Birkbeck UNIVERSITY OF LONDON

(Concepts of) Machine Learning

Lecture 5: Genetic and Evolutionary Algorithms

George Magoulas
gmagoulas@dcs.bbk.ac.uk

Contents

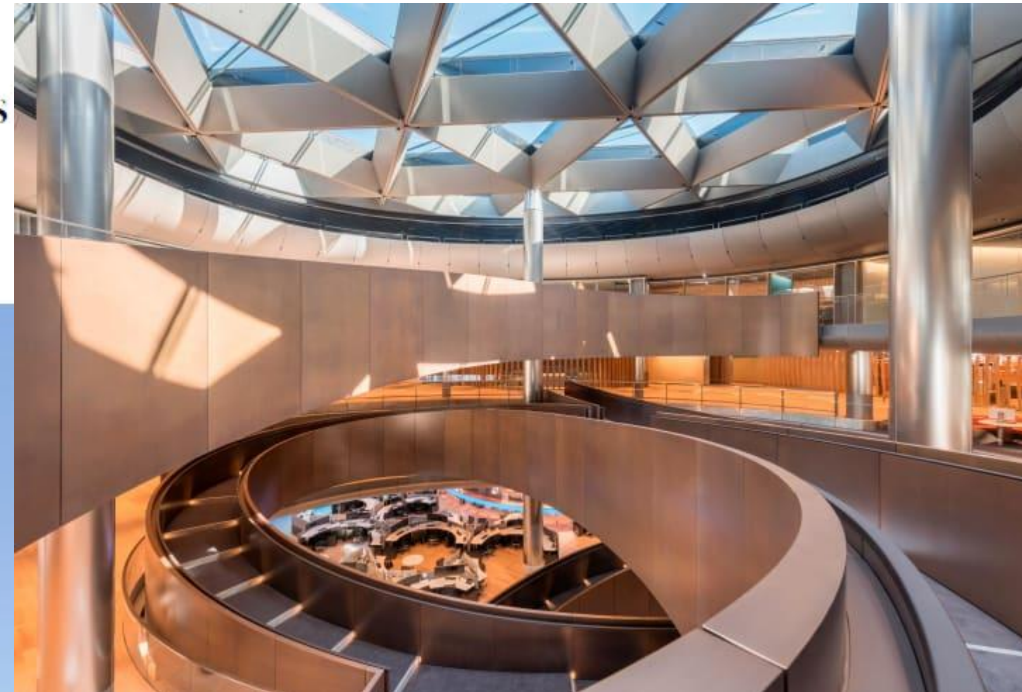
- Some problems are too difficult to solve
- Bio-inspired machine learning
- Evolution and computation
- Genetic algorithms: basic principles and concepts, mechanisms, elitism, pseudocode
- Convergence of GAs
- Evolutionary algorithms (EA): the evolution strategy, stages and pseudocode
- Examples

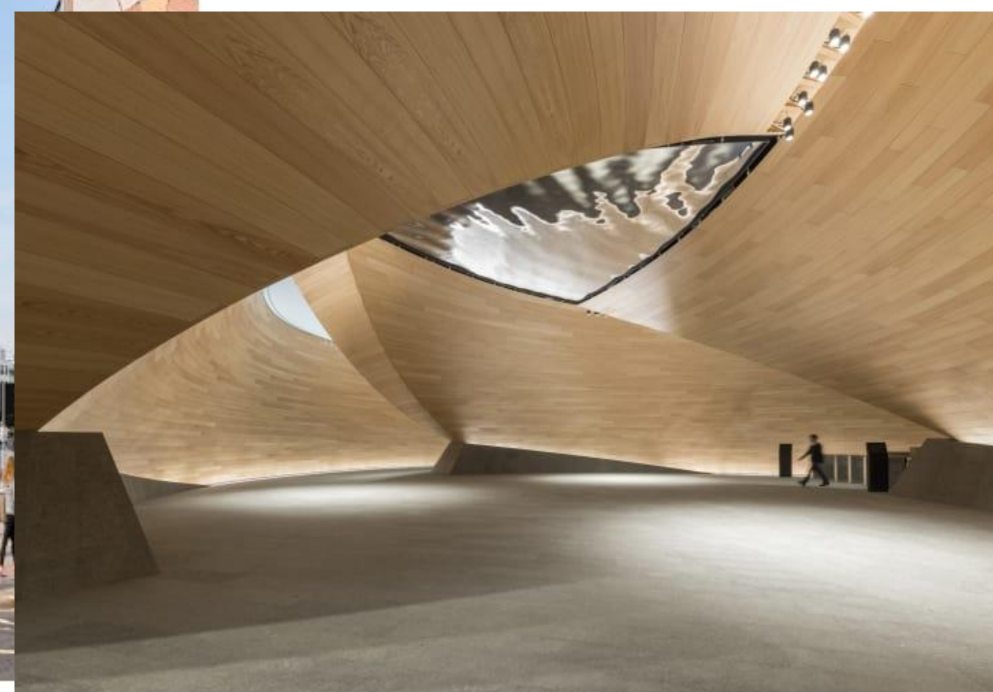
Architecture

Bloomberg's European HQ named UK's new building

Published 10th October 2018

Credit: James Newton





Designed by Foster+Partners

- "This was one of those projects where we gladly demonstrate the power of automation. We developed optimization methods (usually used in the field of Artificial Intelligence) in order to automate the generation of complex steel details for the atrium of the building. The elements were optimized and the whole process from 3D geometry generation, over blueprint creation to the CNC machine ready files was completely automated. "

Some problems are too difficult to solve-1

Reason 1: The size of the search space or the complexity of the objective function may preclude classical optimisation methods

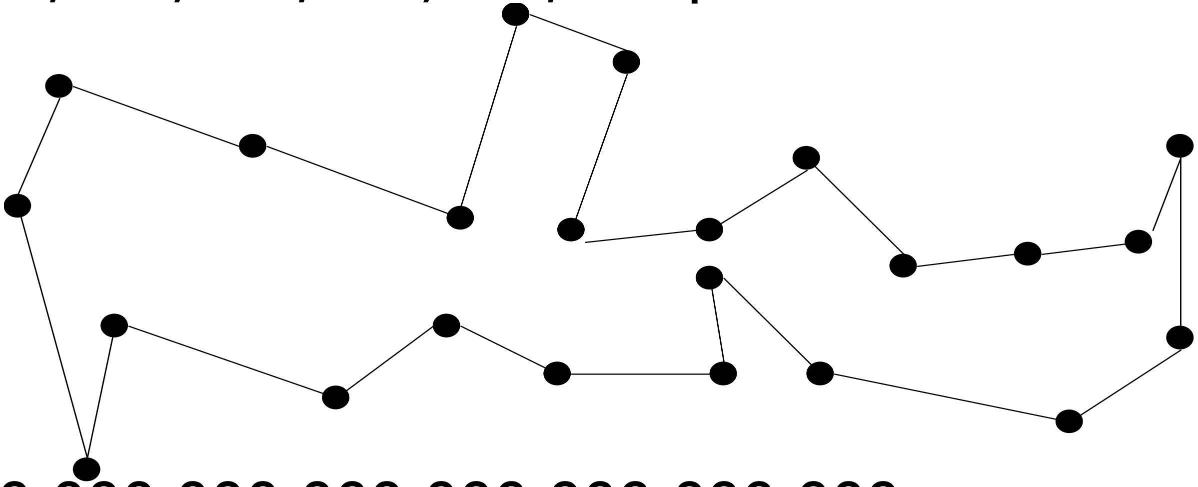
Examples: Optimise $f(x_1, x_2, \ldots, x_{100})$ where f is complex and x_i is 0 or 1. The size of the search space is $2^{100} \approx 10^{30}$. It is not possible to perform an exhaustive search

TSP: find the shortest path through each city and return home

Some problems are too difficult to solve-2

A 10 city TSP has 181,000 possible solutions

A 20 city TSP has 10,000,000,000,000,000 possible solutions



A 50 city TSP has

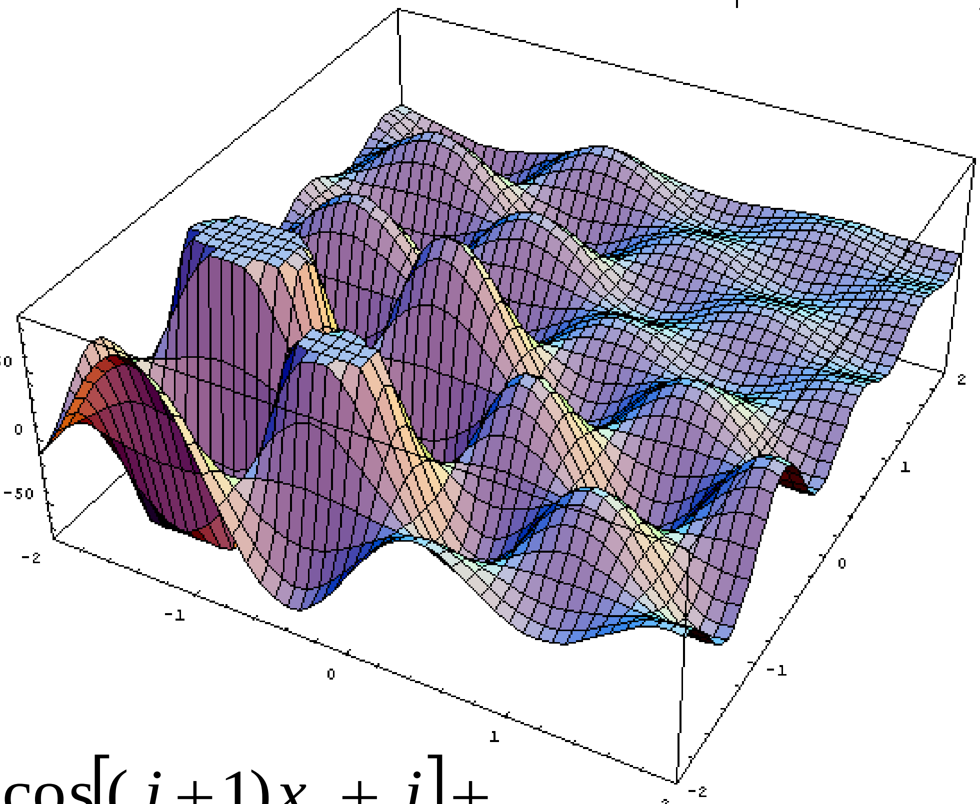
100,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000

i.e. 10^{22} possible solutions

Some problems are too difficult to solve-3

Reason 2: Global optimisation problems

Example: The notorious *Levy No. 5* has 760 local minima; 1 global minimum at (-1.3068,-1.4248)



$$f(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \times \sum_{j=1}^{5} j\cos[(j+1)x_2 + j] + $$
$$+ (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2,$$
$$-10 \le x_i \le 10,\ i = 1,2.$$

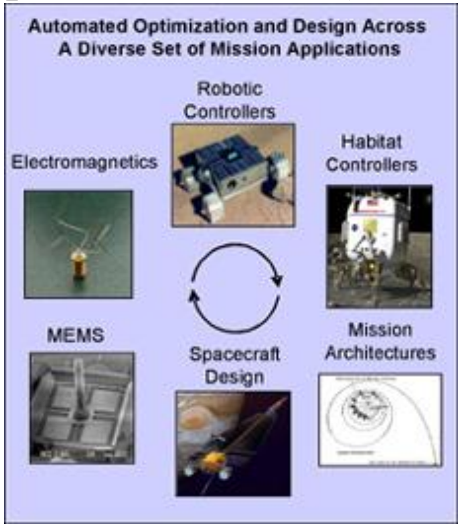
Some problems are too difficult to solve-4

Reason 3: Many real-world problems are heavily constrained

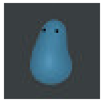
Examples: Optimise $f(x_1, x_2, \ldots, x_n)$ with bounds $l_i \leq x_i \leq u_i$ for $1 \leq i \leq n$ and subject to constraints: $g_i(x) \leq 0$ ($i = 1, \ldots, q$) and $h_j(x) = 0$ ($j = q+1, \ldots, m$)

- Time tabling problems
- Design optimisation problems

(http://www.programmingarchitecture.com/
- Urban planning; Energy Efficiency
- Antennas; evolution of a 2D car)
- Machine learning (neural networks)



Bio-inspired machine learning

- In biological evolution, **learning and evolution are two principal forms of adaptation that differ in time and space**.
- **Evolution** is a process involving selective reproduction and substitution based on presence of population of individuals displaying some variability. 
- **Learning** is a set of adjustments taking place within each individual in the population during its own lifetime.

Evolution and learning

- Evolution is a type of adaptation that captures relatively slow environmental changes that involves several generations, i.e. evolution operates at the *phylogenetic level*.
- Learning includes various set of mechanisms that lead to adaptive changes in an individual during its lifetime, i.e. learning operates on the *ontogenetic level*.

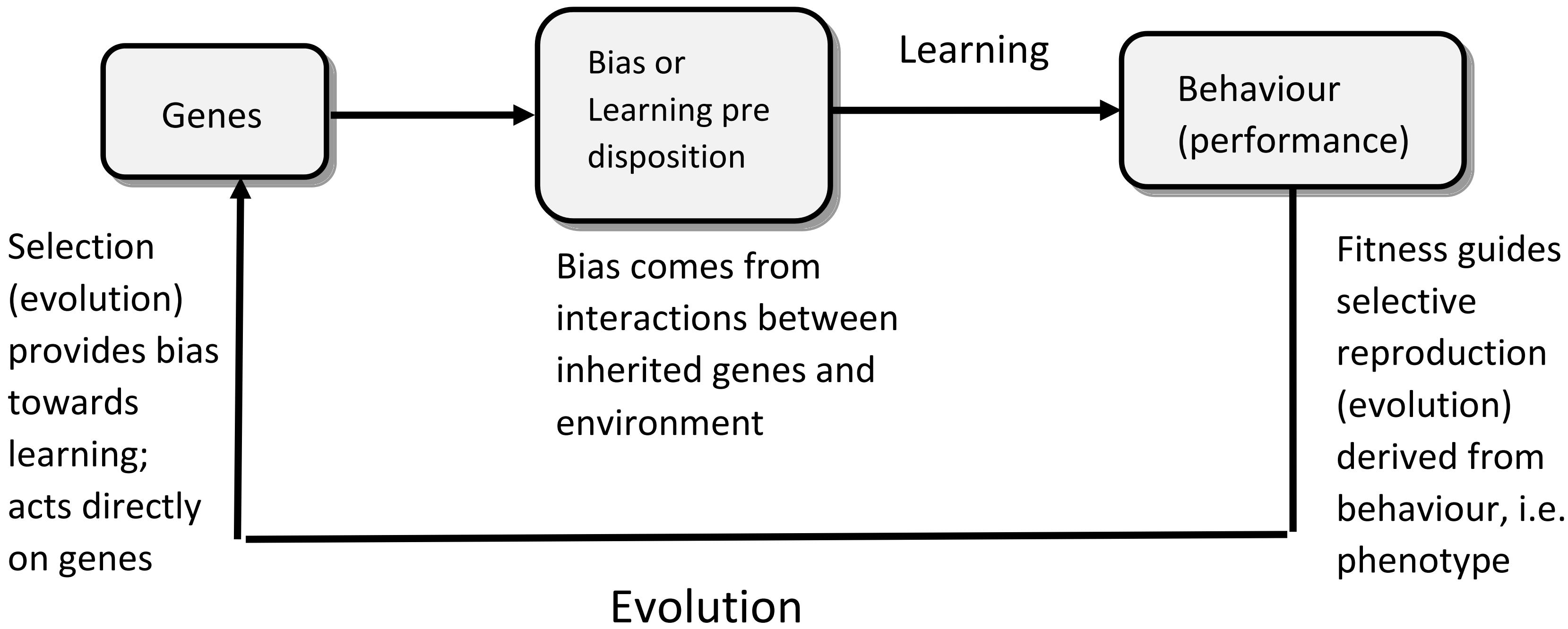
Evolution and learning

The idea of interaction between learning and evolution was first proposed by Baldwin (1896) and Lloyd Morgan (1896) and is commonly referred to as the Baldwin Effect. Waddington (1942) also proposed a similar kind of interaction which is called canalisation or genetic assimilation.

Evolution and learning

- The key concept in all the aforementioned theories is that **what a species must initially learn during each individual's lifetime, can overtime become part of the genetic makeup of that species**, i.e. what is initially learned eventually becomes innate
- The structure of all cognitive abilities that we possess like language acquisition, reasoning arise from the interactions between learning and evolution.

Bio-inspired machine learning



Changes in the environment might be slow and subtle as in *concept drift* or they might occur abruptly as in *concept shift*.

Bio-inspired machine learning

1. Randomly create an initial **population** of different artificial genotypes, each of which encodes machine learning models configurations (e.g. free parameters)
2. Train (i.e. learning) and evaluate each individual of the population to determine the **fitness** (based on performance).
3. Based on chosen **selection** criterion**,** the selected models **reproduce** by creating copies of their genotypes with addition of changes introduced by **genetic operators** like cross over.
4. Repeat steps 1-3 for number of generations till the models satisfy performance/termination criterion set by user

Evolution and computation

- **Simulating evolution on a computer**. The result of such a simulation is a series of optimisation algorithms, usually based on a simple set of rules. Optimisation iteratively improves the quality of solutions until an optimal, or at least feasible, solution is found.

- The evolutionary approach is based on computational models of natural selection and genetics. We call them **evolutionary computation**, an umbrella term that combines **genetic algorithms**, **evolution strategies** and **genetic programming**.
- The behaviour of an individual organism is an inductive inference about some yet unknown aspects of its environment. If, over successive generations, the organism survives, we can say that this organism is capable of learning to predict changes in its environment. 

Some history

- On 1 July 1858, **Charles Darwin** presented his theory of evolution before the Linnean Society of London. This day marks the beginning of a revolution in biology.
- Darwin's classical **theory of evolution**, together with Weismann's **theory of natural selection** and Mendel's concept of **genetics**, now represent the neo-Darwinian paradigm.
- Have a look at: https://en.wikipedia.org/wiki/Modern_synthesis_(20th_century)

- Evolution can be seen as a process leading to the maintenance of a population's ability to survive and reproduce in a specific environment. This ability is called **evolutionary fitness**.
- Evolutionary fitness can also be viewed as a measure of the organism's ability to anticipate changes in its environment.
- The fitness, or the quantitative measure of the ability to predict environmental changes and respond adequately, can be considered as the quality that is optimised in natural life.

How is a population with increasing fitness generated?

- Let us consider a population of rabbits. Some rabbits are faster than others, and we may say that these rabbits possess superior fitness, because they have a greater chance of avoiding foxes, surviving and then breeding.
- If two parents have superior fitness, there is a good chance that a combination of their genes will produce an offspring with even higher fitness. Over time the entire population of rabbits becomes faster to meet their environmental challenges in the face of foxes.

Simulation of natural evolution

- All methods of evolutionary computation simulate natural evolution by creating a population of individuals, evaluating their fitness, generating a new population through genetic operations, and repeating this process a number of times.
- We will start with **Genetic Algorithms** (GAs) as most of the other evolutionary algorithms can be viewed as variations of genetic algorithms.

Genetic Algorithms

- In the early 1970s, John Holland introduced the concept of genetic algorithms.
- Holland was concerned with algorithms that manipulate strings of binary digits.
- Each artificial "chromosome" consists of a number of "genes", and each gene is represented by 0 or 1:

1	0	1	1	0	1	0	0	0	0	0	1	0	1	0	1

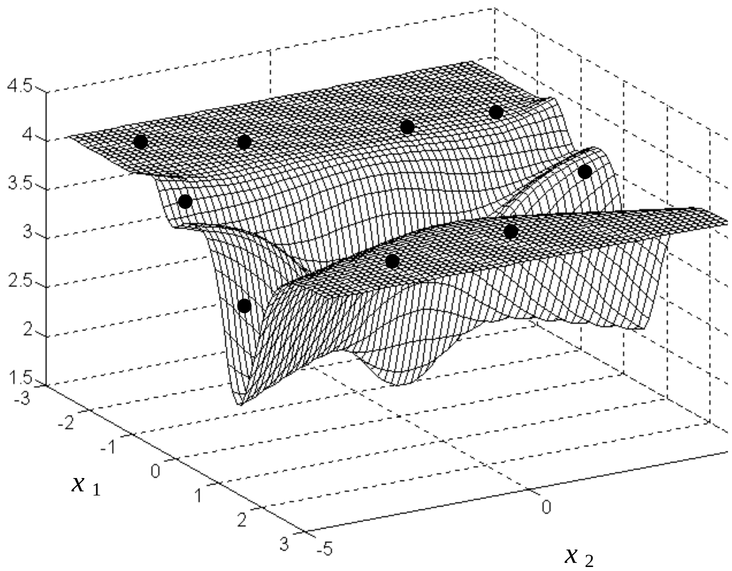
- Two mechanisms link a GA to the problem it is solving: **encoding** and **evaluation**.
- The GA uses a **measure of fitness of individual chromosomes** to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes, and the mutation operator changes the gene value in some randomly chosen location of the chromosome.

Basic principles and key concepts-1

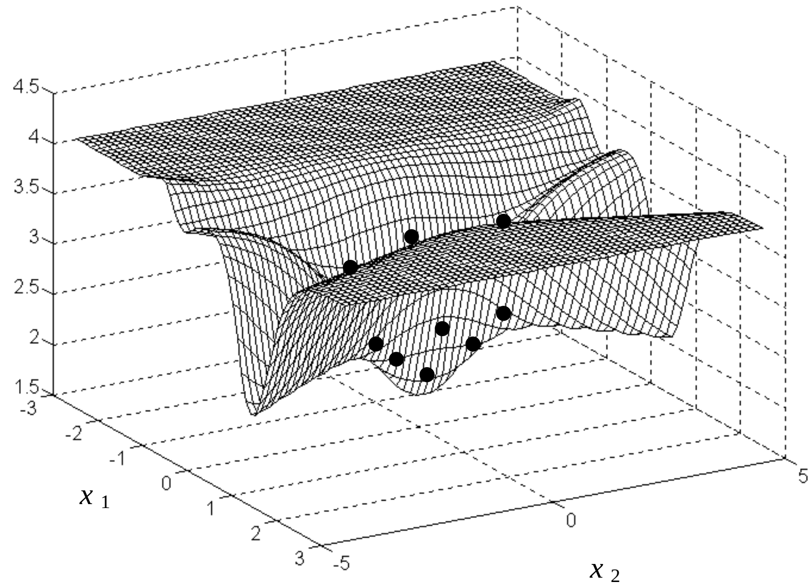
1. Randomly generate an *initial* ***population*** of ***chromosomes***
2. Compute the ***fitness*** of every member of the current population
3. Make an intermediate population by extracting members out of the current population by means of the ***selection operator***
4. Generate the new population by applying the ***genetic operators*** **(crossover, mutation)** to this intermediate population
5. If there is a member of the current population that satisfies the problem requirements then stop, otherwise go to step (2)

Basic principles and key concepts-2

- **Population:** *The maximum number of search points; collection of chromosomes that evolves from generation to generation*



Generation 1



Generation 10

Basic principles and key concepts-3

- **Fitness:** the measure of the performance of an individual on the actual problem; the function value of the search points

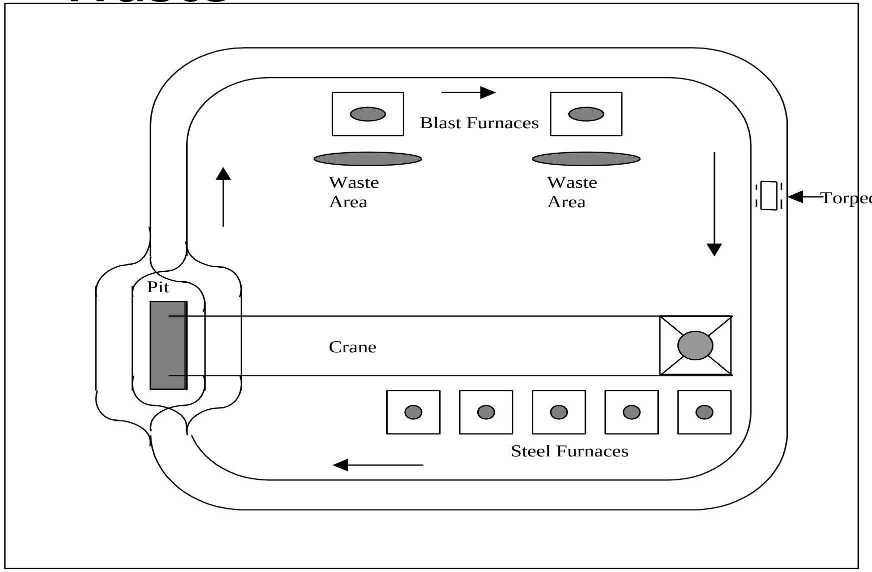
Example: *Fitness function for steel works plant*

Total_Monthly_Cost = Investment_cost + waste_monthly_cost

TMC = 2.1*Nt + 8.3*Nc +16.7*Sf +Waste

Variable	Variable Type	Range	Number of Possibilities
1. Number of Torpedoes	Integer	1-12	12
2. Number of Cranes	Integer	1-2	2
3. Number of Steel Furnaces	Integer	1-6	6
4. Volume of the torpedo (in tonnes)	Real	50-350	300

Table 1 - Range of component values



Basic principles and key concepts-4

- **Encoding:** Discretisation of the variable values; allows manageability in the population of the search points

  - Binary
  - Real-valued

Basic principles and key concepts-5

Binary encoding

binary problems: $x=(a_1, ...,a_l)$, $a_i \epsilon \{0,1\}$

real-valued problem: $x_1 x_2 x_3 \in [0,1]$

00000000→0;
11111111→1;
00000001→1/256=0.0039

$$x = (00000001 \quad 00101000 \quad 10110001)$$

Basic principles and key concepts-6

Real-valued encoding

$$x=(a_1, ...,a_l), \quad a_i \in \Re$$

bounded values: $a_i \in [\min, \max], \quad \min, \max \in \Re$

advantages: increased precision; shorter strings; freedom to use special genetic operators

Basic principles and key concepts-7
Selection of the parent strings

Key step that creates a sub-population for reproduction

A subsequent generation is created from the chromosomes in the current population. To this end:

- a group of chromosomes, generally called "parents", are selected via a specific **selection** routine.
- the genes of the parents are to be mixed and recombined for the production of offspring in the next generation.

It is expected that from this process of evolution (manipulation of genes) the better chromosome will create a larger number of offspring and thus it has a higher chance of surviving in the subsequent generation, emulating the survival-of-the-fittest mechanism in nature.

Basic principles and key concepts-8 Selection schemes

- **Proportionate reproduction:** this scheme selects individuals based on their fitness relative to the rest of the population
- **Tournament selection:** in this process a number of individuals, set by the *tournament size*, is selected from the population at random.

Selection schemes

- *Roulette wheel selection*: this is the most commonly used technique of the proportionate selection mechanism
  - **Sum the fitness of all the population members; named as total fitness *N***
  - **Generate a random number *n* between 0 and the total fitness *N***
  - **Return the first population member whose fitness added to the fitness of the preceding population members, is greater than or equal to *n*.**

Selection schemes

- *Deterministic tournament selection:* the best-fit individual of the tournament is chosen to reproduce. The simplest version, *binary* tournament selection, has a tournament size of two.

```
func tournament_selection(population, tournament size k)
    best = null
    for i=1 to k
        ind = population[random(1, N)]
        if (best == null) or fitness(ind) > fitness(best)
        best = ind
    return best
```

The winner of each tournament (the one with the best fitness) is selected for crossover

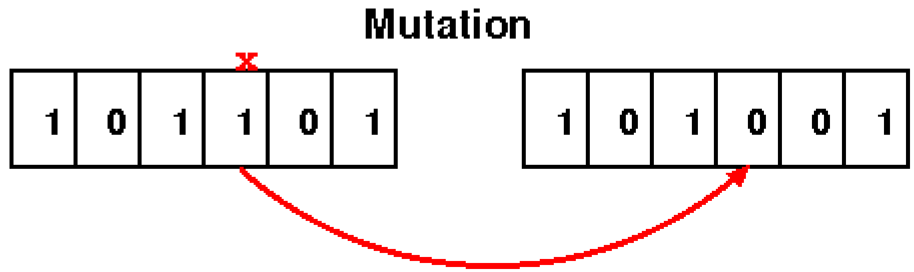
Basic principles and key concepts-9

•**Mutation operator:** determines the probability with which the data structures are modified

-Binary encoding: bit mutation is applied to each offspring individually. It alters each bit randomly with a small probability, called *mutation rate*, with a typical value of <0.1

- Real encoding: Gaussian mutation: $x_i' = x_i + N(0,\sigma)$
where $N(0,\sigma)$ is a random Gaussian number with mean zero and standard deviation σ (mutation stepsize)

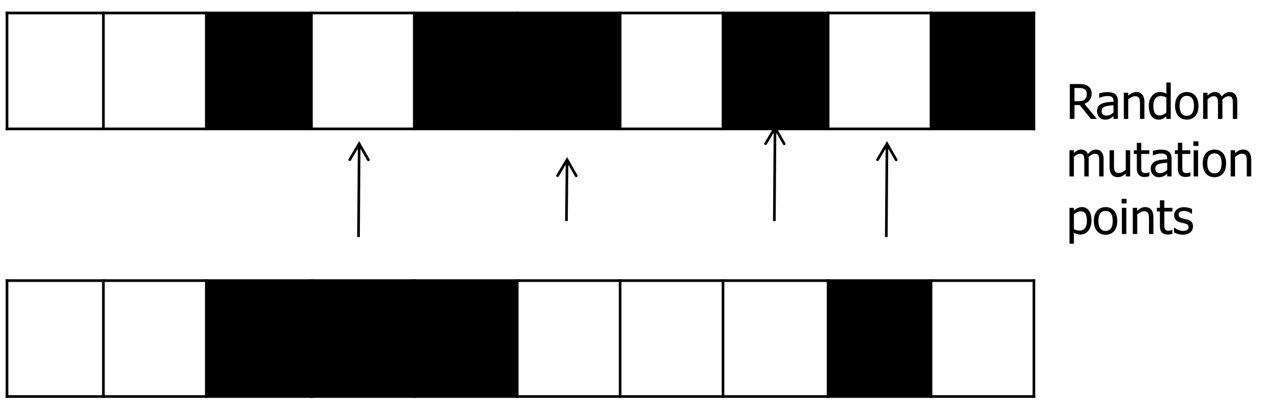
Mutation



- Aims at maintaining **diversity** within the population and **control premature convergence**.
- With small probability, a portion of the new individuals will have some of their bits flipped.
- Mutation alone induces a random walk through the search space

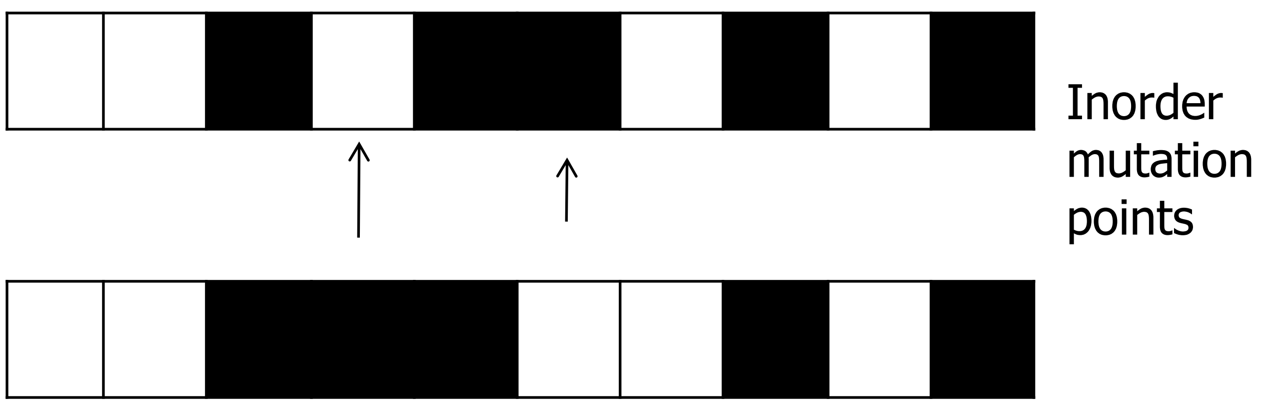
Mutation

- Random mutate: bit positions are chosen randomly and the corresponding bit negates.



Mutation

- Inorder mutate: two bit positions are randomly selected and only bits between these positions are mutated.



Mutation

(i) Binary encoding mutation

Example: Bit mutation on the 4th bit

Original chromosome

1	0	0	1	1	0	1	0

New chromosome

1	0	0	0	1	0	1	0

Mutation

(ii) Real valued encoding mutation

Gaussian mutation when each individual x is a vector of floating-point numbers, i.e. $x = < x_1, \ldots, x_n >$

$$x_i' = x_i + N(0, \sigma)$$

where $N(0,\sigma)$ is a random Gaussian number with mean zero and standard deviation σ (mutation stepsize)

Changing the mutation stepsize: $\sigma(t) = 1 - 0.9\frac{t}{T}$

where $0 \le t \le T$ is the current generation number

Assign a personal stepsize to each individual: $\sigma' = \sigma \cdot e^{N(0,\tau_0)}$

and $x_i' = x_i + N(0, \sigma')$

Assign a personal stepsize to each variable of each individual:

$\sigma_i' = \sigma_i \cdot e^{N(0,\tau_0)}$ and $x_i' = x_i + N(0, \sigma_i')$

Basic principles and key concepts-10

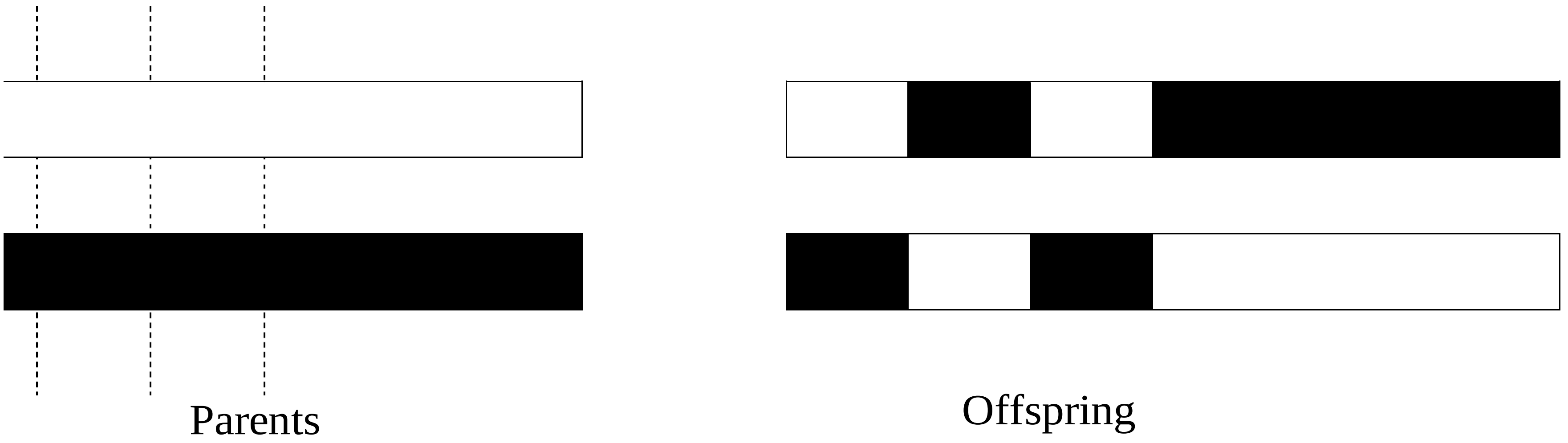
Crossover or recombination operator: exchange subparts of two chromosomes, roughly mimicking biological recombination between two single-chromosome organisms

1-point crossover with crossover site = 3

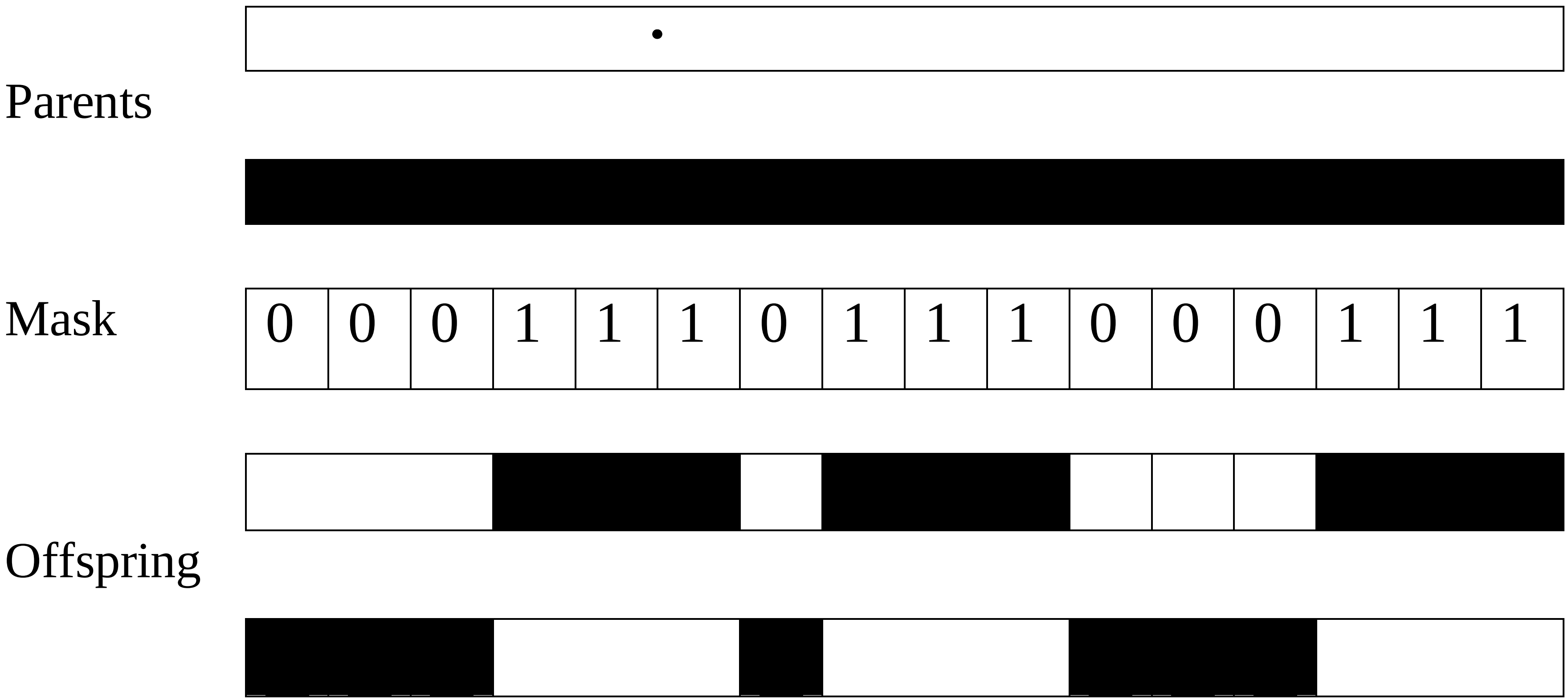
Chromosome	Crossover site=3	Resulting offspring
001101011	001 | 101011	001001101
111001101	111 | 001101	111101011

***One-point crossover*:** crossover point can be set randomly. The probability of crossover: *crossover rate* takes values 0.6-1.0

***Multi-point crossover*:** $m=3$



Uniform crossover



Examples:

1-point crossover with crossover site = 3

Chromosome	Crossover site=3	Resulting offspring
001101011	001 | 101011	001001101
111001101	111 | 001101	111101011

2-point crossover with crossover site = {4,7}

Chromosome	Crossover site={4,7}	Resulting offspring
001101011	0011 |010 | 11	0011 **011** 11
111001101	1110 | 011 | 01	1110 **010** 01

2-point crossover with crossover site = {7,4}

Chromosome	Crossover site={7,4}	Resulting offspring
001101011	0011 |010 | 11	**1110** 010 **01**
111001101	1110 | 011 | 01	**0011** 011 **11**

Basic principles and key concepts-11

Typical values used in practice

For large population size (100+)
Crossover rate = 0.6
Mutation rate = 0.001

For small population size (e.g. 30)
Crossover rate = 0.9
Mutation rate = 0.01

Basic principles and key concepts-12

Elitism is an optional characteristic of a GA that makes sure that the fittest chromosome of a population of *N* chromosomes is passed on to the next generation unchanged; it can never be replaced by another chromosome.

Without elitism this chromosome may be lost.
Extended forms of elitism are also possible where the best *m* chromosomes of the population are retained.
Simple elitism is the case where $m=1$.
The effect of elitism is that the number of offspring that are generated each generation is reduced from *N* to *N-m* replacing the worst *N-m* individuals in the population

GENETIC ALGORITHM MODEL AND MAIN STAGES

```
{
//start with an initial time
    t:=0;
//initialise a usually random population of
//individuals
STAGE 1: initpopulation P(t);
//evaluate fitness of all individuals of population
STAGE 2: evaluate P(t);
//test for termination criterion (time, fitness,
//etc.)
    while not done do
        //increase the time counter
        t:=t+1;
//select a sub-population for offspring production
STAGE 3: P':=selectparents P(t);
        //recombine the "genes" of selected parents
STAGE 4: recombine P'(t);
        //perturb the mated population stochastically
STAGE 5: mutate P'(t);
        //evaluate the new fitness
STAGE 6: evaluate P'(t);
        //select the survivors from actual fitness
STAGE 7: P:=survive P,P'(t);
    end
}
```

Convergence of genetic algorithms

- *How likely is it that the better bit patterns survive from one generation of a genetic algorithm to another?*

- This depends on the probability with which they are selected for the generation of new child strings and with which they survive the recombination and mutation steps.

GAs convergence problem

- John Holland (1975) suggested the notion of *schemata* for the convergence analysis of GAs.

  *Schemata are bit patterns which function as representatives of a set of binary strings. The bit patterns can contain each of the three symbols 0, 1 or *.*

- <u>Example:</u> The schema **00** is a representative of all strings of length 6 with two zeros in the central positions, such as: 100000, 110011, 010010, etc.

schema

H =								
*	1	0	*	0	1	0	0	1

This schema describes the following subset of the search space:

H={(**0** 1 0 **0** 0 1 0 0 1),(**0** 1 0 **1** 0 1 0 0 1),
(**1** 1 0 **0** 1 0 0 1),(**1** 1 0 **1** 0 1 0 0 1)}

The schema theorem

In the long run the best bit patterns will diffuse to the whole population

- The function $f(x)$ has to be maximised. The function is defined over all binary strings of length l and is called the fitness of the strings.
- The number of strings in the population, in generation t, that contain the bit pattern H is $O(H,t)$.
- The diameter of a bit pattern H is $d(H)$, with $d(H) \geq 1$

  (The diameter of a bit pattern is defined as the length of the pattern's shortest substring that still contains all fixed bits in the pattern. For example, the bit pattern **1*1** has diameter 3 because the shortest fragment that contains both constant bits is the substring 1*1 and its length is 3).
- Two parent strings from the current population are always selected for the creation of a new string.

The schema theorem

The probability that a parent string H_j will be selected from N strings $H_1, H_2, \ldots, H_N$ is

$$p(H_j) = \frac{f(H_j)}{\sum_{i=1}^{N} f(H_i)}$$

This means that strings with higher fitness are more likely to be selected than strings with lower fitness

Let f_μ is the average fitness of all strings in the population, i.e.

$$f_\mu = \frac{1}{N}\sum_{i=1}^{N} f(H_i)$$

The probability $p(H_j)$ can be rewritten as

$$p(H_j) = \frac{f(H_j)}{N f_\mu}$$

The schema theorem

The probability that a schema H will be passed on to a child string can be calculated in the following three steps:

i) **Selection.** The whole population is the basis for each individual parent selection. The probability P that a string is selected which contains the schema H is:

$$P = \frac{f(H_1)}{Nf_\mu} + \frac{f(H_2)}{Nf_\mu} + K + \frac{f(H_k)}{Nf_\mu}$$

where H_1, H_2, …, H_k represent all strings of the generation which contain the bit pattern H. If there are no such strings then P=0.

The schema theorem

The *fitness of the schema H* in the generation *t* is defined as:

$$f(H) = \frac{f(H_1) + f(H_2) + \Lambda + f(H_k)}{O(H,t)}$$

Thus *P*, i.e. *the probability that a string, which contains the schema H, is selected* can be rewritten as

$$P = \frac{O(H,t) f(H)}{N f_\mu}$$

The *probability* P_1 *that two strings, which contain the schema H, are selected as parent strings* is given by

$$P_1 = \left(\frac{O(H,t) f(H)}{N f_\mu} \right)^2$$

The *probability* P_2 *that from two selected strings only one contains the pattern H* is

$$P_2 = 2 \frac{O(H,t) f(H)}{N f_\mu} \left(1 - \frac{O(H,t) f(H)}{N f_\mu} \right)$$

The schema theorem

(ii) Recombination. For the recombination of two strings a cut-off point is selected between the positions *1* and $l-1$ and then crossover is carried out.

The probability W that a schema H is transmitted to the new string depends on two cases:

- If both parent strings contain H, then they pass on this substring to the new string.
- If only one of the strings contains H, then the schema is inherited at most half of the time.

The substring H can also be destroyed with probability $\frac{d(H)-1}{l-1}$ during crossover.

The schema theorem

This means that

$$W \geq \left(\frac{O(H,t) f(H)}{N f_{\mu}} \right)^2 + \frac{2}{2} \frac{O(H,t) f(H)}{N f_{\mu}} \left(1 - \frac{O(H,t) f(H)}{N f_{\mu}} \right) \left(1 - \frac{d(H)-1}{l-1} \right)$$

The probability W is greater than or equal to the term on the right because in some favourable cases the bit string is not destroyed by crossover (one parent string contains H and the other parent part of the H).

After some calculations

$$W \geq \frac{O(H,t) f(H)}{N f_{\mu}} \left[1 - \frac{d(H)-1}{l-1} \left(1 - \frac{O(H,t) f(H)}{N f_{\mu}} \right) \right]$$

The schema theorem

(iii) Mutation. When two strings are recombined, the information contained in them is copied bit by bit to the child string. A mutation can produce a bit flip with the probability p.

This means that a pattern H with $b(H)$ fixed bits will be preserved after copying with probability

$$(1-p)^{b(H)}$$

If a mutation occurs *the probability W of the schema H being passed on to a child string* changes according to W_{new}, where

$$W_{new} \geq \frac{O(H,t)f(H)}{Nf_{\mu}}\left[1-\frac{d(H)-1}{l-1}\left(1-\frac{O(H,t)f(H)}{Nf_{\mu}}\right)\right](1-p)^{b(H)}$$

The schema theorem

If in each generation N new strings are produced, *the expected value of the number of strings which contain H in the generation t+1 is* NW_{new}, i.e.

$$\langle O(H,t+1)\rangle \geq \frac{O(H,t)\,f(H)}{Nf_{\mu}}\left[1-\frac{d(H)-1}{l-1}\left(1-\frac{O(H,t)\,f(H)}{Nf_{\mu}}\right)\right]\quad (1-p)^{b(H)}$$

This results states that in the long run the best bit patterns will diffuse to the whole population; when the mutation rate is too high ($p \approx 1$) schemata are destroyed.

Evolutionary Algorithms
Evolution strategies – 1

- Each individual is represented by its genetic material/characteristics and a set of strategy parameters, which model the behaviour of this individual in its environment.
- Evolution consists of evolving both genetic characteristics and the strategy parameters, where the evolution of genetic characteristics is controlled by the strategy parameters.
- Mutation changes are only accepted if successful, i.e. they produce an individual that possess better fitness.
- Offspring can also be produced from more than one set of parents.

Evolution strategies – 2

Multi-membered ES are denoted by $(\mu+\lambda)$ and (μ, λ) the so-called PLUS STRATEGY and COMMA STRATEGY, respectively

- in the *plus* case, the parental generation is taken into account during selection, while
- in the *comma* case only the offspring undergoes selection, and the parents die off.
- μ denotes the population size, and λ denotes the number of offspring produced per generation.

Evolution strategies – 3

Step 1 of the evolution process: Crossover

- **Local crossover**: where one offspring is generated from two parents using randomly selected components of the parents.

or

- **Global crossover**: where the entire population of individuals takes part in producing one offspring. Components are randomly selected from randomly selected individuals and used to generate offspring.
- ES can also work without crossover- this is how ES were initially proposed.

Evolution strategies - 4

Step 2 of the evolution: Mutation applied in two-stages

- STAGE 1: Mutate the standard deviation for current generation and each individual (strength of mutation)
  - $\sigma_{g+1,n} = \sigma_{g,n} e^{\tau \xi_\tau}$

    $\tau = 1/\sqrt{I}, \quad$ the n-th individual has I genetic variables

    $\xi_\tau \sim N(0,1)$

  - or $\quad \sigma_{g+1,n} = \sigma_{g,n}(1 + \tau \xi_\tau)$

N: the standard normal distribution (usually independent random samples extracted from N)

Evolution strategies - 5

- STAGE 2: Mutate the genetic material for each individual
  - $\boldsymbol{x}_{\boldsymbol{g}+1,\boldsymbol{n}} = \boldsymbol{x}_{\boldsymbol{g},\boldsymbol{n}} + \boldsymbol{\sigma}_{\boldsymbol{g}+1,\boldsymbol{n}}\xi$

    $\xi \in \boldsymbol{R}_{+}^{\boldsymbol{I}}, \quad \text{the } \boldsymbol{n}\text{-th individual has } \boldsymbol{I} \text{ genetic variables}$

    $\xi_{\boldsymbol{i}} \sim \boldsymbol{N}(0,1)$

- Mutated individuals are accepted only if the fitness of the mutated individual is better than the original individual.
- ES do not model mutation as a purely random process. A random process would mean that a child is completely independent of its parents.

Evolution strategies -6

Step 3 of the evolution: Selection

- (μ+λ) **selection**: λ offspring from μ parents, $1 \leq \mu \leq \lambda < \infty$

  Next generation consists of μ best individuals selected from the μ parents (of the previous generation) and the λ offspring. It needs to implement a form of *elitism* in case the fittest parents must survive to the next generation.

  or

- (μ, λ) **selection**: **next generation** consists of the μ best individuals selected from the λ offspring. Their parents are "forgotten", $1 \leq \mu < \lambda < \infty$

Evolution strategies -7

The (μ, λ) **selection** scheme is more realistic and therefore more successful in several applications, because no individual may survive forever (which could at least theoretically occur using the plus strategy). Only by *forgetting* highly fit individuals can a permanent adaptation of the stepsizes take place and one can avoid long stagnation phases due to misadapted σ_i 's. This means that these individuals have built an internal model that is no longer appropriate for further progress, and thus should better be discarded.

EA Model

```
{
//initialise the time counter
     t := 0;
//initialise the population of individuals
     InitPopulation(P(t));
//evaluate fitness of all individuals
     F_P(t):= Evaluate(P(t));
//test for termination criterion (time,
//fitness, etc.)

     while not done do
       t := t + 1;
//some methods use selection for offspring
//production like in Gas. Not general
       Q(t) := SelectParents(P(t));
//recombine the ``genes'' of selected parents
//like in GAs. Not general
       R(t) := Recombine(Q(t));
//perturb the population stochastically; more
//important than recombination
       M(t) := Mutate(R(t));
//evaluate the new fitness
       F_M(t):= Evaluate(M(t));
//select the survivors for the next generation.
//Not general

       P(t + 1) := Survive(F_P(t), F_M(t));
     end
}
```

Application examples - simple numeric example

A simple GA with population size 4, single-point crossover and bitwise mutation is applied on the fitness function:

$$f(x) = \text{number of ones in bit string } x$$

where x is chromosome of length 8. The initial, randomly generated, population is:

Chromosome label	Chromosome string	Fitness
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

Simple numeric example

-Two pairs of chromosomes are chosen as parents: chromosomes B and D constitute the first pair, and chromosomes B and C the second pair of parents.

-- Parents B and D cross over after the first bit position to form offspring E and F, and parents B and C do not cross over, instead forming offspring that are exact copies of B and C.

$$E = 10110100$$

$$F = 01101110$$

-Offspring E is mutated at the sixth bit position to form E_m, offspring F and C are not mutated at all, and offspring B is mutated at the first bit position to form B_m.

$$E_m = 10110000$$

$$B_m = 01101110$$

GA: simple numeric example

What will the population be after one generation?

What is the fitness of each member of the new population?

Chromosome label	*Chromosome string*	*Fitness*
E_m	*10110000*	*3*
F	*01101110*	*5*
C	*00100000*	*1*
B_m	*01101110*	*5*

GA- Maximise a function

Let us find the maximum value of the function $(15x - x^2)$ where parameter x varies between 0 and 15. For simplicity, we assume that x takes only integer values. Thus, chromosomes can be built with only four genes:

Integer	*Binary code*	*Integer*	*Binary code*	*Integer*	*Binary code*
1	0 0 0 1	6	0 1 1 0	11	1 0 1 1
2	0 0 1 0	7	0 1 1 1	12	1 1 0 0
3	0 0 1 1	8	1 0 0 0	13	1 1 0 1
4	0 1 0 0	9	1 0 0 1	14	1 1 1 0
5	0 1 0 1	10	1 0 1 0	15	1 1 1 1

Maximising a function

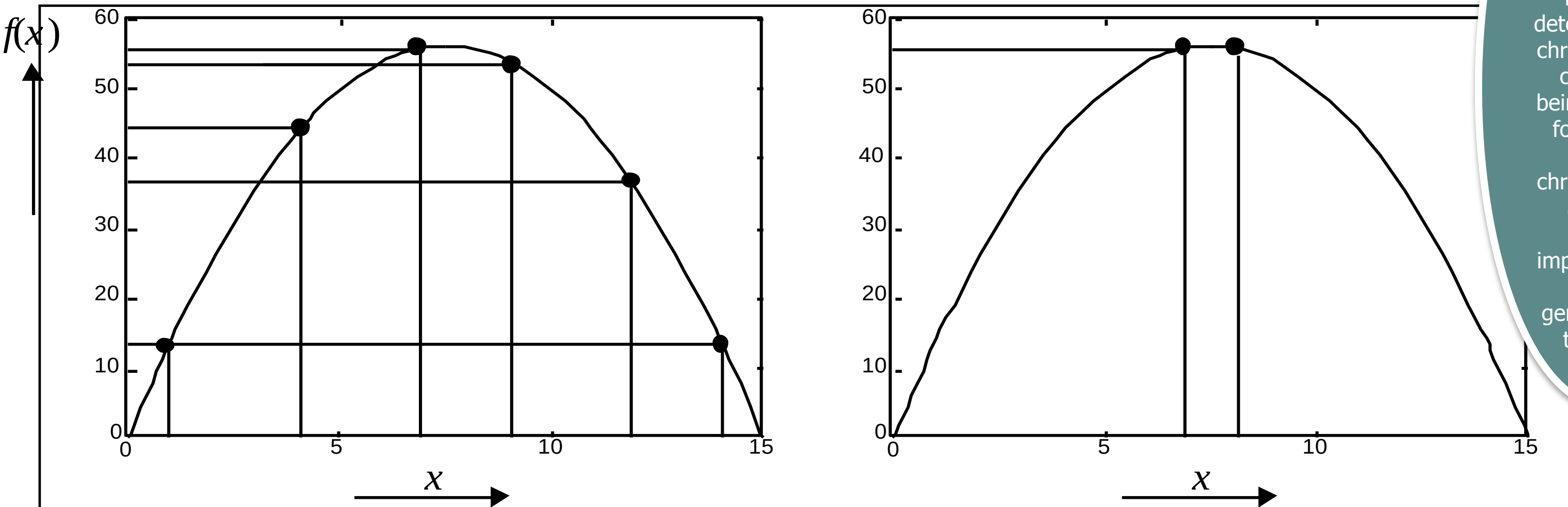
Suppose that the size of the chromosome population N is 6, the crossover probability p_c equals 0.7, and the mutation probability p_m equals 0.001. The fitness function in our example is defined by

$$f(x) = 15\,x - x^2$$

The fitness function and chromosome locations

Chromosome label	*Chromosome string*	*Decoded integer*	*Chromosome fitness*	*Fitness ratio, %*
X1	1 1 0 0	12	36	16.5
X2	0 1 0 0	4	44	20.2
X3	0 0 0 1	1	14	6.4
X4	1 1 1 0	14	14	6.4
X5	0 1 1 1	7	56	25.7
X6	1 0 0 1	9	54	24.8

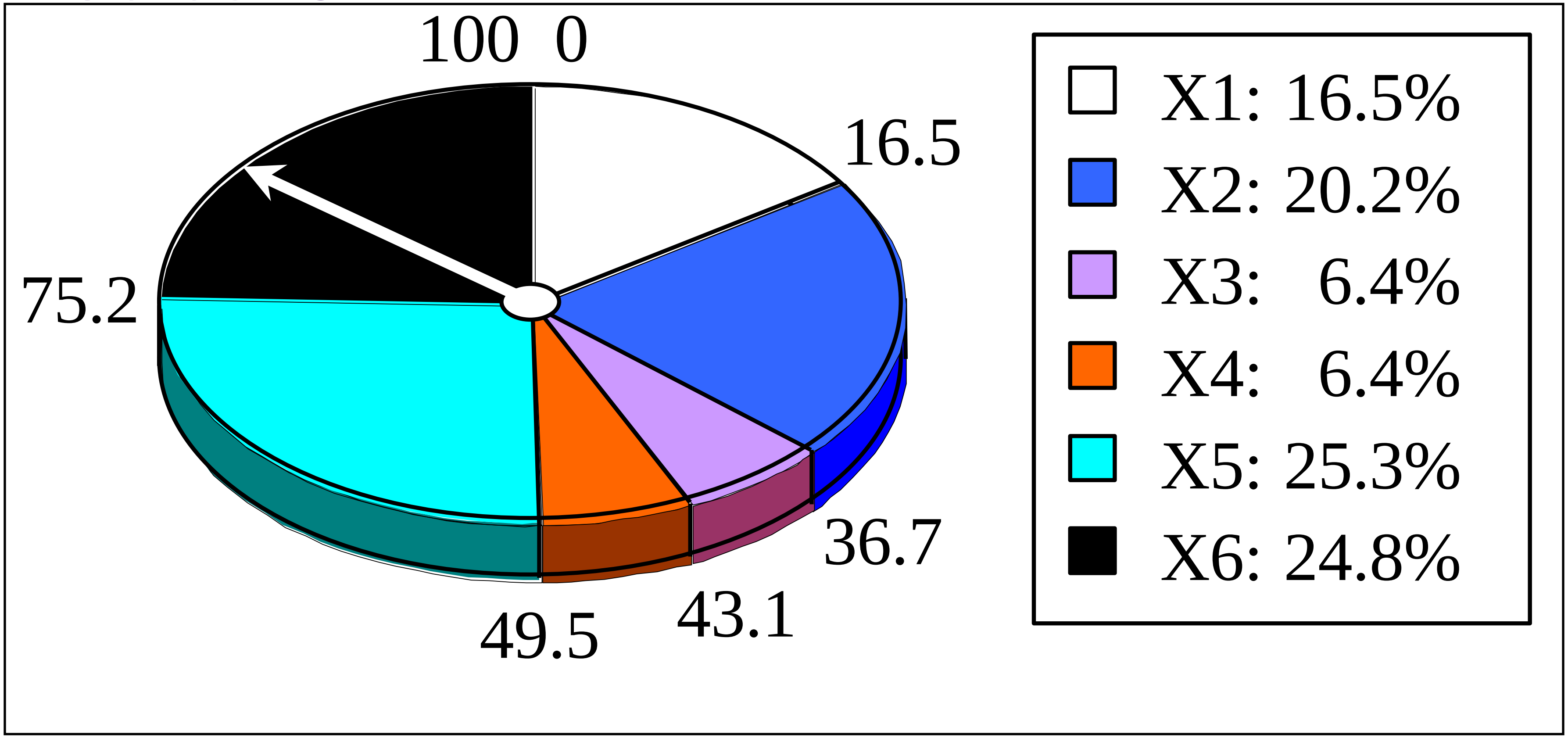
ratio of the individual chromosome's fitness to the population's total fitness. This ratio determines the chromosome's chance of being selected for mating. The chromosome's average fitness improves from one generation to the next.



(*a*) Chromosome initial locations. (*b*) Chromosome final locations.

Roulette wheel selection

The most commonly used chromosome selection techniques is the **roulette wheel selection**.

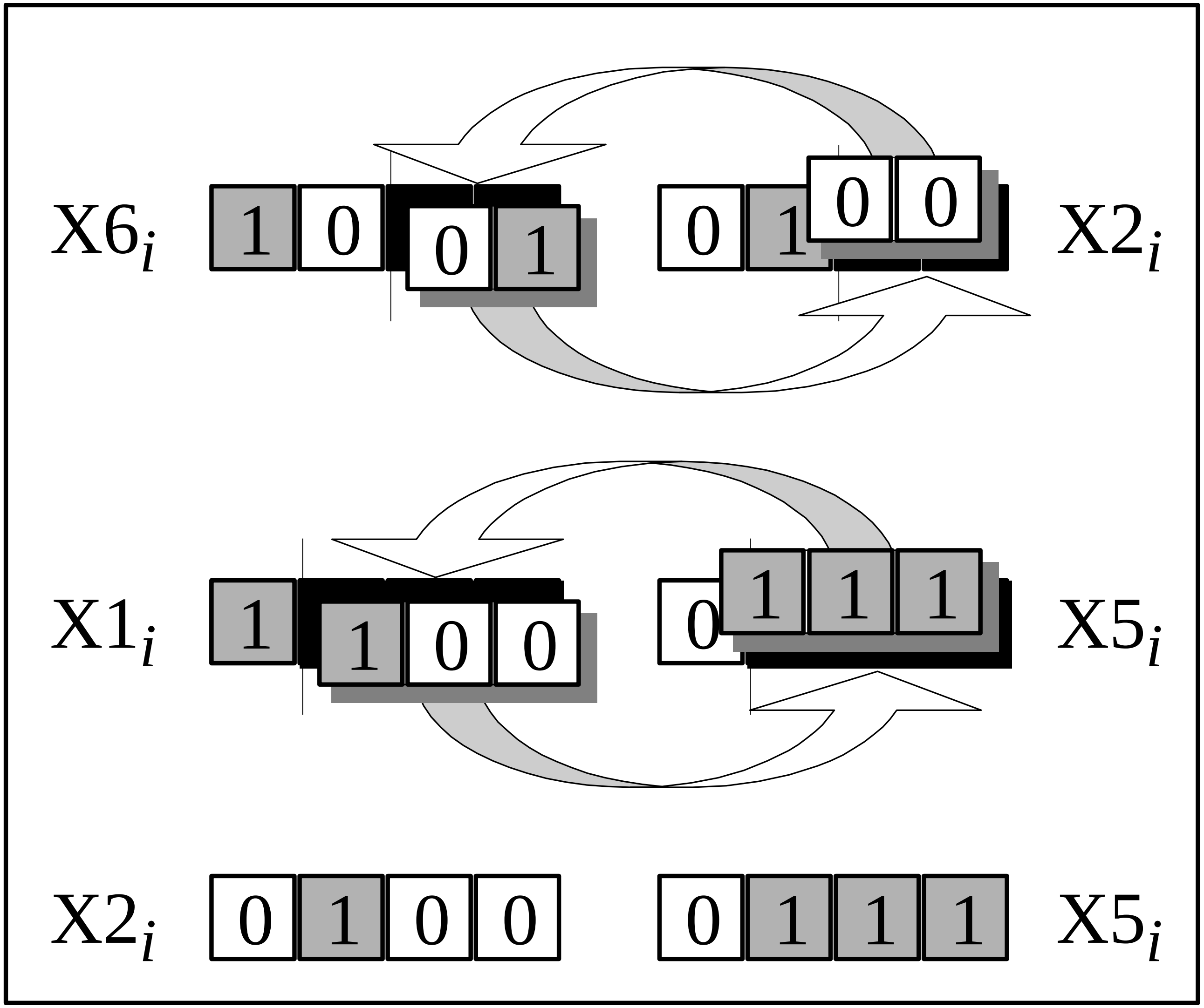


Crossover operator

- In our example, we have an initial population of 6 chromosomes. Thus, to establish the same population in the next generation, the roulette wheel would be spun six times.
- Once a pair of parent chromosomes is selected, the **crossover** operator is applied.

- First, the crossover operator randomly chooses a crossover point where two parent chromosomes “break”, and then exchanges the chromosome parts after that point. As a result, two new offspring are created.
- If a pair of chromosomes does not cross over, then the chromosome cloning takes place, and the offspring are created as exact copies of each parent.

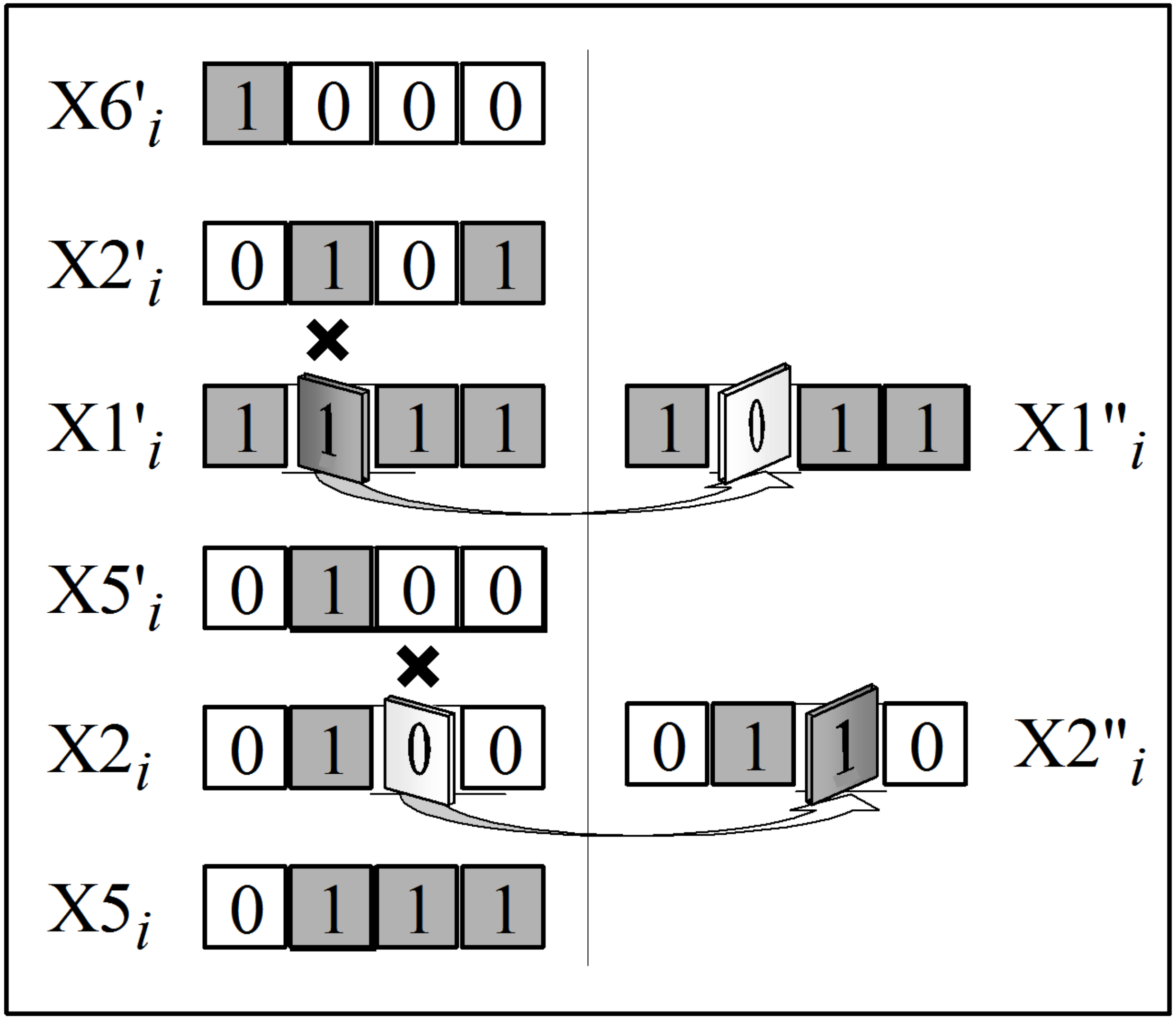
Crossover



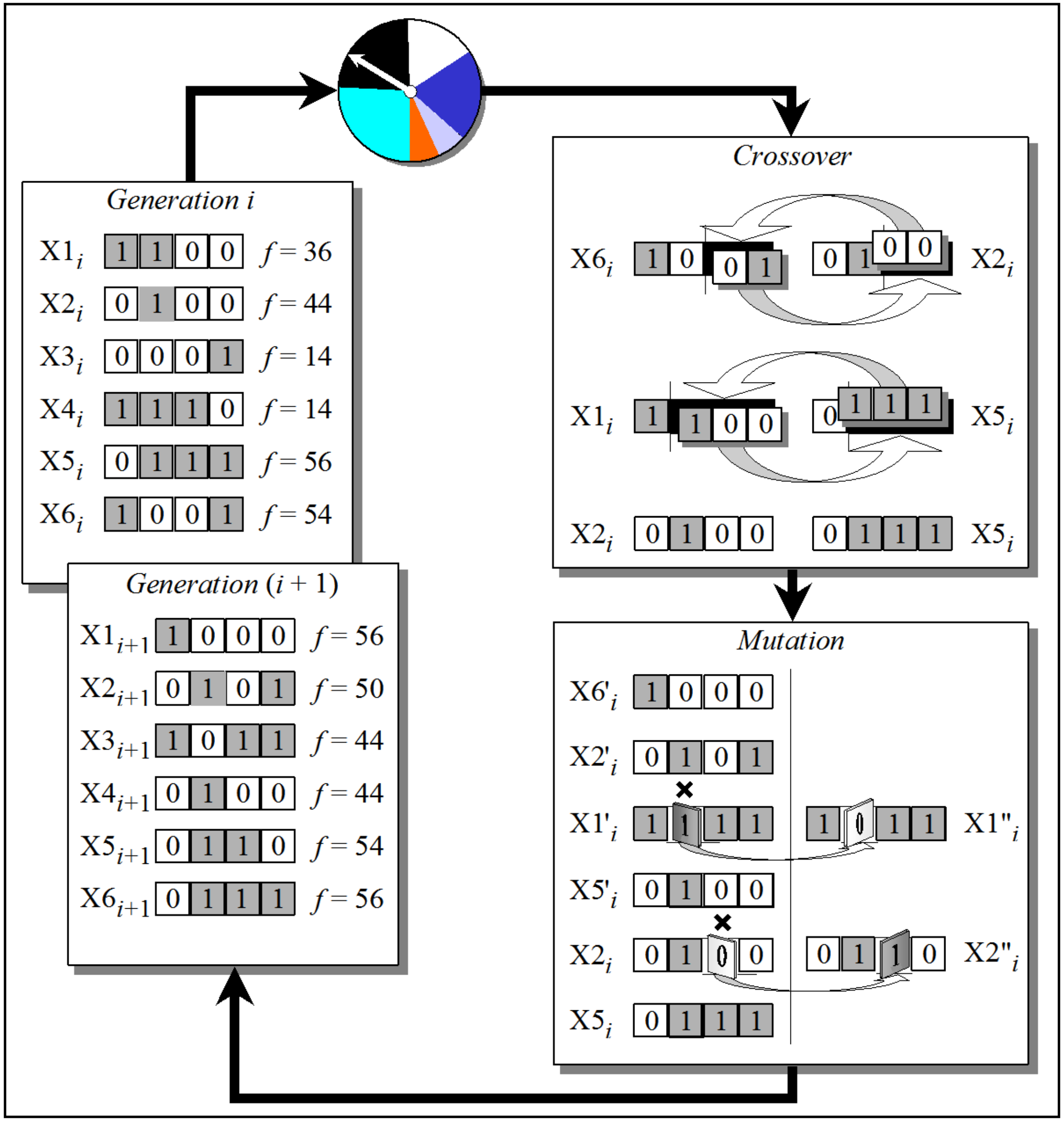
Mutation operator

- Mutation represents a change in the gene.
- Mutation is a background operator. Its role is to provide a guarantee that the search algorithm is not trapped on a local optimum.
- The mutation operator flips a randomly selected gene in a chromosome.
- The mutation probability is quite small in nature, and is kept low for GAs, typically in the range between 0.001 and 0.01.

Mutation


X6'$_i$
1 0 0 0
X2'$_i$
0 1 0 1
X1'$_i$
1 1 1 1
1 0 1 1
X1"$_i$
X5'$_i$
0 1 0 0
X2$_i$
0 1 0 0
0 1 1 0
X2"$_i$
X5$_i$
0 1 1 1

The genetic algorithm cycle



Useful Reading

1. Negnevitsky, "Artificial Intelligence: a Guide to Intelligent Systems", sections 7.1-7.3, 7.5-7.6.
2. Eiben A.E., Smith J.E. (2007), Introduction to Evolutionary Computing, Springer, Chapters 3, 4, 8.3.1. Available online at: https://docs.google.com/file/d/0By995HEqDrWQakRxNFc4OVVPQWc/edit?usp=sharing
3. Blickle, T. and Thiele, L.: A Comparison of Selection Schemes used in Genetic Algorithms (2. Edition). TIK Report No. 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, 1995. Online at: http://www.tik.ee.ethz.ch/file/6c0e384dceb283cd4301339a895b72b8/TIK-Report11.pdf
4. Introduction to Genetic Algorithms www.obitko.com/tutorials/genetic-algorithms/
5. Magoulas, G.D., Eldabi, T., and Paul R.J. Adaptive Stochastic Search Methods for Parameter Adaptation of Simulation Models, Proceedings of the IEEE International Symposium on Intelligent Systems, Varna, Bulgaria, 2002, vol. 2, 22-26. http://www.dcs.bbk.ac.uk/~gmagoulas/IEEE_IS02_P1029uk.pdf
6. Salimans T., Ho J., Chen X., Sidor S., Sutskever I (2017)., Evolution Strategies as a Scalable Alternative to Reinforcement Learning. Online at: https://openai.com/blog/evolution-strategies/

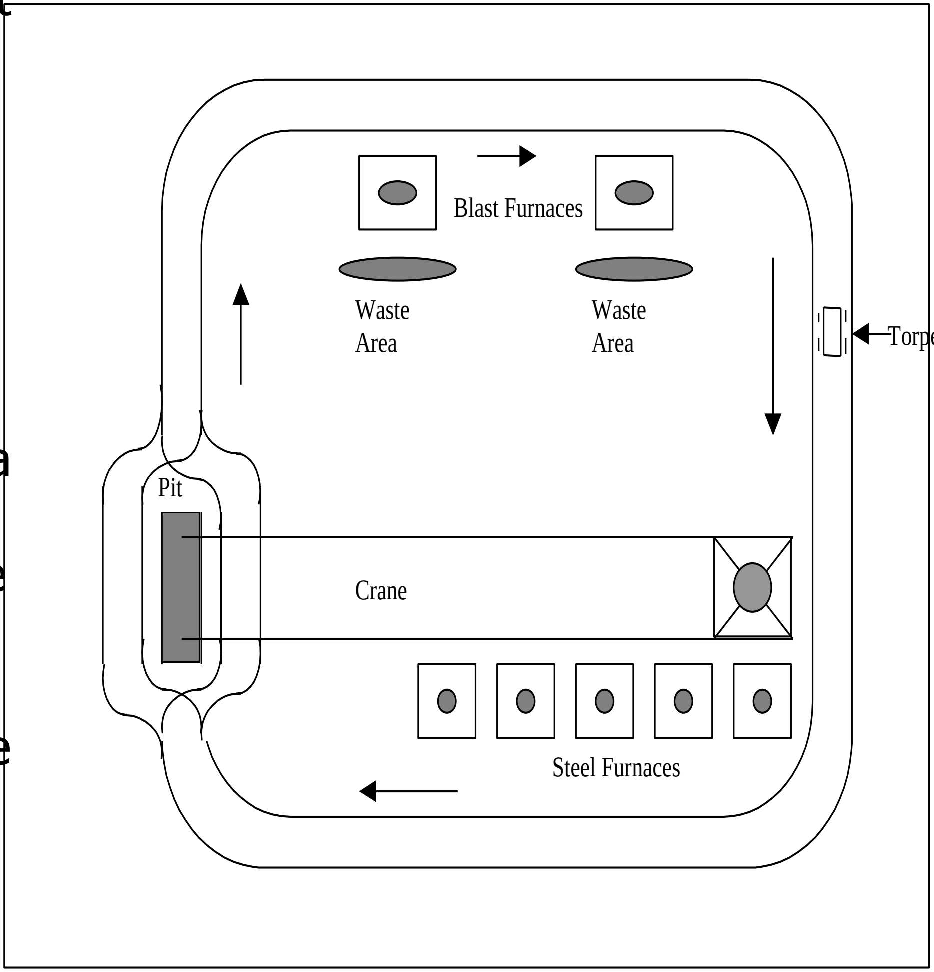
Next

Advanced learning and evolution

Appendix

A design problem: configuration of a steelworks plant

There are two blast furnaces, which melt iron at certain daily volumes, in a plant that blows and fills as many torpedoes as available; these are used to transport molten iron. If no torpedo is available, the molten iron is dropped on the floor and waste is produced. Each torpedo can hold a fixed quantity of molten iron. All torpedoes with molten iron travel to a pit, where cranes-carrying ladles are filled from torpedoes, one at a time. The ladle holds 100 tons of molten iron, which is exactly the volume of a steel furnace that is fed from the crane. There are five steel furnaces, which produce the final product of the steelworks



Variable	Variable Type	Range	Number of Possibilities
1. Number of Torpedoes	Integer	1-12	12
2. Number of Cranes	Integer	1-2	2
3. Number of Steel Furnaces	Integer	1-6	6
4. Volume of the torpedo (in tonnes)	Real	50-350	300

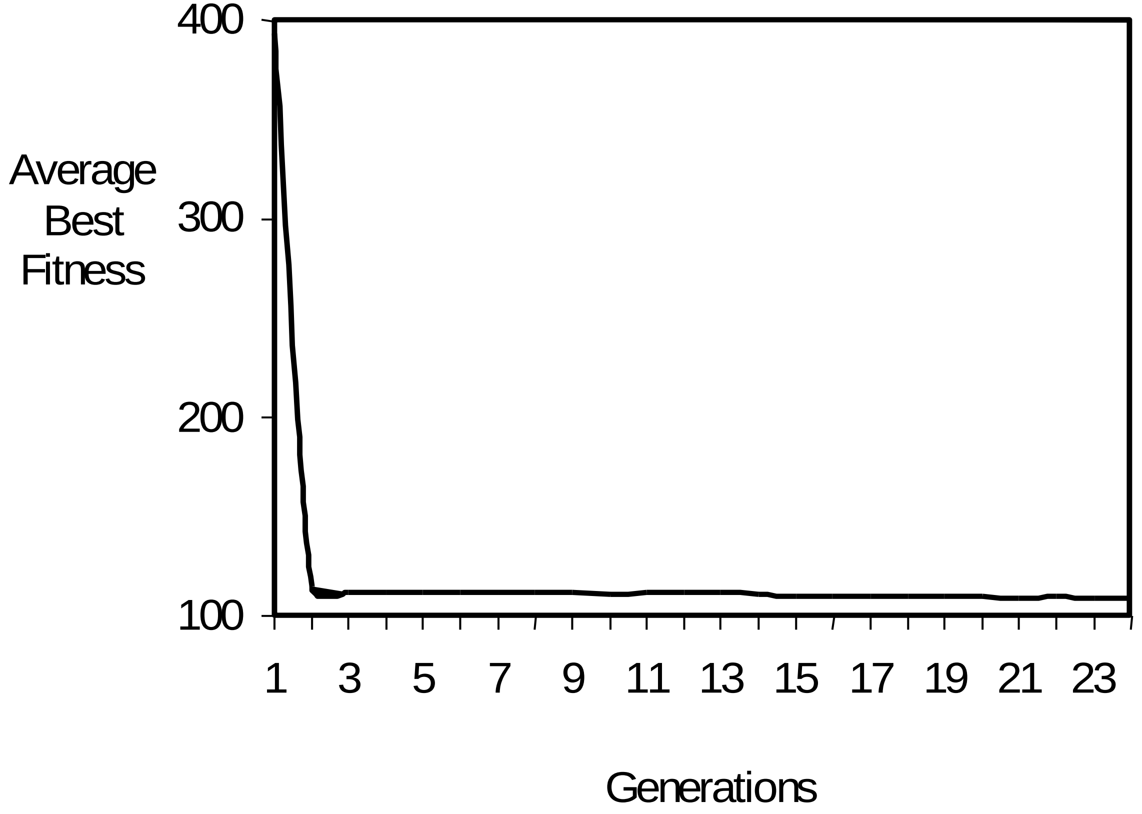
Table 1 - Range of component values

Total_Monthly_Cost = Investment_cost + waste_monthly_cost

TMC = 2.1*Nt + 8.3*Nc +16.7*Sf +Waste

Method	SA	GA
#Torpedo	6	4
#Cranes	2	2
#Furnaces	5	5
Torpedo Volume	260	235
Objective function (£K)	112.70	108.5

Comparative results for Simulated Annealing (SA) and Genetic algorithms (GA)



Average best fitness for a 20 member population GA

Simulated Annealing-1

- The name comes from the analogy to the behavior of physical systems by melting a substance and lowering its temperature slowly until it reaches freezing point (physical annealing).
- SA is based on random evaluations of the objective function, in such a way that transitions out of a local minimum are possible.

Simulated Annealing-2

Candidate solutions are updated following the relation

$$x_{k+1} = x_k + \Delta x, \qquad (1)$$

where Δx is random noise from a uniform distribution.

SA applies the *Metropolis* criterion, i.e. it either accepts or rejects a candidate solution depending on the probability

$$P(\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}) = \begin{cases} 1 & if \ \Delta f = f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) < 0 \\ e^{-\frac{\Delta f}{T}} & \text{otherwise} \end{cases}. \qquad (2)$$

Simulated Annealing-3

This, if the sign of $\Delta f = f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)$

is *negative*, then the new point can be accepted with probability 1; otherwise, it depends on the probability value and the threshold value θ

i.e. $P(\mathbf{x}_k \to \mathbf{x}_{k+1}) = \exp(-\Delta f / T) > \theta, \quad \theta \in (0,1)$

The effectiveness of the method depends the parameter *T* that is called temperature; it controls the noise reduction rate:

$$T(k) = \frac{T_0}{1 + \ln k} \qquad (3)$$

With high values of T, SA behaves like a random search
Low T values make it work like a hill climbing procedure
Start with a high temperature value and gradually reduce it