run is repeated sequentially, keeping the best individual at each run.

Algorithm

1. *Define niche radius* r.
2. *Define modified-fitness function* $m(\mathbf{x})$ *by equating it to the original fitness function* $f(\mathbf{x})$ *here at the start.*
3. *Run the GA and pick up the best individual at the end of the run.*
4. *Update* $m(\mathbf{x})$ *as* [3]

$$m_{n+1}(\mathbf{x}) = m_n(\mathbf{x}) \cdot g(\mathbf{x}, \mathbf{s}_n) \tag{14}$$

where n *is the number of so-far run,* s_n *is the best individual in the* n*-th run and*

$$g(\mathbf{x}, \mathbf{s}_n) = \begin{cases} (d_{xs}/r)^\alpha & \textit{if } d_{xs} < r \\ 1 & \textit{otherwise} \end{cases} \tag{15}$$

is called derating function where d_{xs} *is a distance between* $\mathbf{x})$ *and* $\mathbf{s}_n)$ *while* m_0 *is the original raw fitness function of each individual.*

5. *Run the GA using the modified fitness function and keep the best individual found in the run,*
6. *Update the modified fitness function*
7. *If the raw fitness of the best individual exceeds the solution threshold, (See also below) then display this as a solution.*
8. *If all solutions have not been found, then return to step 5.*

[3] This is called a *Power Derating Function* when we think of another alternative called *Exponential Derating Function:*

$$g_e(x, s) = \begin{cases} \exp((\log m(x, s)) \cdot (r - d_{xs}/r)) & \text{if } d_{xs} < r \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

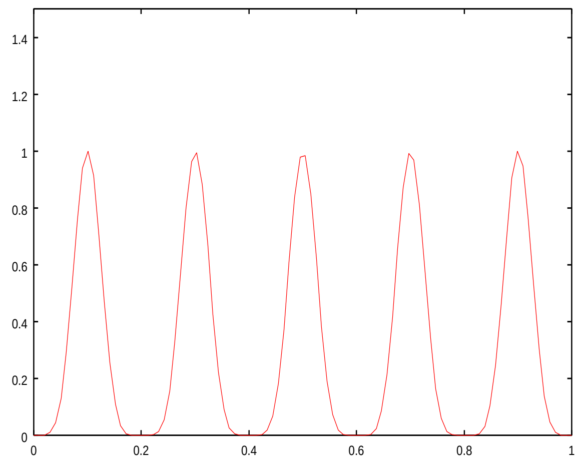
- *Solution Threshold is*
    - · *Lower fitness limit for maxima of interest, assuming we know how many peaks. If it's not of the case, set the threshold to zero.*

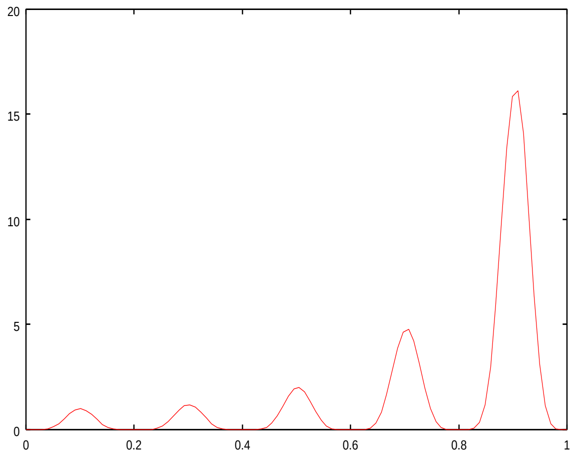
Excersize *Like Figure 13, draw a graph of* $y = (x/r)^\alpha$ *with* $r = 1$ *and* $\alpha = 0.5, 1, 2, 4, 8$ *to know what* $g(\mathbf{x}, \mathbf{s}_n)$ *looks like.*

It would be interesting to try a multi-modal EC to the following two test functions.

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

(2) $y = -2((x - 0.2)/0.8)^2 \sin^6(5\pi x)$





14 Multi Objective Genetic Algorithm (MOGA)

So far we have learned how to get the possible solution(s) which fulfills one objective function for the problem, that is, the goal is maximize the fitness function. In real world problem, however, we have usually multiple objectives or criteria to be fulfilled simultaneously.

Those objectives sometimes conflict with each other. Like "time" and "money": The more we want to earn money, the less time to spent the money; or "reliability" of the product and "cost" to produce it in a manufactural factory. Or, suppose an Opera Company trys to employ one Soprano singer. The criteria is voice, beauty-or-not), slim-or-not, language-capability (Italian, German, etc). However God tend not to give us two talents at a time, alas.

Then, first of all, when we have multiple objective function, we must define an important concept of parate optimal or equvalently non-dominated solution.

Definition (Parate Optimal or Non-dominated Solution) *A candidate solution is called a non-dominated iff there is no ohter better solution w.r.t. all the objectives.*

To be more specific, assume we have n objective functions;

$$f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), \cdots f_n(\mathbf{x})$$

where $\mathbf{x}$ is a candidate solution. Now if a new candidate solution $\mathbf{y}$ improves all the objetives for $\mathbf{x}$, i.e.,

$$f_i(\mathbf{y}) > f_i(\mathbf{x}) \text{ for } \forall i$$

we say

"$\mathbf{y}$ *dominate* $\mathbf{x}$."

When no suche $\mathbf{y}$ exists, we say

"$\mathbf{x}$ *is non-dominated*" or "*Parete Optimum.*"

A toy example: We now assume the two objective functions as follows.

$$\begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$$

- x=0 is optimum w.r.t. f_1 but not so good w.r.t. f_2.
- x=2 is optimum w.r.t. f_2 but not so good w.r.t. f_1.
- Any other point in between is a compromise or trade-off and is a Pareto-optimum.
- But the solution x=3, e.g., is not a Pareto-optimum since this point is not better than the solution $x = 2$ w.r.t. either objective.
- If we plot in the f_1-f_2 space, an increase in f_1 in some reagion means a decrease in f_2, or vice versa which implys that the solutions in the region are Parete optimum, while in other region an increase in f_1 make f_2 increas (decrease). See Figure **??**. This f_1-f_2 space is called a *Trade-off Space.*

Thought Experiment: What if we plot all individuals of generation 0 and, say, generation 100?

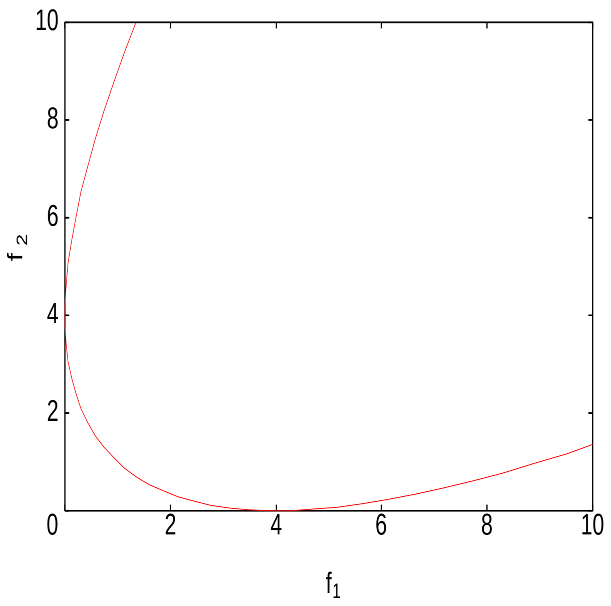


Figure 13: Trade-off space for $f_1(x) = x^2$ and $f_2(x) = (x-2)^2$.

How an implementation goes? Evolution is rather similar more or less to a GA with single fitness function. The main difference is we have multiple objective function. Hence we merge these multiple objective function into one fitness function. So far many ideas have been proposed. Among all:

- Weighted sum approach.
  - · The fitness function is caluculated as
  $$f(\mathbf{x}) = \sum_{i=1}^{N} w_i f_i(\mathbf{x}) \tag{16}$$
  where w_i expresses an importance of the i-th objectives.
  - · Note that for any set of weight > 0, the optimum is always a non-dominated solution but opposite is not always true.
- The minimux approach
  - · The fitness function is caluculated by minimizing the maximum of n objective functions.
- Target vector approach
  - · The fitness function is caluculated by minimizing the vector
  $$(f_1, f_2, f_3, \cdots, f_n)$$
  from a predifined goal.

- Median/Average ranking approach

  · The rank $r_1(\mathbf{x}_i)$ of each individual x_i in the population w.r.t. i-th objective function is calculated. Then the fitness is defined as median/average of these $r_i \quad (i = 1, \cdots, n)$.

- Parete ranking approach

  · Ranking is according to "*how many individuals in the population they are dominated by.*

We now take a look at a typical implemetation of MOGA.

Algorithm (A Multi Objective GA)

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.* [4]
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.*

15 Evolution of Sturcture — From a Simple Structure to Complex Ones

15.1 Finite State Machine

[4] Step 5 implies that the new child is only inserted into the population if it dominates the most similar individual, or if it has a lower ranking, i.e. a lower degree of dominance.

The restricted replacement strategy also constitutes an extreme form of elitism, as the only way of replacing a non-dominated individual is to create a child that dominates it.

The similarity of two individuals is measured using a distance function.