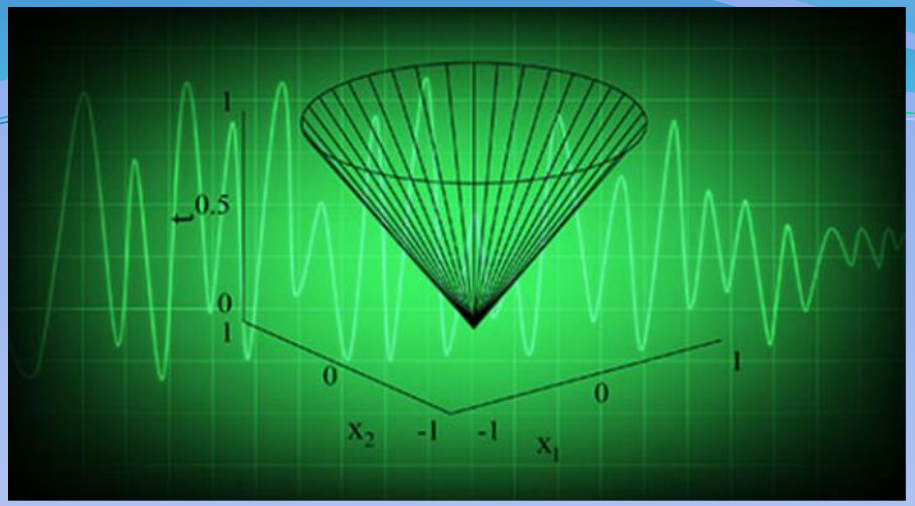
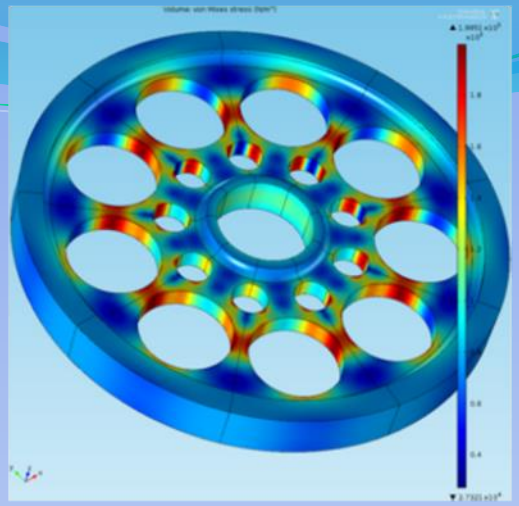




Advanced Optimization

Lecture 2

Shahryar Rahnamayan, PhD, PEng, SMIEEE

Department Chair and Associate Professor

NICI
Nature-Inspired Computational Intelligence

SIRC 3330

11:10 AM - 2:00 PM

Fri., Sept. 13, 2019

spring fws as

$$fws(x_1,x_2,x_3) = (x_3+2)x_2x_1^2 \tag{7}$$

where

A tension/compression coil spring

A sample real-world benchmark problem

$x_1 = d$ wire diameter, inch
$x_2 = D$ mean coil diameter, inch
$x_3 = N$ number of active coils

subject to the following four constraints

$$g_1(x) = 1.0 - \frac{x_2^3 x_3}{71875x_1^4} \le 0 \quad \text{deflection constraint} \tag{8}$$

$$g_2(x) = \frac{x_2(4x_2 - x_1)}{12566x_1^3(x_2 - x_1)} + \frac{2.46}{12566x_1^2} - 1.0 \le 0 \quad \text{stress constraint}$$

$$g_3(x) = 1.0 - \frac{140.54x_1}{x_2^2 x_3} \le 0 \quad \text{surge wave frequency constraint}$$

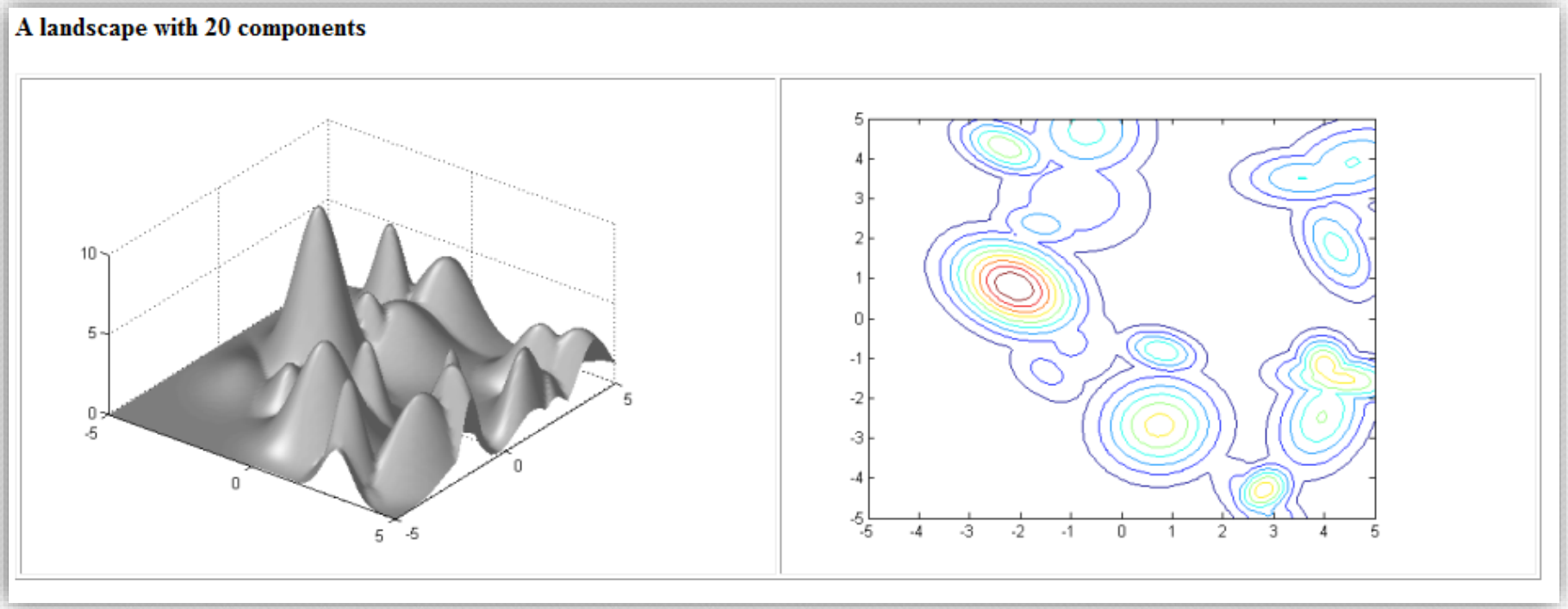
$$g_4(x) = \frac{x_2 + x_1}{1.5} - 1.0 \le 0 \quad \text{outer diameter constraint}$$

with the following lower and upper bounds on design variables:

$$0.05 \le x_1 \le 0.20 \tag{9}$$
$$0.25 \le x_2 \le 1.30$$
$$2 \le x_3 \le 15$$

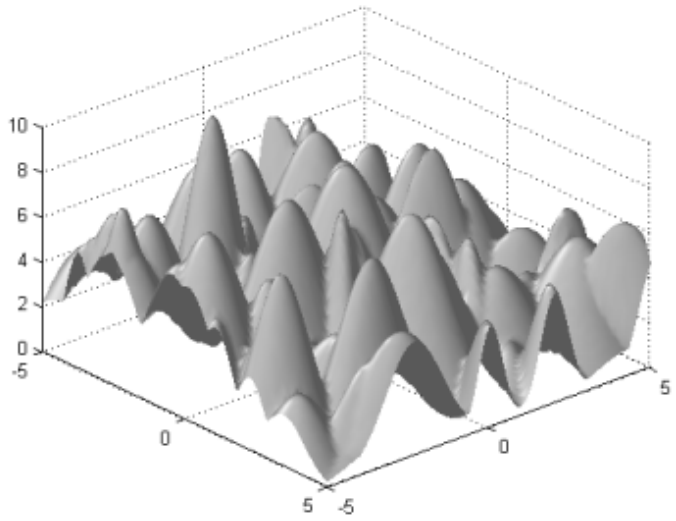
Why do we use benchmark functions?

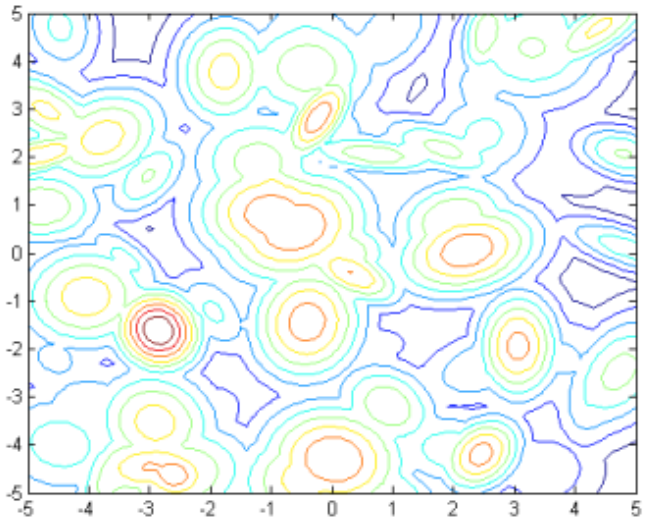
Gaussians Landscape Generator



http://itee.uq.edu.au/~marcusg/msg.html

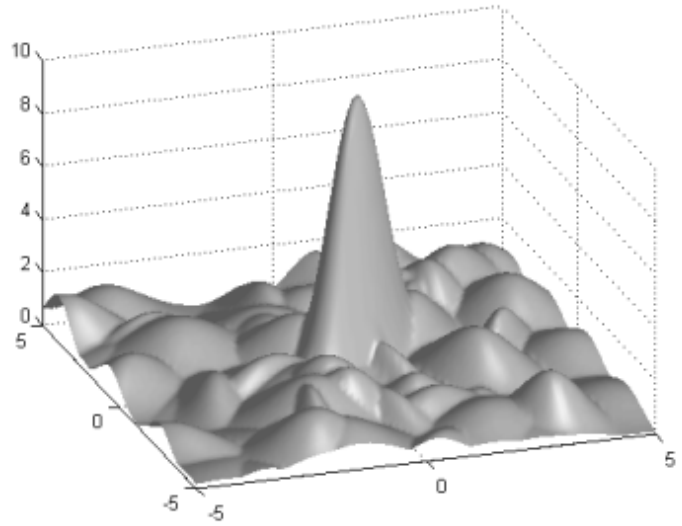
A landscape with 100 components

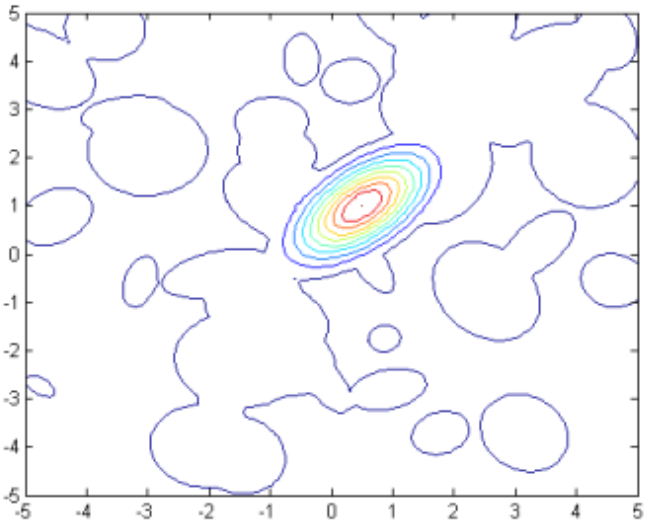




http://itee.uq.edu.au/~marcusg/msg.html

A landscape where the global optimum is significantly better than local optima.





http://itee.uq.edu.au/~marcusg/msg.html

Landscape generators
Vs.
Benchmark functions

Assignment One

You need to implement the following 12 benchmark functions in MATLAB, for each function also you need to plot 3-D map for 2-D version of that. Please use the "Publish" component in MATLAB after finishing all your implementations and plotting, this component generates a report including your codes also results (i.e., plots) as a PDF file. You need to submit the report which is created by the "Publish" component. Add your group members' name and student number to the report as a comment.

Note: Your code should be well-commented.

Implementation Assignment (1): Implementing in Matlab, and plotting 3D map

Search range: $[-100,100]^D$

Summary of the CEC'14 Test Functions

	No.	Functions	$F_i^*=F_i(x^*)$
Unimodal Functions	1	Rotated High Conditioned Elliptic Function	100
	2	Rotated Bent Cigar Function	200
	3	Rotated Discus Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Ackley's Function	500
	6	Shifted and Rotated Weierstrass Function	600
	7	Shifted and Rotated Griewank's Function	700
	8	Shifted Rastrigin's Function	800
	9	Shifted and Rotated Rastrigin's Function	900
	10	Shifted Schwefel's Function	1000
	11	Shifted and Rotated Schwefel's Function	1100
	12	Shifted and Rotated Katsuura Function	1200

High Conditioned Elliptic Function

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} \boldsymbol{x}_i^2$$

Bent Cigar Function

$$f_2(\boldsymbol{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$$

Discus Function

$$f_3(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$$

Rosenbrock's Function

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

Ackley's Function

$$f_5(\boldsymbol{x}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

Weierstrass Function

$$f_6(\boldsymbol{x}) = \sum_{i=1}^{D}(\sum_{k=0}^{k\max}[a^k \cos(2\pi b^k (x_i + 0.5))]) - D\sum_{k=0}^{k\max}[a^k \cos(2\pi b^k \cdot 0.5)]$$

a=0.5, b=3, $kmax$=20

Griewank's Function

$$f_7(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Rastrigin's Function

$$f_8(\boldsymbol{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$$

Modified Schwefel's Function

$$f_9(\boldsymbol{x}) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i), \qquad z_i = x_i + 4.209687462275036e+002$$

$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}) & \text{if } |z_i| \le 500 \\ (500 - \text{mod}(z_i, 500)) \sin(\sqrt{|500 - \text{mod}(z_i, 500)|}) - \dfrac{(z_i - 500)^2}{10000D} & \text{if } z_i > 500 \\ (\text{mod}(|z_i|, 500) - 500) \sin(\sqrt{|\text{mod}(|z_i|, 500) - 500|}) - \dfrac{(z_i + 500)^2}{10000D} & \text{if } z_i < -500 \end{cases}$$

Katsuura Function

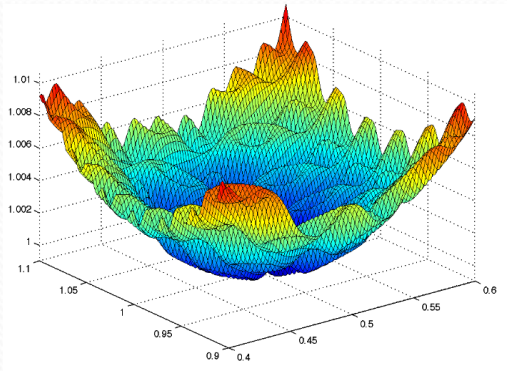
$$f_{10}(\boldsymbol{x}) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{|2^j x_i - round(2^j x_i)|}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2}$$

HappyCat Function

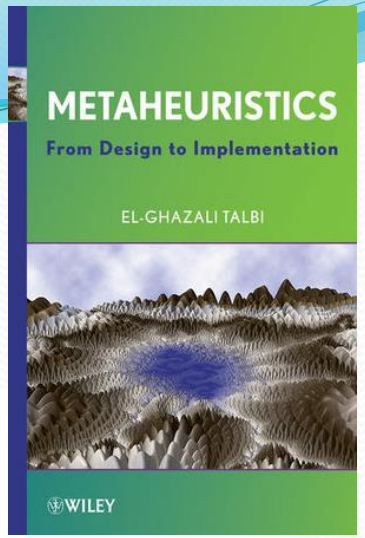
$$f_{11}(\boldsymbol{x}) = \left| \sum_{i=1}^{D} x_i^2 - D \right|^{1/4} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i) / D + 0.5$$

HGBat Function

$$f_{12}(\boldsymbol{x}) = \left| (\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2 \right|^{1/2} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i) / D + 0.5$$



Textbook's Organization



Chapter 1
Common concepts for metaheuristics

Chapter 2
Single-solution based metaheuristics

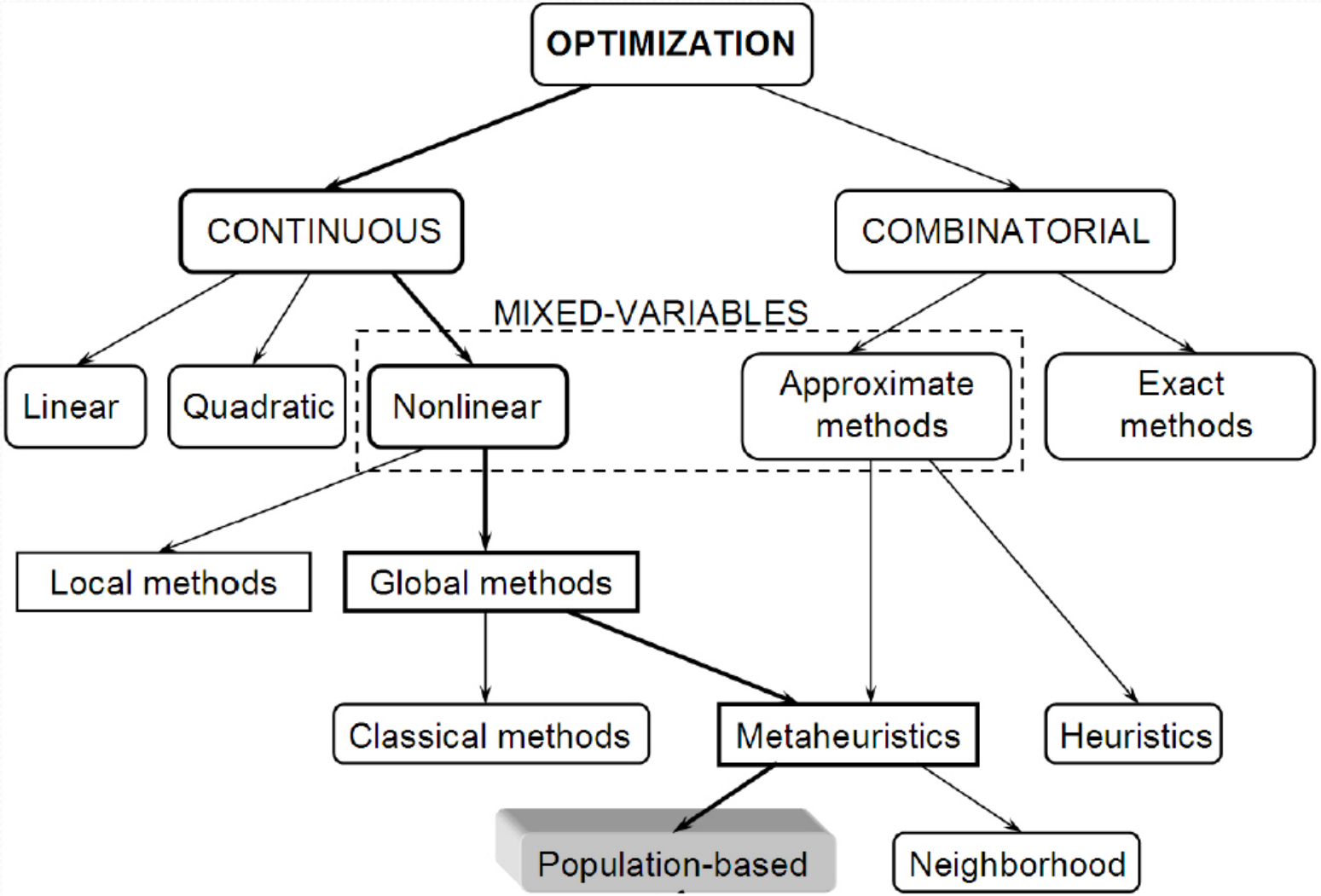
Chapter 3
Population-based metaheuristics

Chapter 4
Metaheuristics for multi-objective optimization

Chapter 5
Hybrid metaheuristics

Chapter 6
Parallel metaheuristics

Optimization Methods Classifications



[Feoktistov 2006]

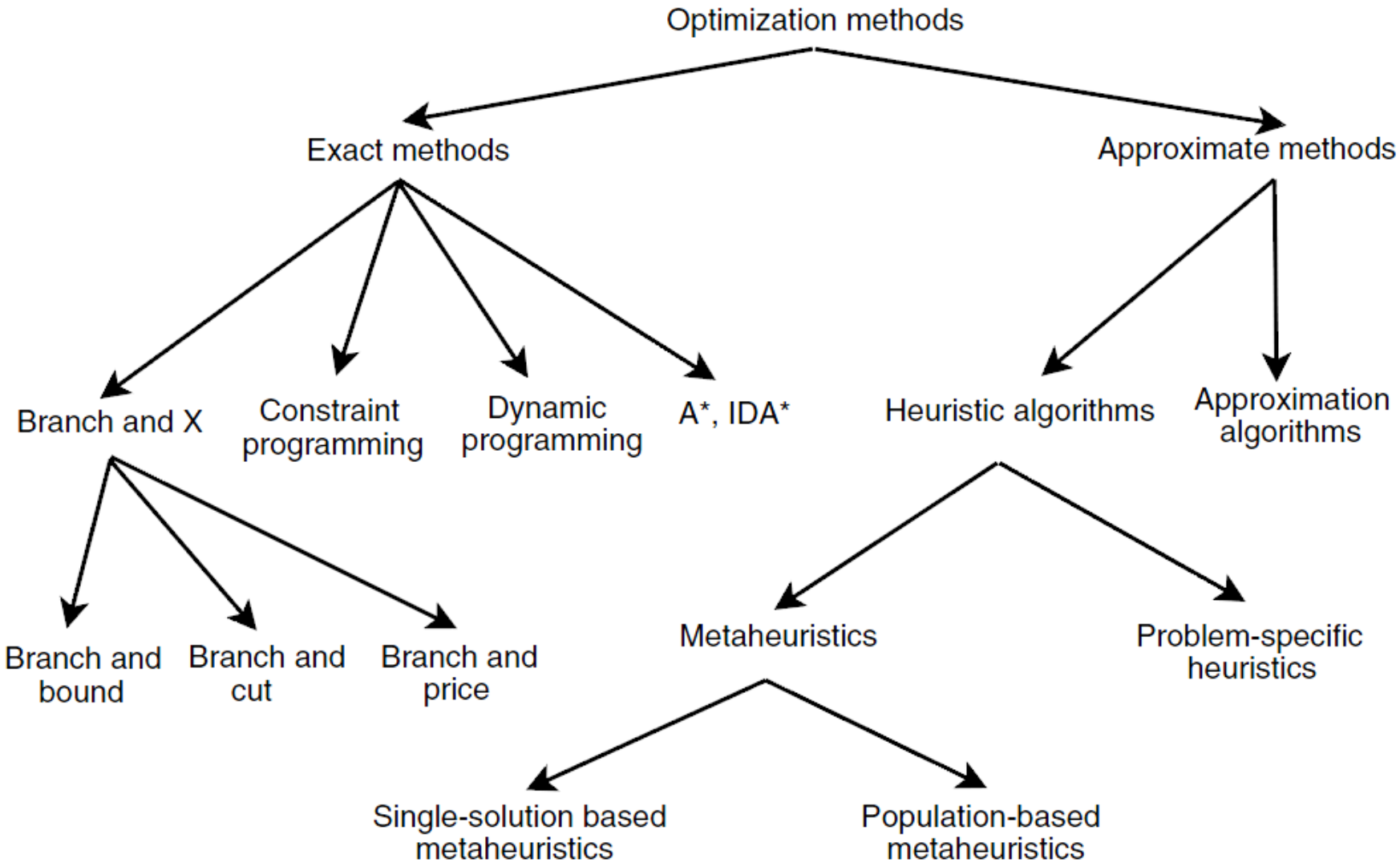
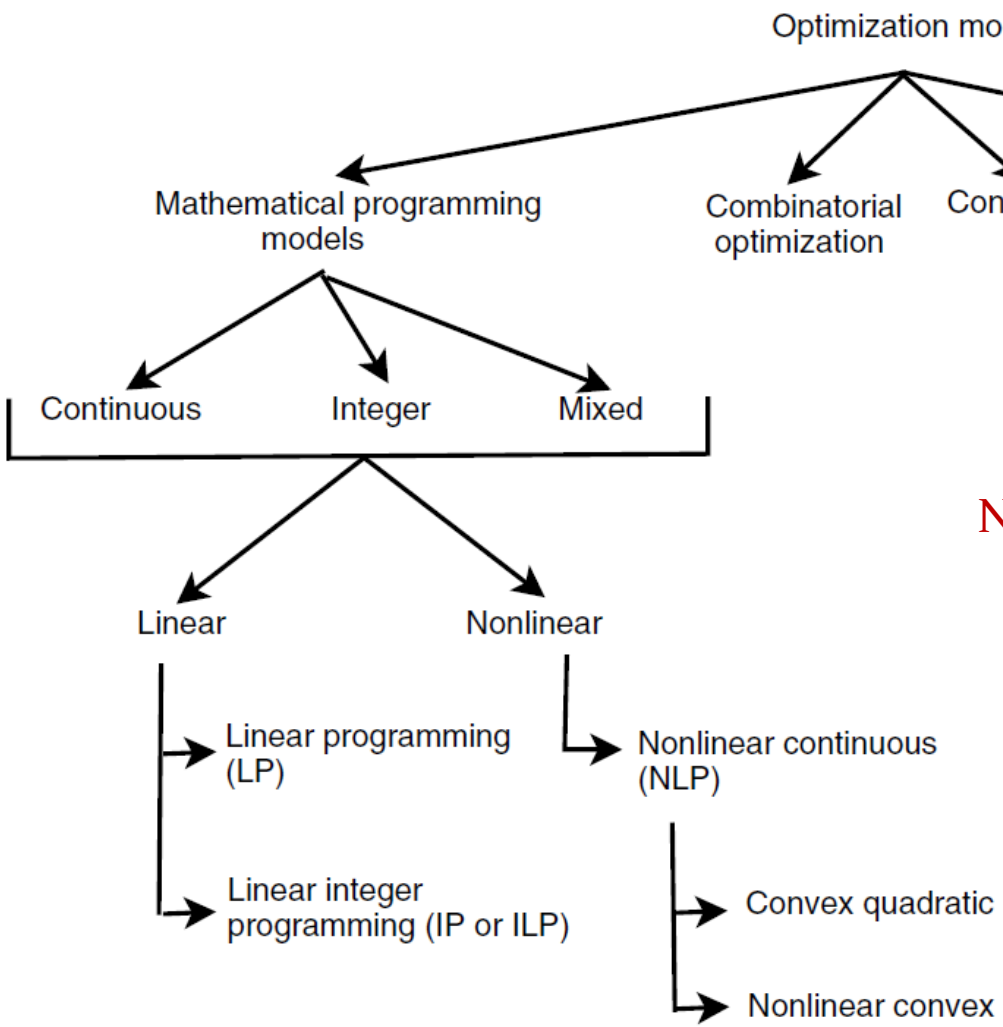


FIGURE 1.7 Classical optimization methods.

Classical optimization models



Non-linear and non-convex?

Optimization Methods

- Exact Methods: Guarantees the optimality

- Approximate Methods: They don't guarantee the optimality but they give a boundary estimation (closeness to the optimal solution)

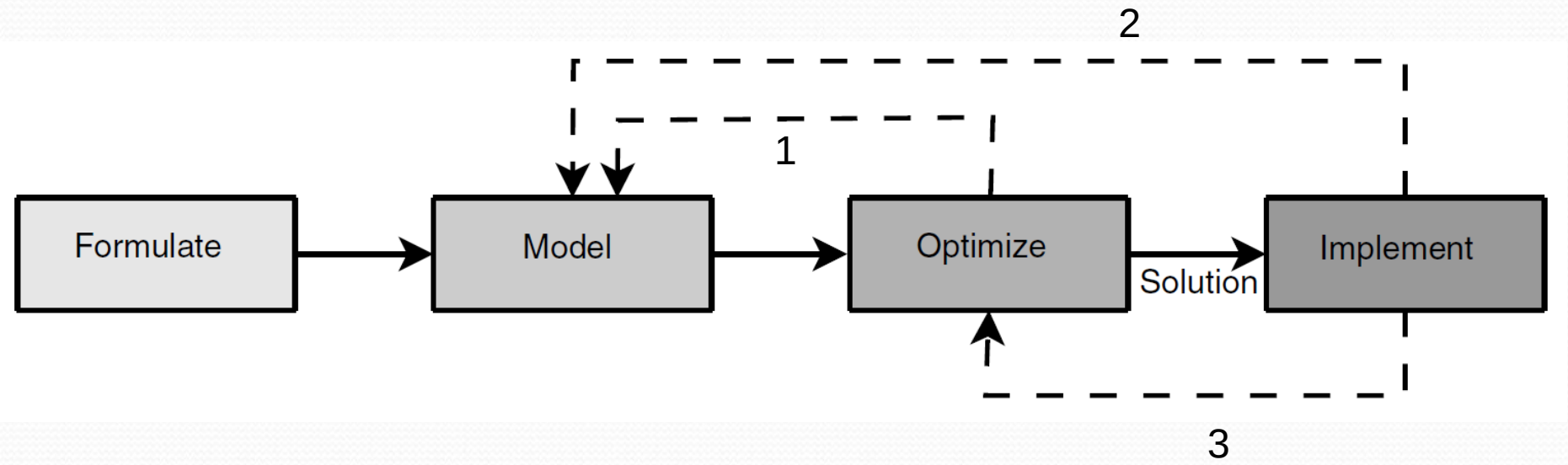
- Metaheuristics: They obtain "acceptable" solutions but with not a boundary estimation

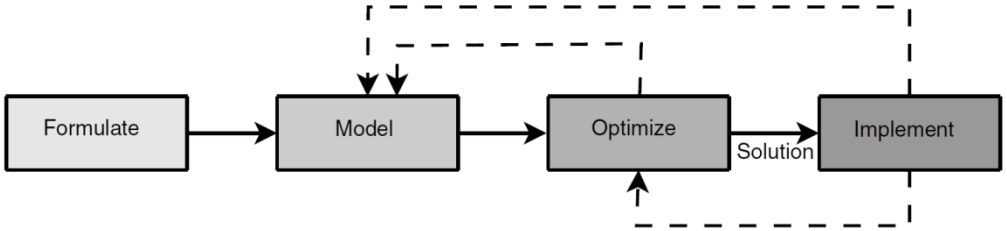
Metaheuristics

- *Origin in the old Greek word heuriskein, which means* the art of discovering new strategies (rules) to solve problems. The suffix *meta, also* a Greek word, means "upper level methodology."

- Metaheuristic search methods can be defined as upper level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems.

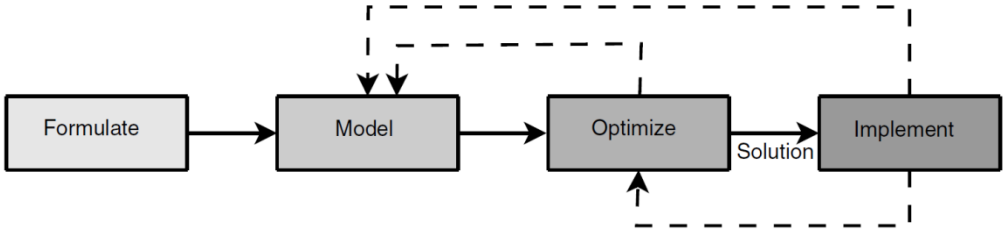
Optimization Procedure





Formulate the problem

- A decision problem is identified.
- An initial statement of the problem is made.
- This formulation may be imprecise.
- The internal and external factors and the objective(s) of the problem are outlined.
- Many decision makers may be involved in formulating the problem.



Model the problem

- In this important step, an abstract mathematical model is built for the problem (if possible).
- The modeler can be inspired by similar models in the literature. This will reduce the problem to well-studied optimization models.
- Usually, models we are solving are simplifications of the reality. They involve approximations and sometimes they skip processes that are complex to represent in a mathematical model (see next slide).

Assignments two

Assignment Two

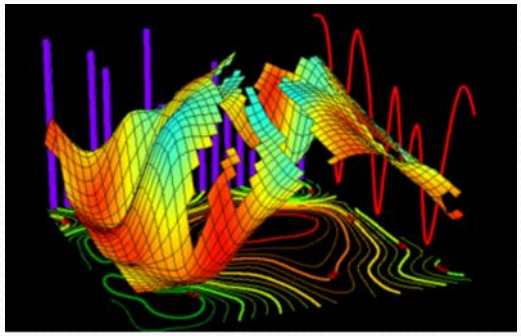
From the literature, find and explain six optimization problems, which are related to your research field (thesis) or interest. Problems can be in various fields. For each problem, you need to present the complete model of optimization problem, including, a) the problem statement, b) objective function(s), c) decision variables, d) constraints, and e) type of problem (Discrete, Continuous, Dynamic, ...), f) reference.

BQ. A highly critical question:

Exact solution for approximate model or approximate solution for exact model?

NFL: No free lunch theorem

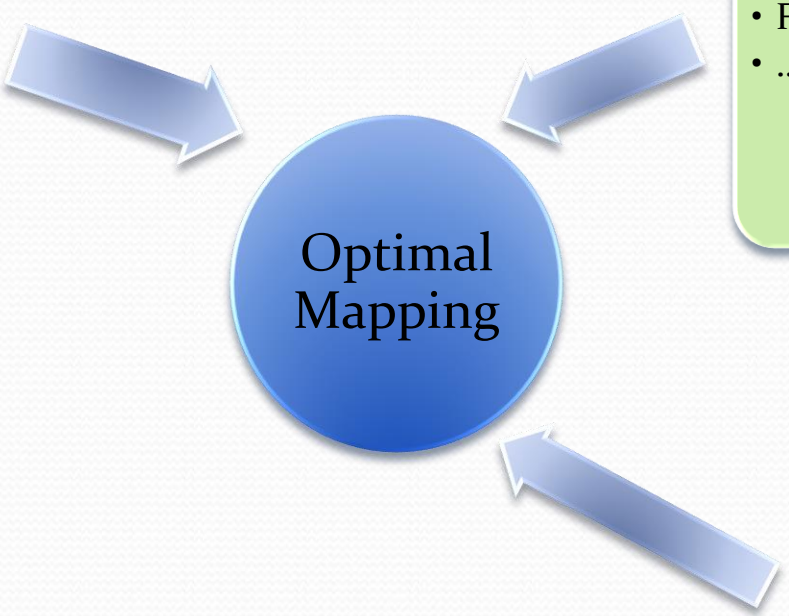
"Any two optimization algorithms are equivalent when their performance is averaged across all possible problems", D. Wolpert and W. Macready, 1997



NFL and Optimal Mapping Challenge

Type of Problem

- Single/multi Objective
- Uni-/multi Modal
- Noisy/Noise-free
- Large-Scale
- Expensive
- Dynamic
- ....



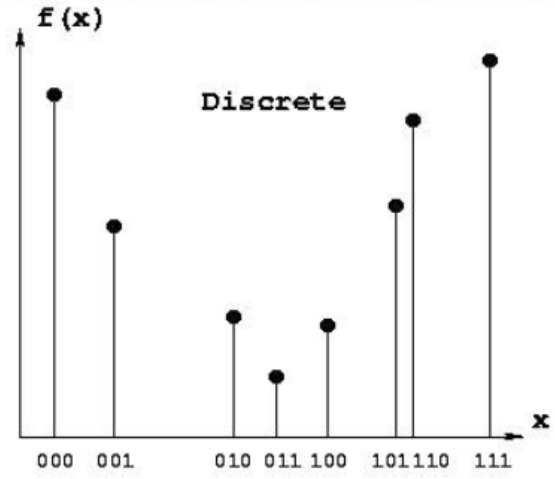
Goal

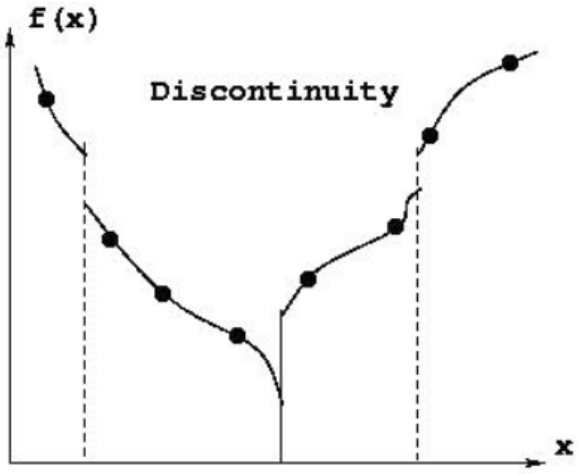
- Find first solution fast
- Find more solutions
- Find robust solution
- ....

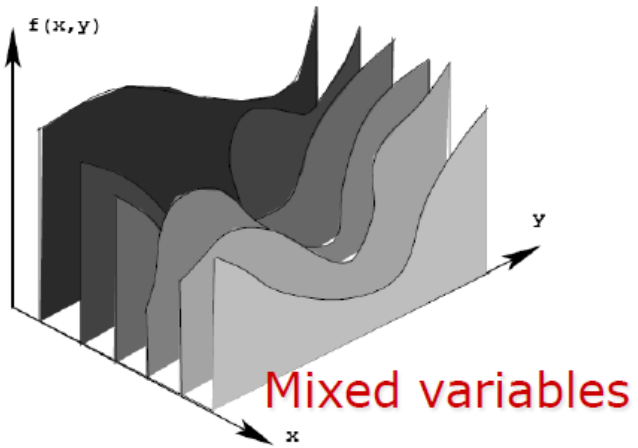
Type of Algorithm

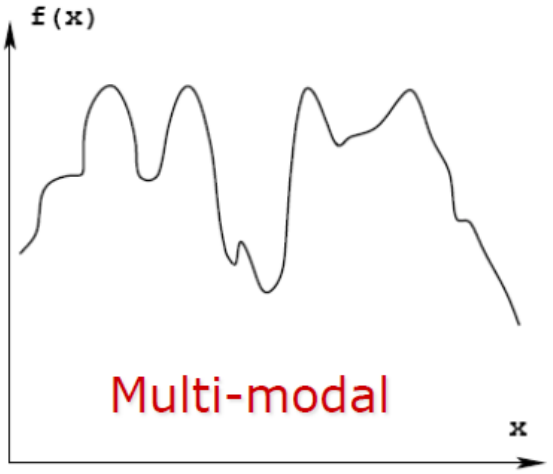
- **Classical methods**
- **GA**
- **PSO**
- **DE**
- **...**

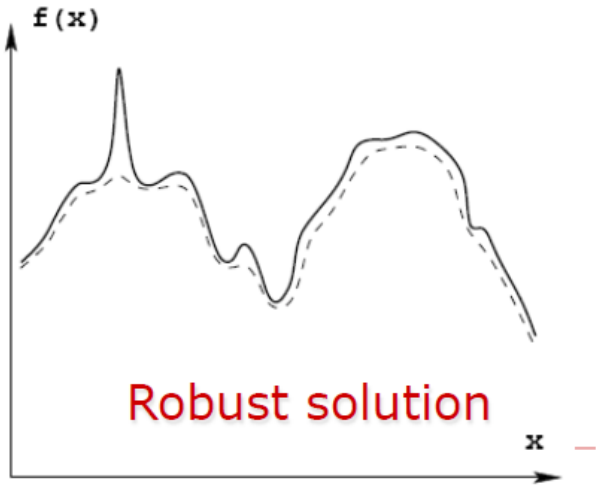
Different Problem Complexities

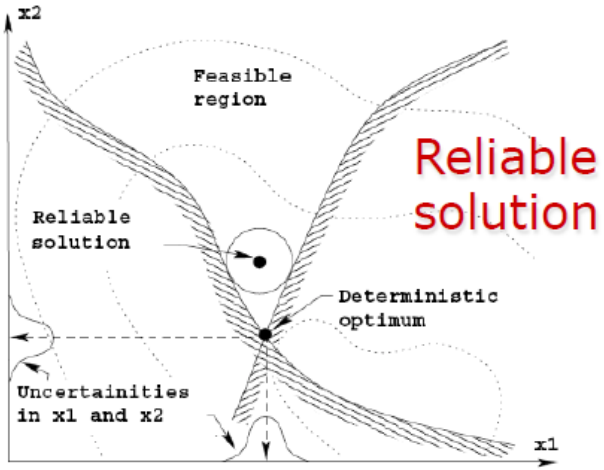






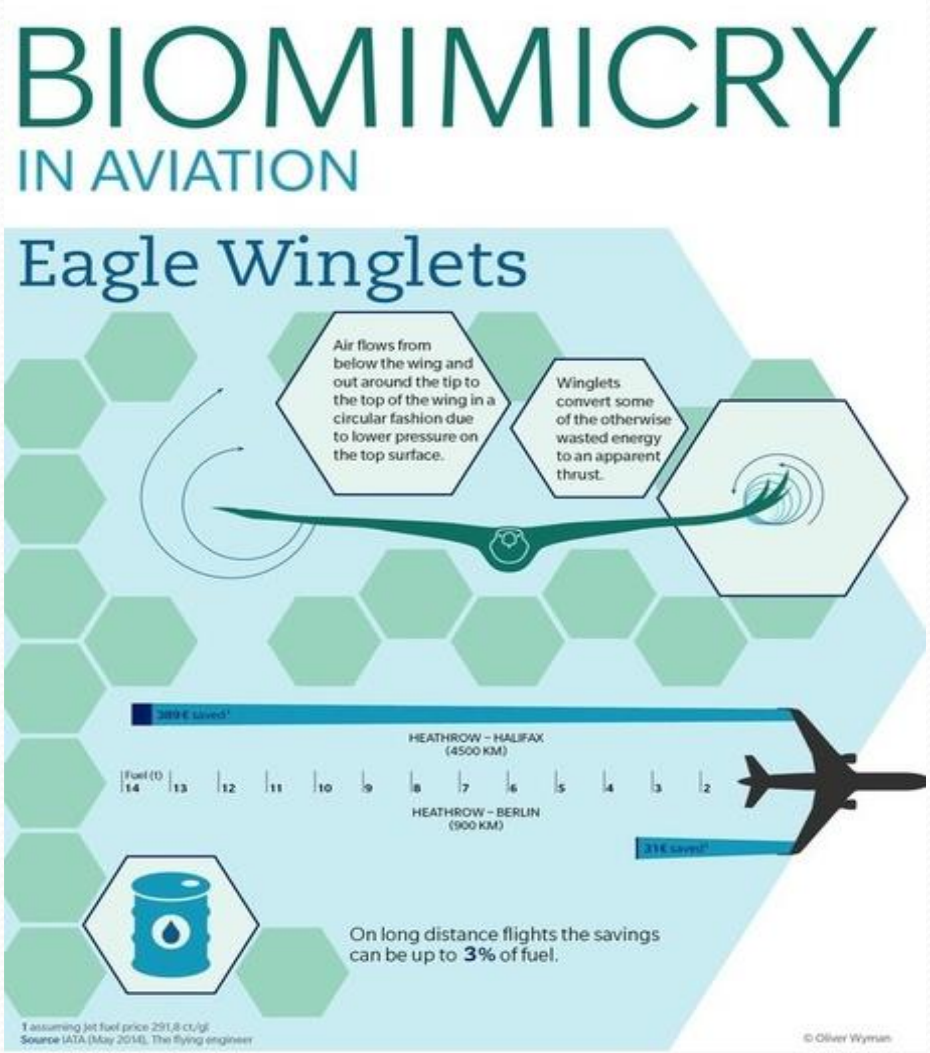






Eagle's Wings Inspire More Fuel Efficient Planes

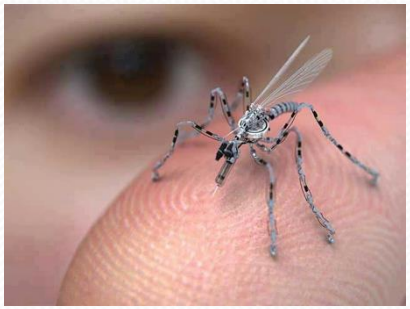
Scoop.it!



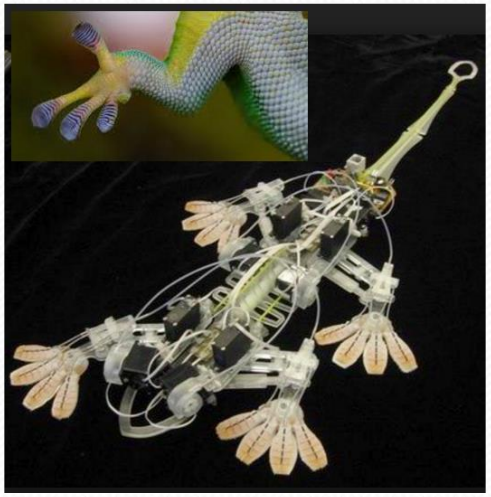
From www.adafruit.com - November 3, 2014 3:24 AM

"[...] The wing tips of steppe eagles are an ideal shape to maximize lift with a minimal wingspan. The curvature at the end of the wing reduces drag. Engineers designing the A380 copied that design, resulting in fuel savings of up to 3%, depending on if it is a long or short distance flight."

Nature Inspired Intelligent Techniques for Problem Solving



DNA-based circuits



Reinforcement Learning (RL)

Digital Evolution

Nanoshapes obtained using the DNA origami

Partial Swarm Intelligence

Artificial Neural Networks (ANN)

Robotic Design

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.

Can evolution be intelligent?

- Intelligence can be defined as the capability of a system to adapt its behavior to ever-changing environment. According to Alan Turing, the form or appearance of a system is irrelevant to its intelligence.

- Evolutionary computation simulates evolution on a computer. The result of such a simulation is a series of optimization algorithms, usually based on a simple set of rules. Optimization 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.

Simulation of natural evolution

- In July 1, 1858, Charles Darwin presented his theory of evolution. 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.

- Neo-Darwinism is based on processes of reproduction, mutation, competition and selection. The power to reproduce appears to be an essential property of life. The power to mutate is also guaranteed in any living organism that reproduces itself in a continuously changing environment.
- Processes of competition and selection normally take place in the natural world, where expanding populations of different species are limited by a finite space.

- 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 optimized in natural life.

- Genetic Algorithm (GA) [2:56]

Prof. Richard Lenski

- Experimental Evolution

Genetic Algorithms

- In the early 1970s, John Holland introduced the concept of genetic algorithms.
- His aim was to make computers do what nature does. 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

- Nature has an ability to adapt and learn without being told what to do. In other words, nature finds good chromosomes blindly. GAs do the same. 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 genetic algorithms

Step 1: Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population N, the crossover probability p_c and the mutation probability p_m.

Step 2: Define a fitness function to measure the performance, or fitness, of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

Step 3: Randomly generate an initial population of chromosomes of size N:

$x_1, x_2, \ldots, x_N$

Step 4: Calculate the fitness of each individual chromosome:

$f(x_1), f(x_2), \ldots, f(x_N)$

Step 5: Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness.

Step 6: Create a pair of offspring chromosomes by applying the genetic operators - crossover and mutation.

Step 7: Place the created offspring chromosomes in the new population.

Step 8: Repeat *Step 5* until the size of the new chromosome population becomes equal to the size of the initial population, *N*.

Step 9: Replace the initial (parent) chromosome population with the new (offspring) population.

Step 10: Go to *Step 4*, and repeat the process until the termination criterion is satisfied.

Genetic algorithms

- GA represents an iterative process. Each iteration is called a generation.
- Because GAs use a stochastic search method, the fitness of a population may remain stable for a number of generations before a superior chromosome appears.
- A common practice is to terminate a GA after a specified number of generations and then examine the best chromosomes in the population. If no satisfactory solution is found, the GA is restarted.

Genetic algorithms - evolution of a 2D car in Unity [5:41]

AI Simulation of Fish Tank (Genetic algorithm with Neural networks)[11:14]

Genetic Algorithm (GA): Three case studies

Genetic algorithms: Case study I

A simple example will help us to understand how a GA works. Let us find the maximum value of the function ($15x - x^2$) where parameter x varies between 0 and 15. For simplicity, we may assume that x takes only integer values. Thus, chromosomes can be built with only four genes:

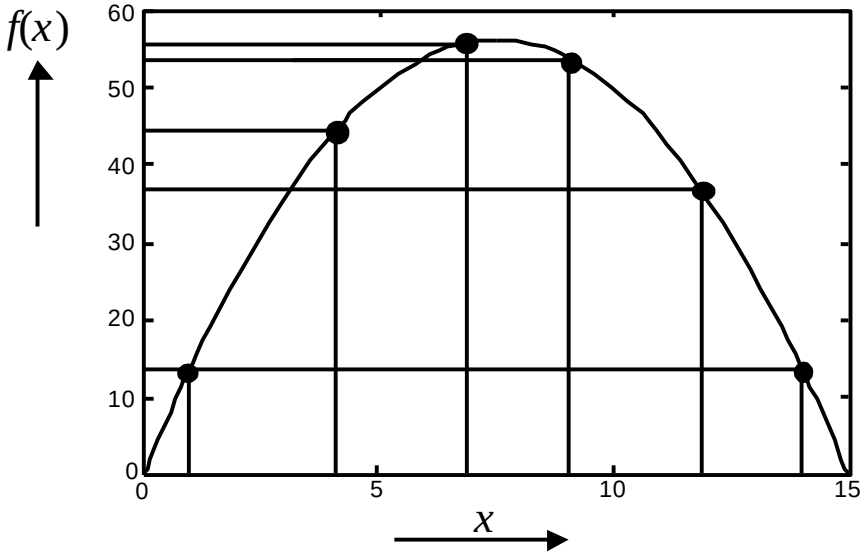
Integer	*Binary code*	*Integer*	*Binary code*	*Integer*	*Binary code*
1	0 0 0 1	6	0 1 1 0	11	1 0 1 1
2	0 0 1 0	7	0 1 1 1	12	1 1 0 0
3	0 0 1 1	8	1 0 0 0	13	1 1 0 1
4	0 1 0 0	9	1 0 0 1	14	1 1 1 0
5	0 1 0 1	10	1 0 1 0	15	1 1 1 1

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

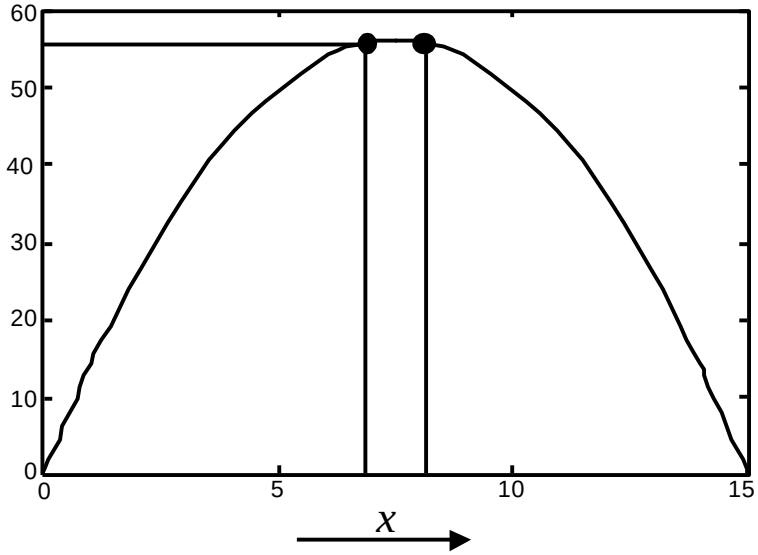
$$max\ f(x) = 15\ x - x^2$$

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



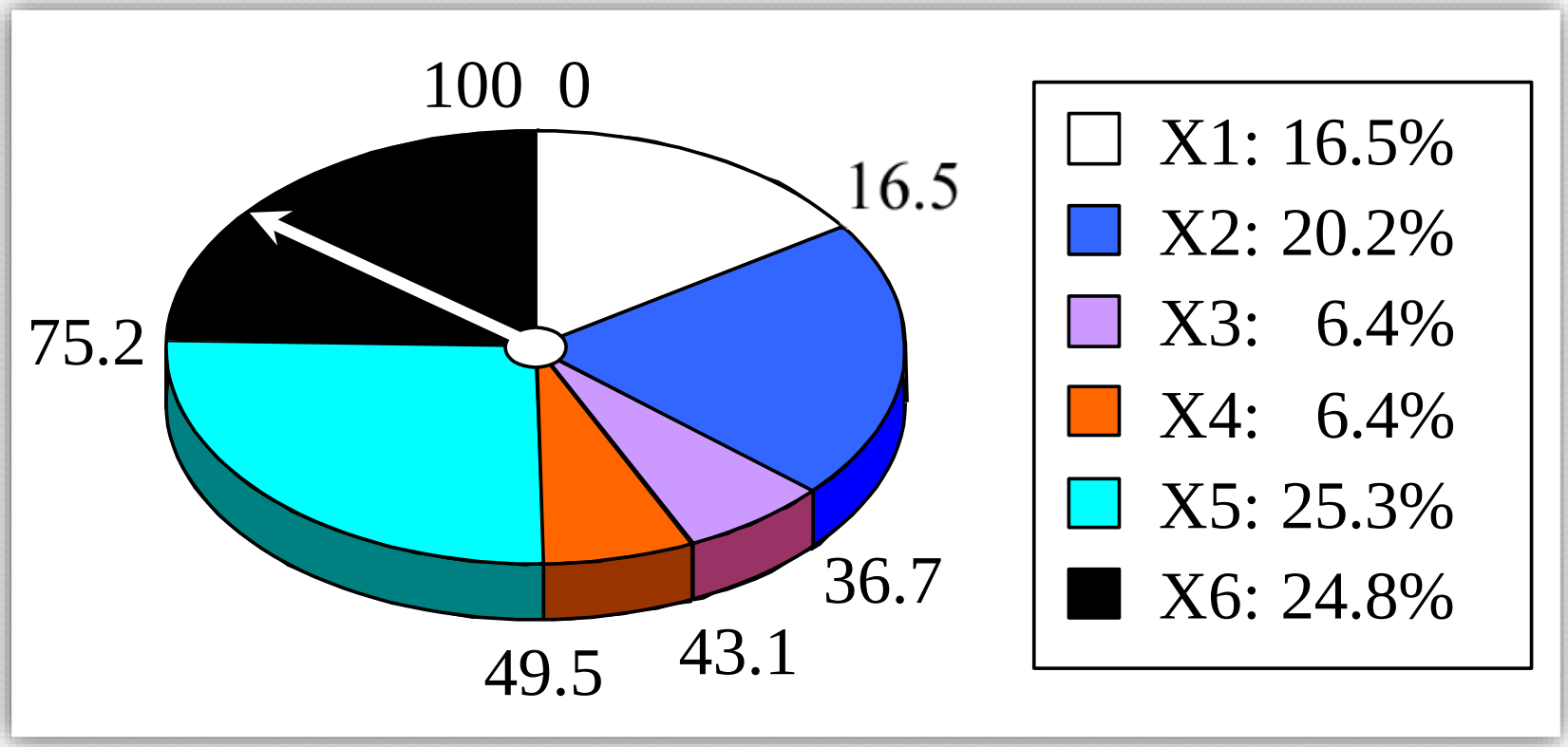
(*a*) Chromosome initial locations.



(*b*) Chromosome final locations.

- In natural selection, only the fittest species can survive, breed, and thereby pass their genes on to the next generation. GAs use a similar approach, but unlike nature, the size of the population remains unchanged from one generation to the next.
- The last column in the previous Table shows the 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.

Roulette wheel selection



Crossover operator

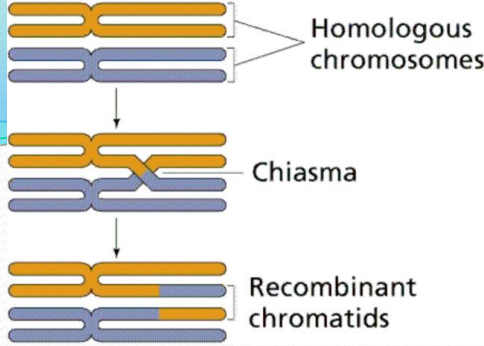
- In our example, we have an initial population of 6 chromosomes. Thus, to establish the same population in the next generation, the roulette wheel would be spun six times.

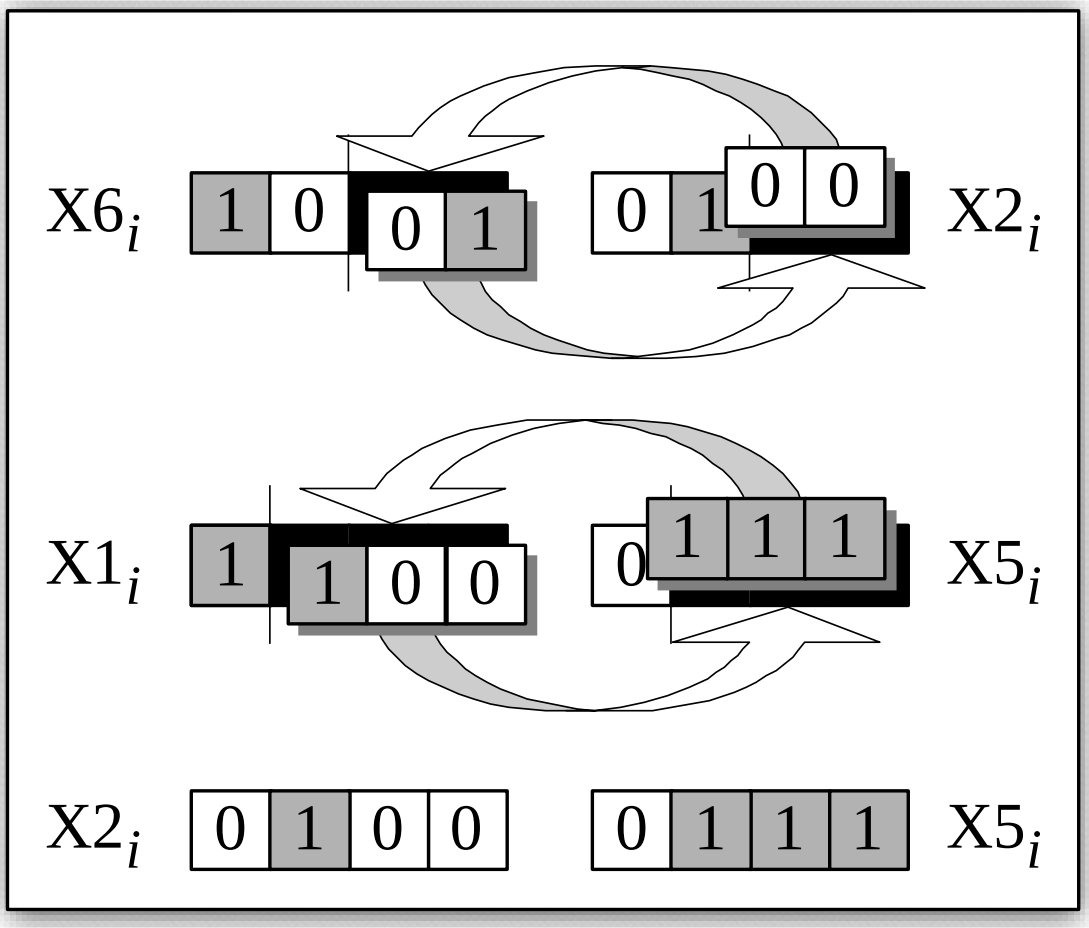
- Once a pair of parent chromosomes is selected, the crossover operator is applied.

- 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

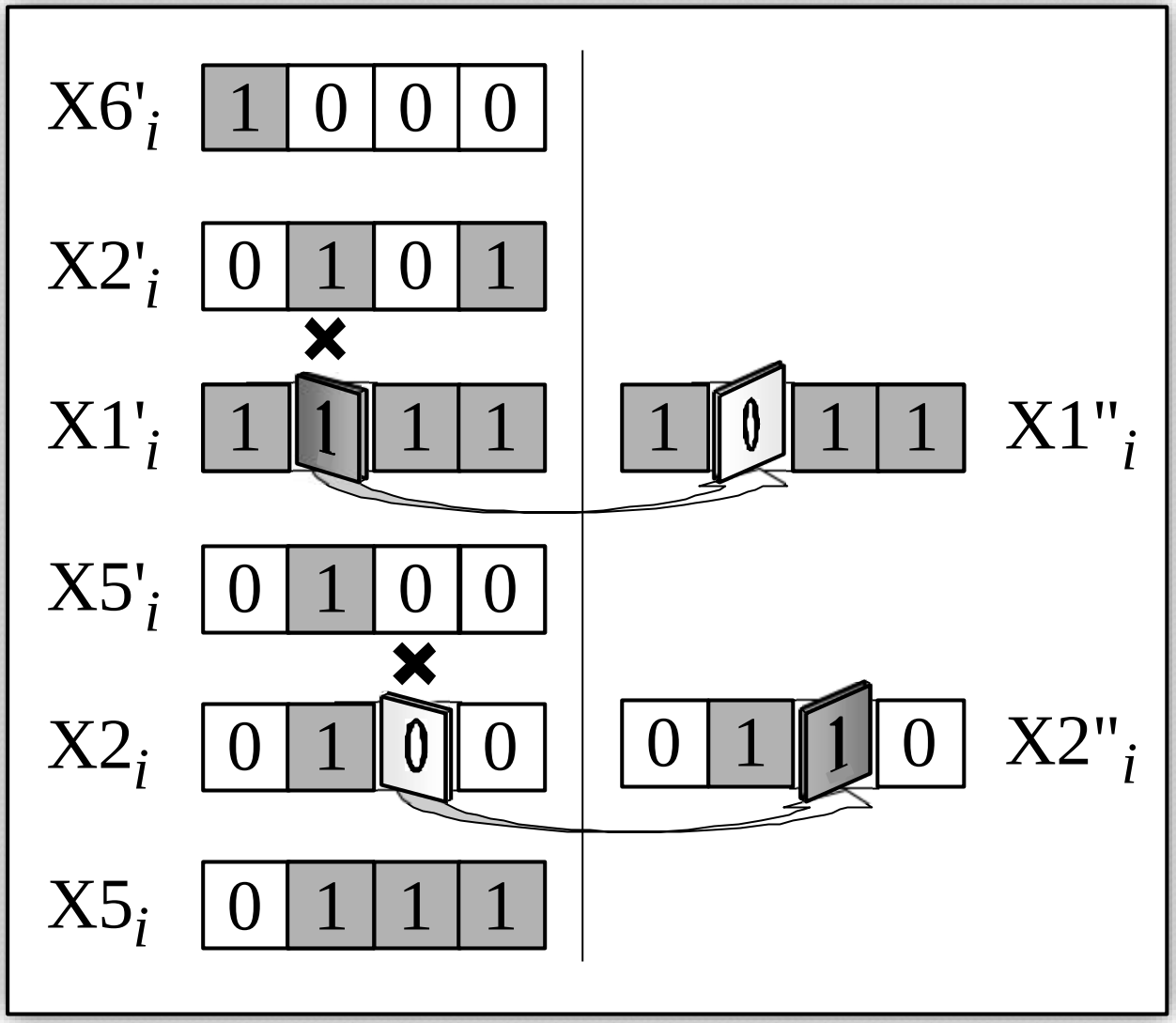

Homologous chromosomes
Chiasma
Recombinant chromatids


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

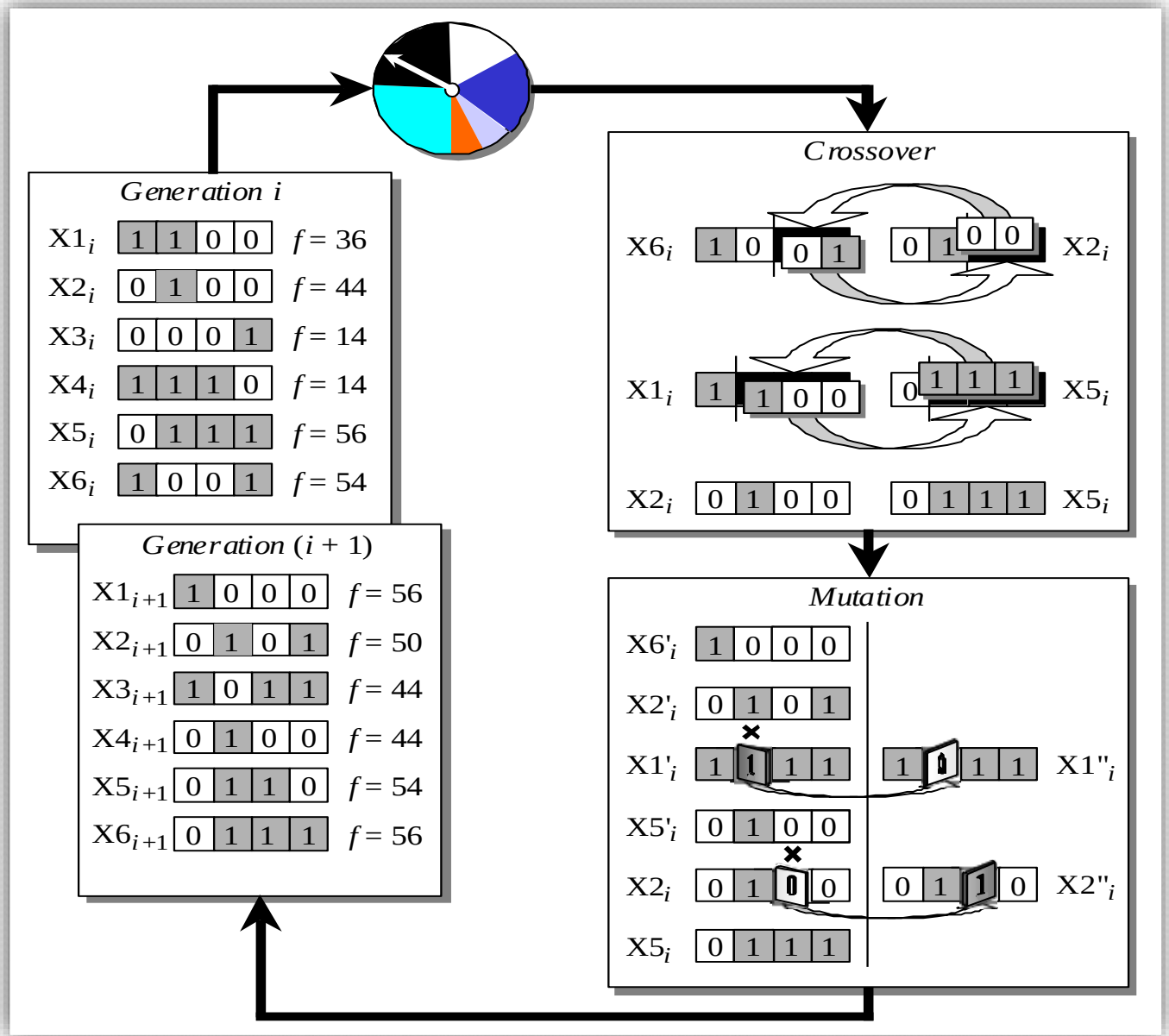
Mutation operator

- 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
X2i
0 1 0 0
0 1 1 0
X2"i
X5i
0 1 1 1

The genetic algorithm cycle



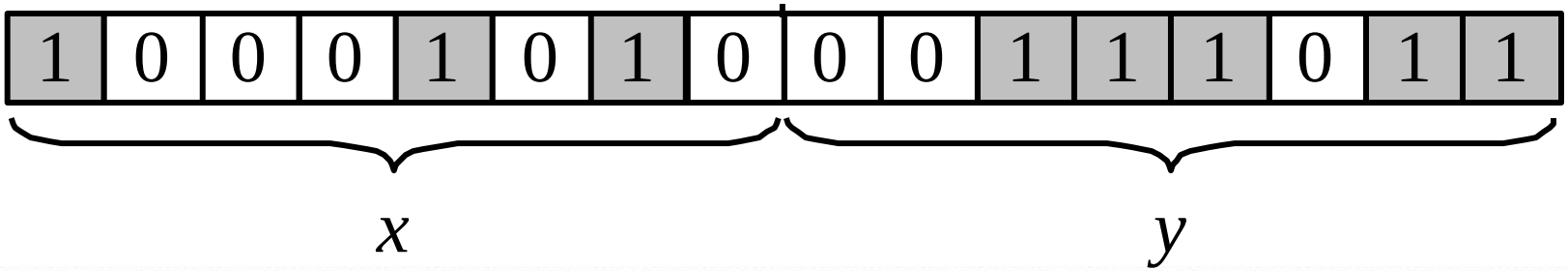
Genetic algorithms: Case study II

- Suppose it is desired to find the maximum of the "peak" function of two variables:

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3)\, e^{-x^2-y^2}$$

where parameters *x* and *y* vary between -3 and 3.

- The first step is to represent the problem variables as a chromosome - parameters *x* and *y* as a concatenated binary string:



- We also choose the size of the chromosome population, for instance 6, and randomly generate an initial population.
- The next step is to calculate the fitness of each chromosome. This is done in two stages.
- First, a chromosome, that is a string of 16 bits, is partitioned into two 8-bit strings:

1	0	0	0	1	0	1	0

and

0	0	1	1	1	0	1	1

- Then these strings are converted from binary (base 2) to decimal (base 10):

$$(10001010)_2 = 1\times 2^7 + 0\times 2^6 + 0\times 2^5 + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 = (138)_{10}$$

and

$$(00111011)_2 = 0\times 2^7 + 0\times 2^6 + 1\times 2^5 + 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 = (59)_{10}$$

- Now the range of integers that can be handled by 8-bits, that is the range from 0 to (2^8 - 1), is mapped to the actual range of parameters *x* and *y*, that is the range from -3 to 3:

$$\frac{6}{256-1} = 0.0235294$$

- To obtain the actual values of *x* and *y*, we multiply their decimal values by 0.0235294 and subtract 3 from the results:

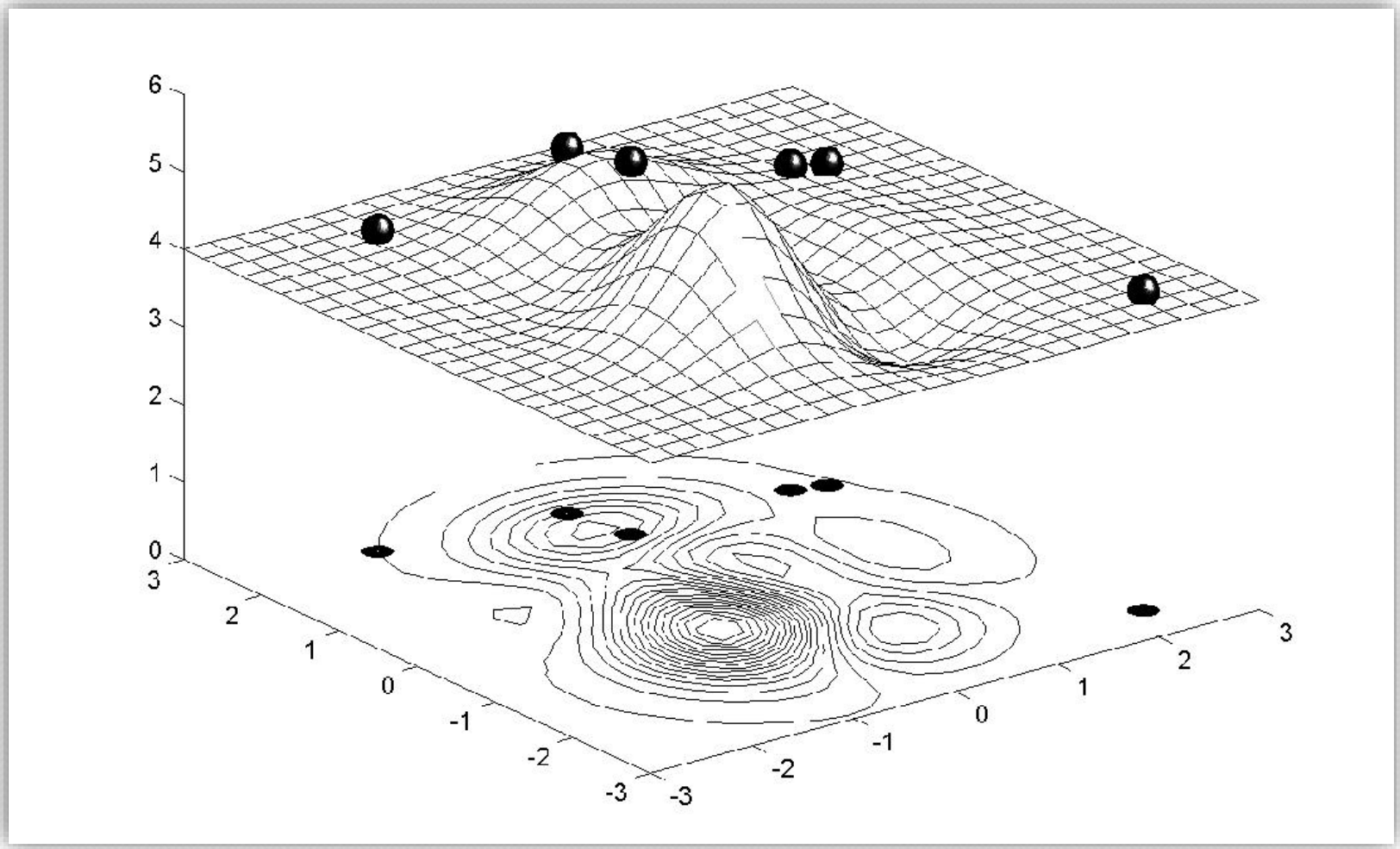
$$x = (138)_{10} \times 0.0235294 - 3 = 0.2470588$$

and

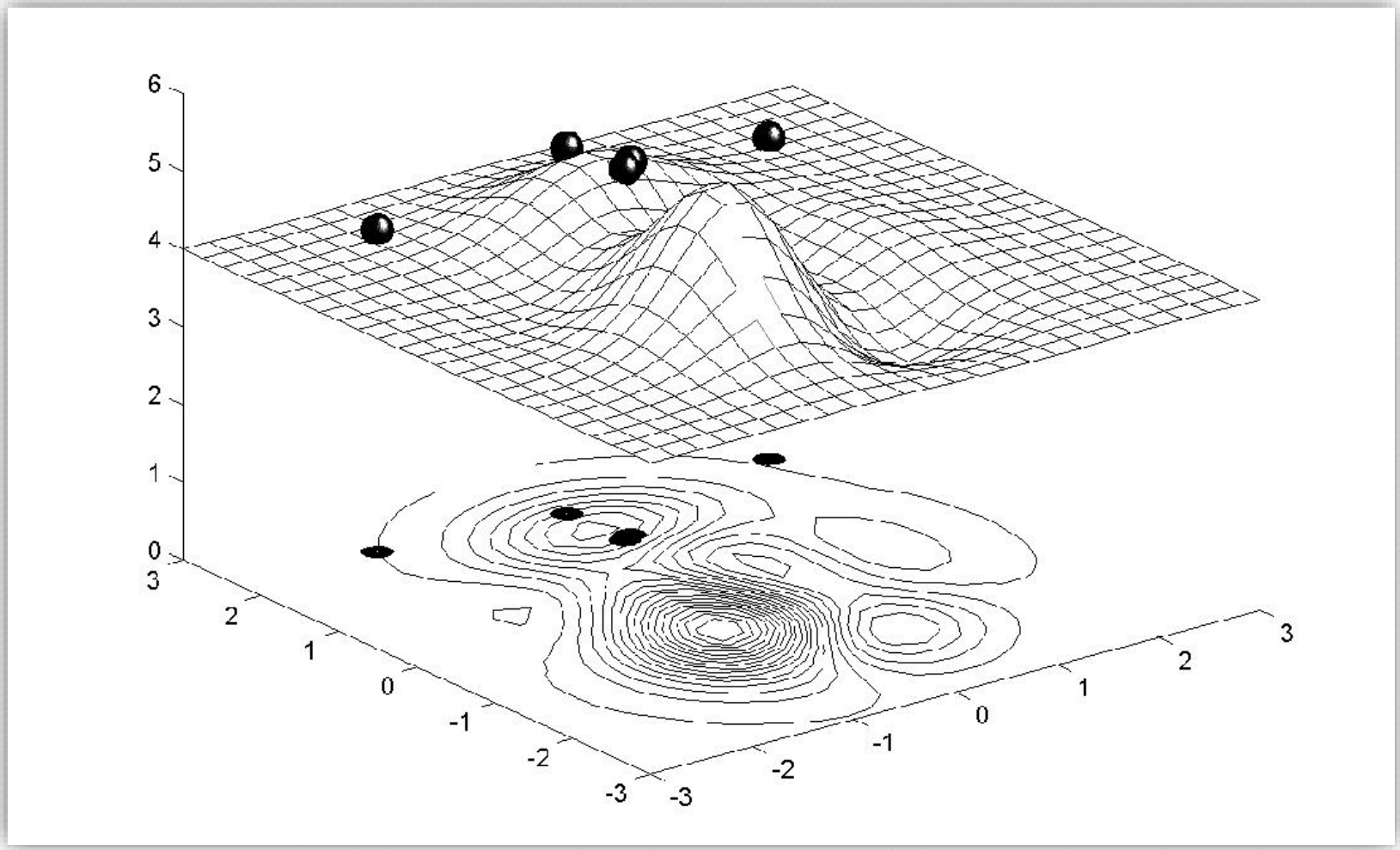
$$y = (59)_{10} \times 0.0235294 - 3 = -1.6117647$$

- Using decoded values of *x* and *y* as inputs in the mathematical function, the GA calculates the fitness of each chromosome.
- To find the maximum of the “peak” function, we will use crossover with the probability equal to 0.7 and mutation with the probability equal to 0.001. As we mentioned earlier, a common practice in GAs is to specify the number of generations. Suppose the desired number of generations is 100. That is, the GA will create 100 generations of 6 chromosomes before stopping.

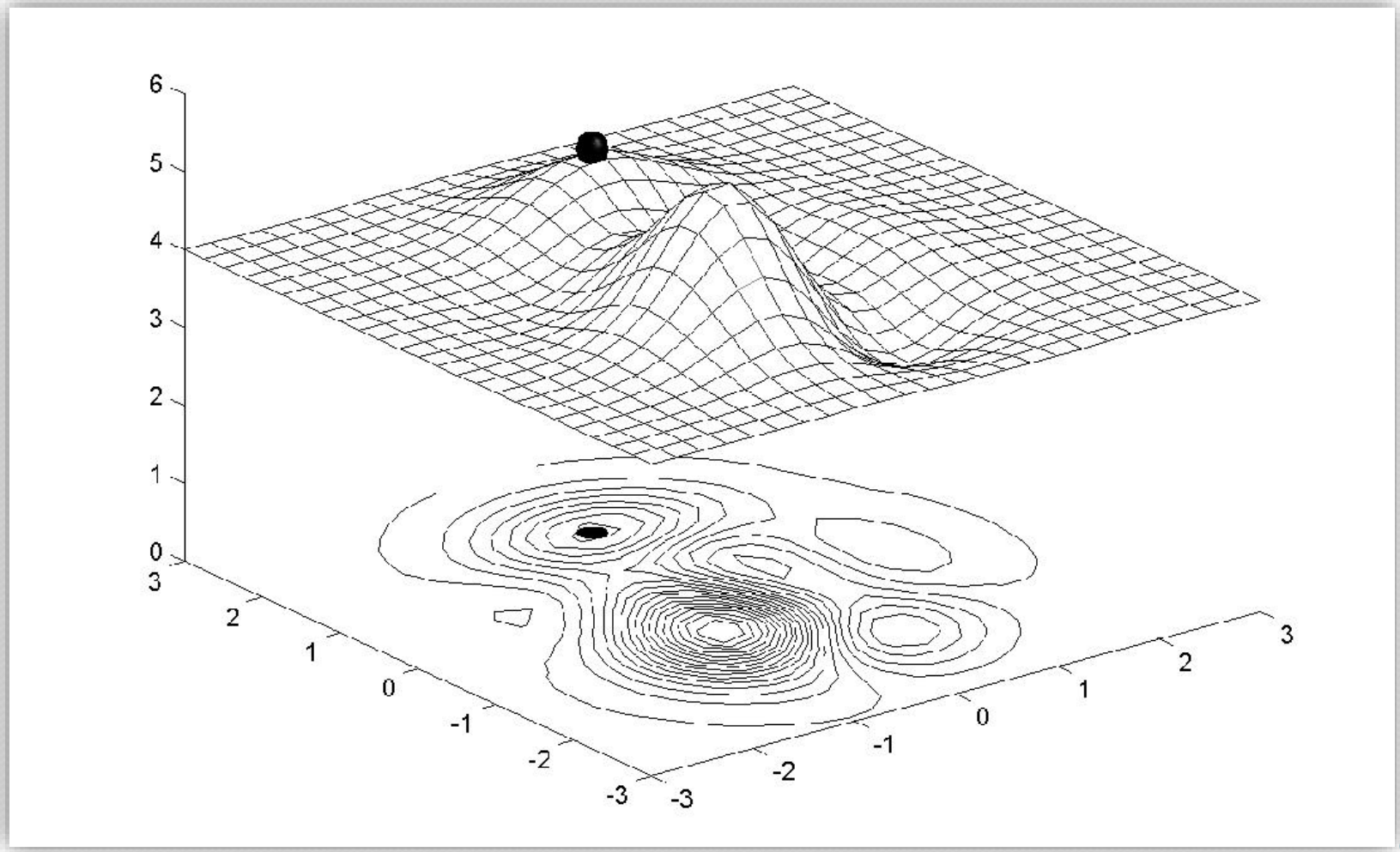
Chromosome locations on the surface of the "peak" function: initial population



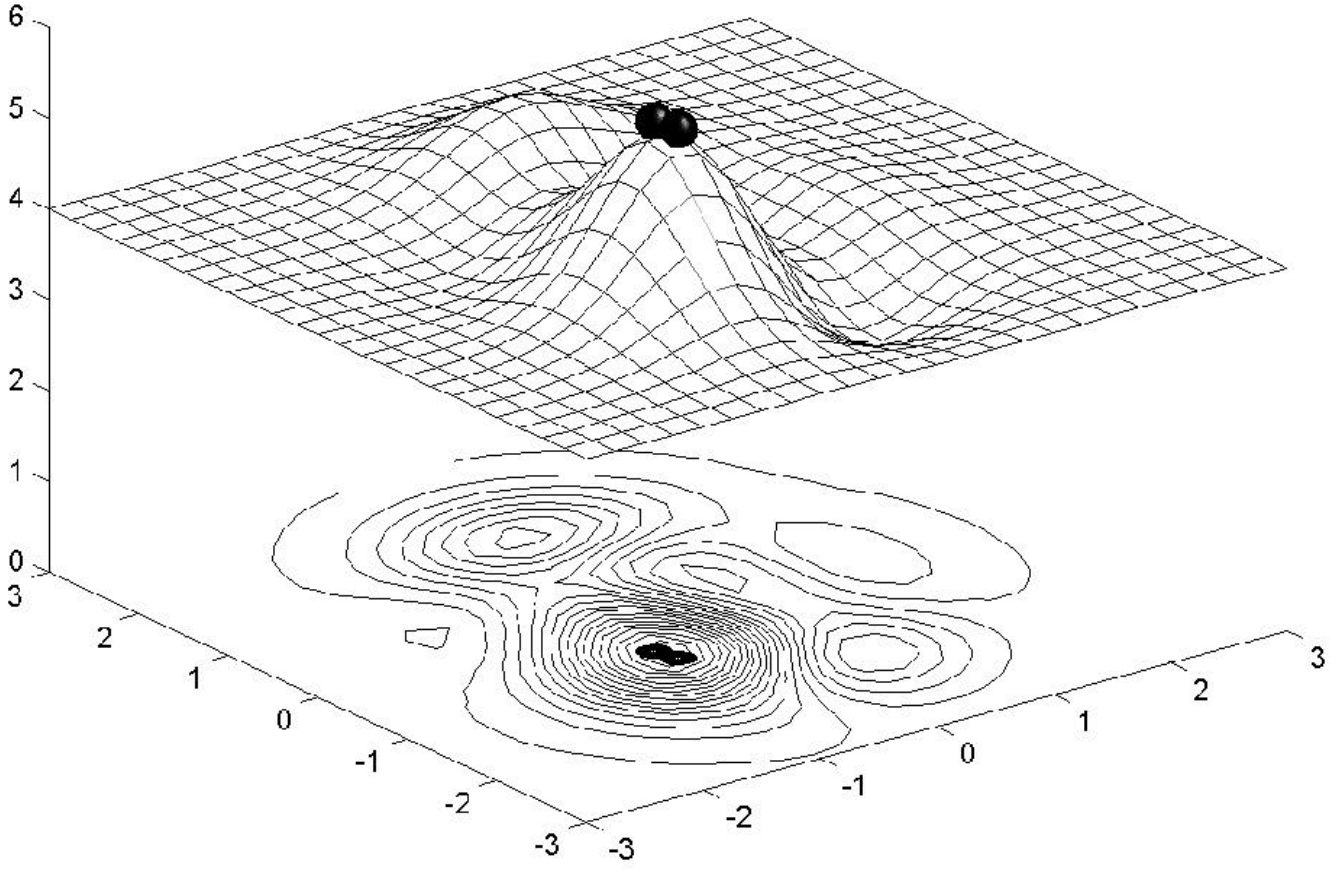
Chromosome locations on the surface of the "peak" function: first generation



Chromosome locations on the surface of the “peak” function: local maximum

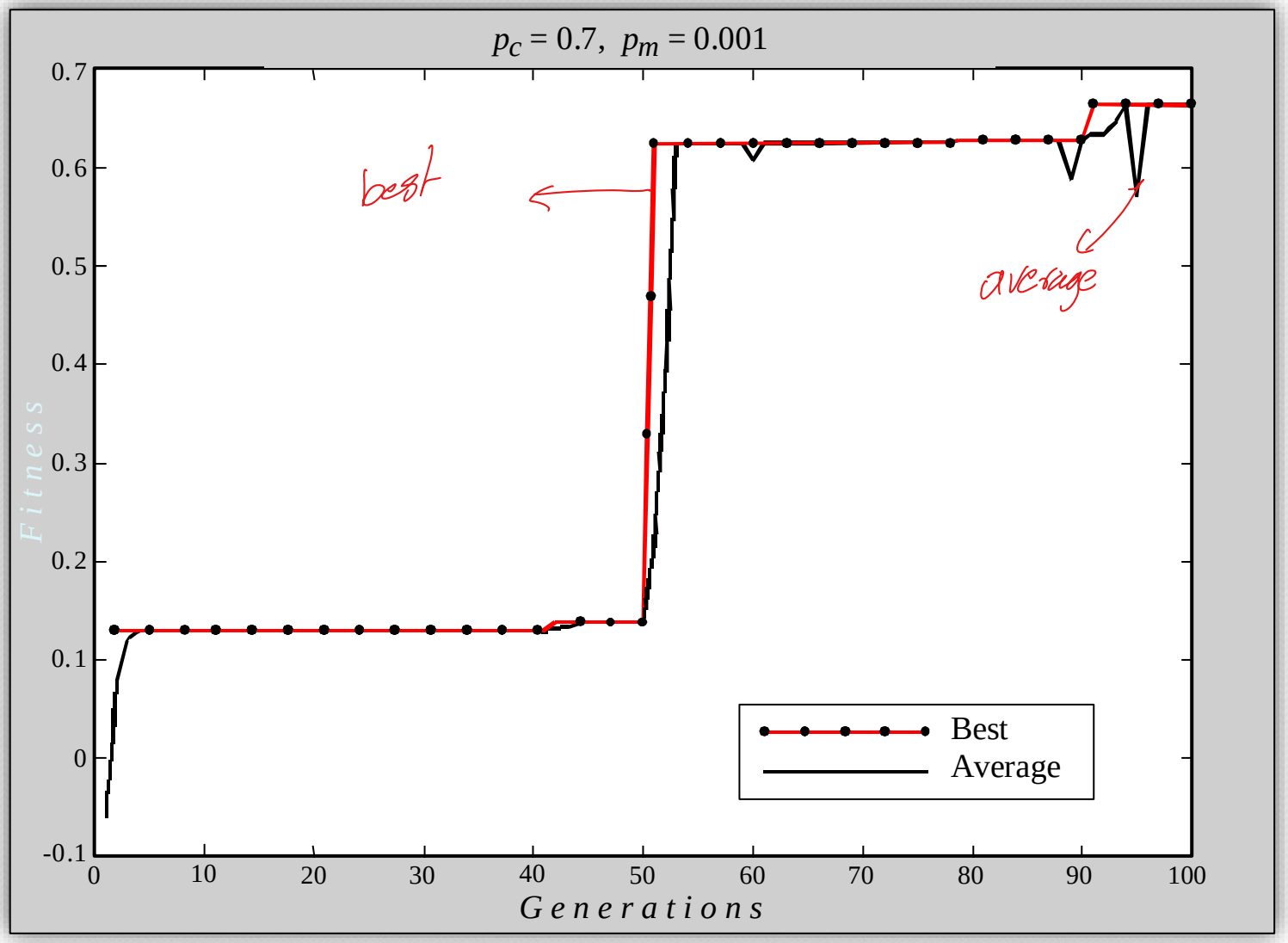


Chromosome locations on the surface of the “peak” function: global maximum

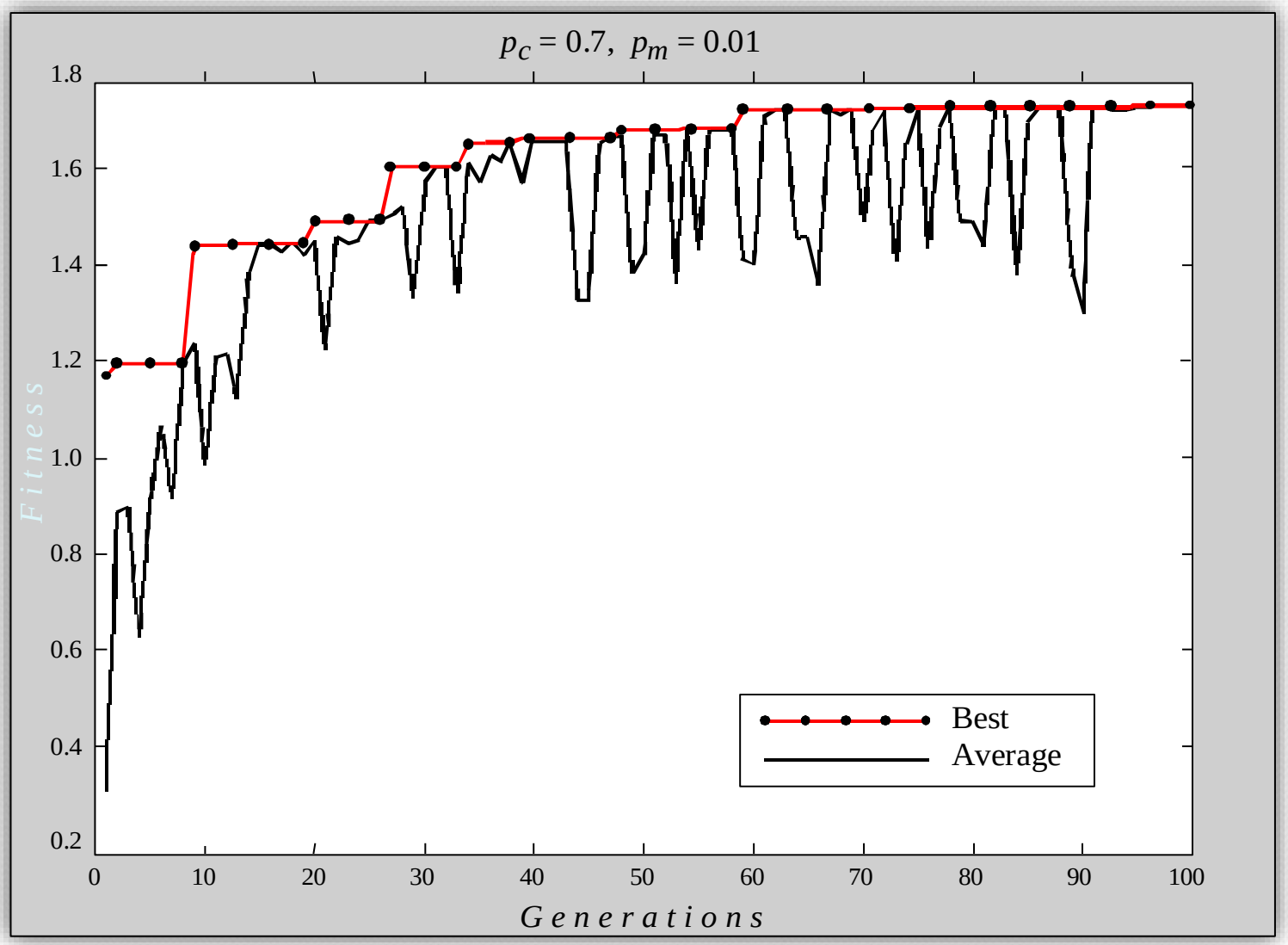


Performance graphs for 100 generations of 6 chromosomes: local maximum

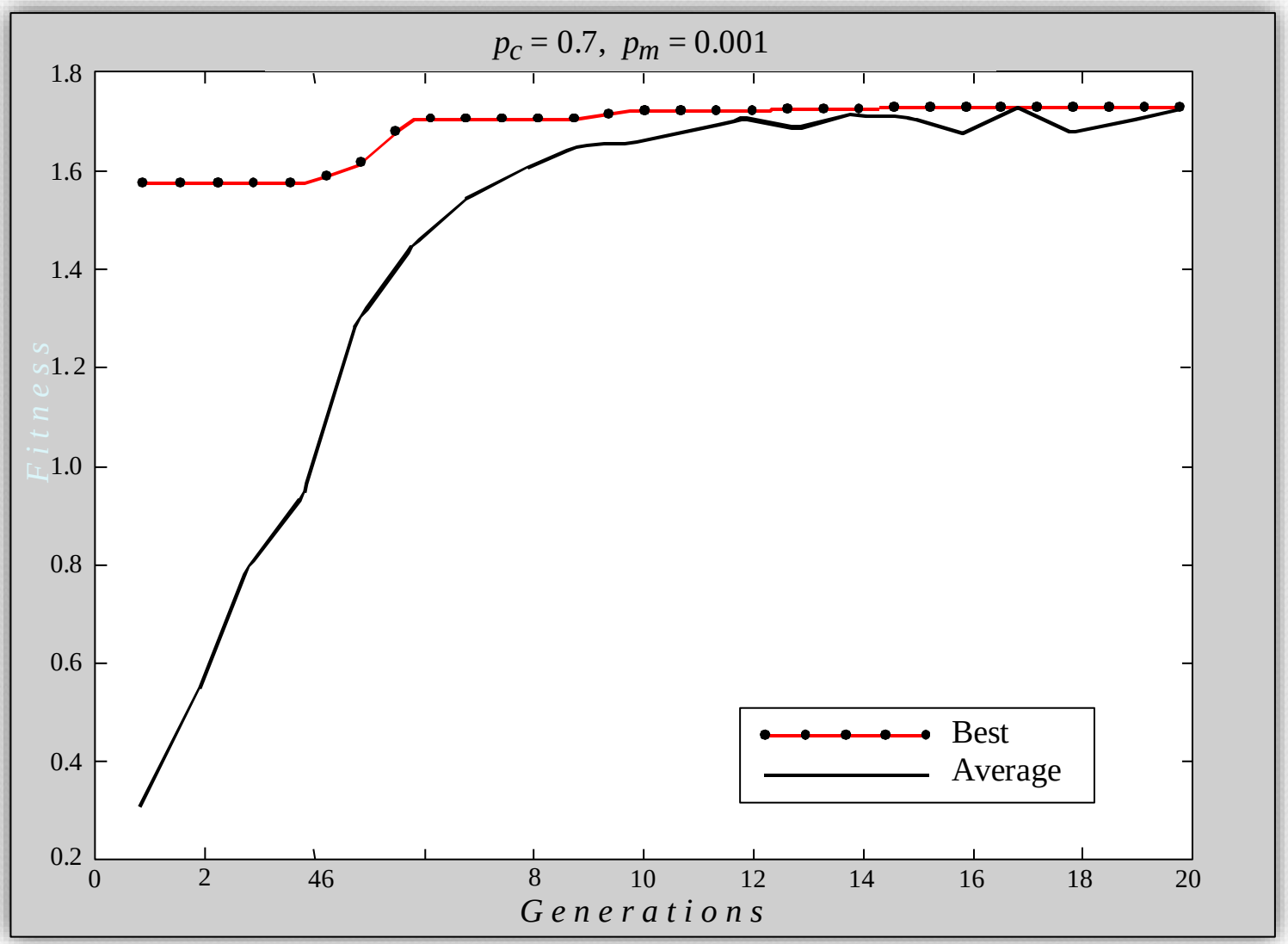
max



Performance graphs for 100 generations of 6 chromosomes: global maximum



Performance graphs for 20 generations of 60 chromosomes



Case study III: Maintenance scheduling

- Maintenance scheduling problems are usually solved using a combination of search techniques and heuristics.
- These problems are complex and difficult to solve.
- They are NP-complete and cannot be solved by combinatorial search techniques.
- Scheduling involves competition for limited resources, and is complicated by a great number of badly formalized constraints.

Steps in the GA development

1. Specify the problem, define constraints and optimum criteria;
2. Represent the problem domain as a chromosome;
3. Define a fitness function to evaluate the chromosome performance;
4. Construct the genetic operators;
5. Run the GA and tune its parameters.

Case study

Scheduling of 7 units in 4 equal intervals

The problem constraints:

- The maximum loads expected during four intervals are 80, 90, 65 and 70 MW;
- Maintenance of any unit starts at the beginning of an interval and finishes at the end of the same or adjacent interval. The maintenance cannot be aborted or finished earlier than scheduled;
- The net reserve of the power system must be greater or equal to zero at any interval.

The optimum criterion is the maximum of the net reserve at any maintenance period.

Case study

Unit data and maintenance requirements

Unit number	Unit capacity	Number of intervals required for unit maintenance
1	20	2
2	15	2
3	35	1
4	40	1
5	15	1
6	15	1
7	10	1

loads expected during four intervals are 80, 90, 65 and 70 MW, respectively.

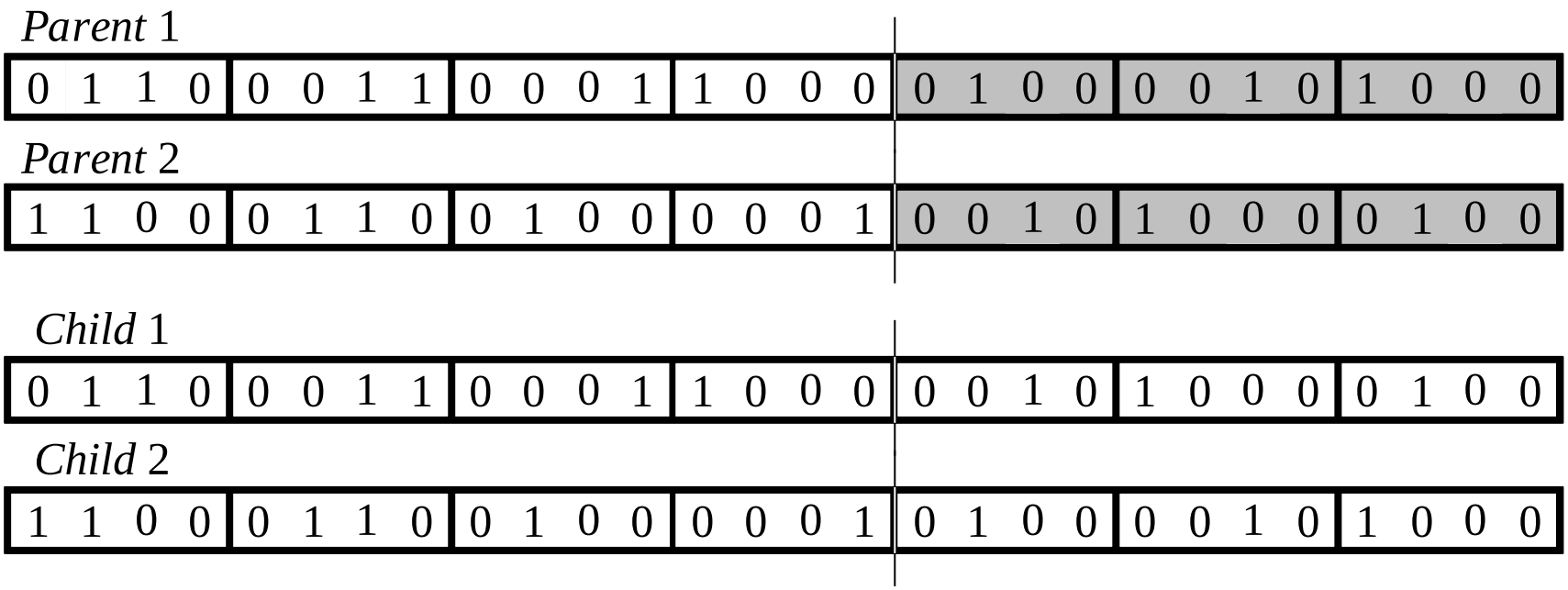
Unit gene pools

Unit 1:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 2:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 3:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 4:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 5:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 6:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 7:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1

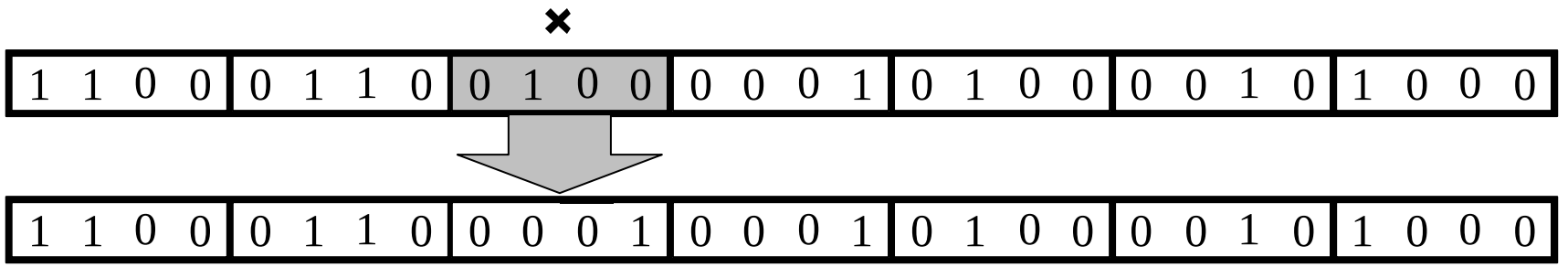
Chromosome for the scheduling problem

Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7
0 1 1 0	0 0 1 1	0 0 0 1	1 0 0 0	0 1 0 0	0 0 1 0	1 0 0 0

The crossover operator

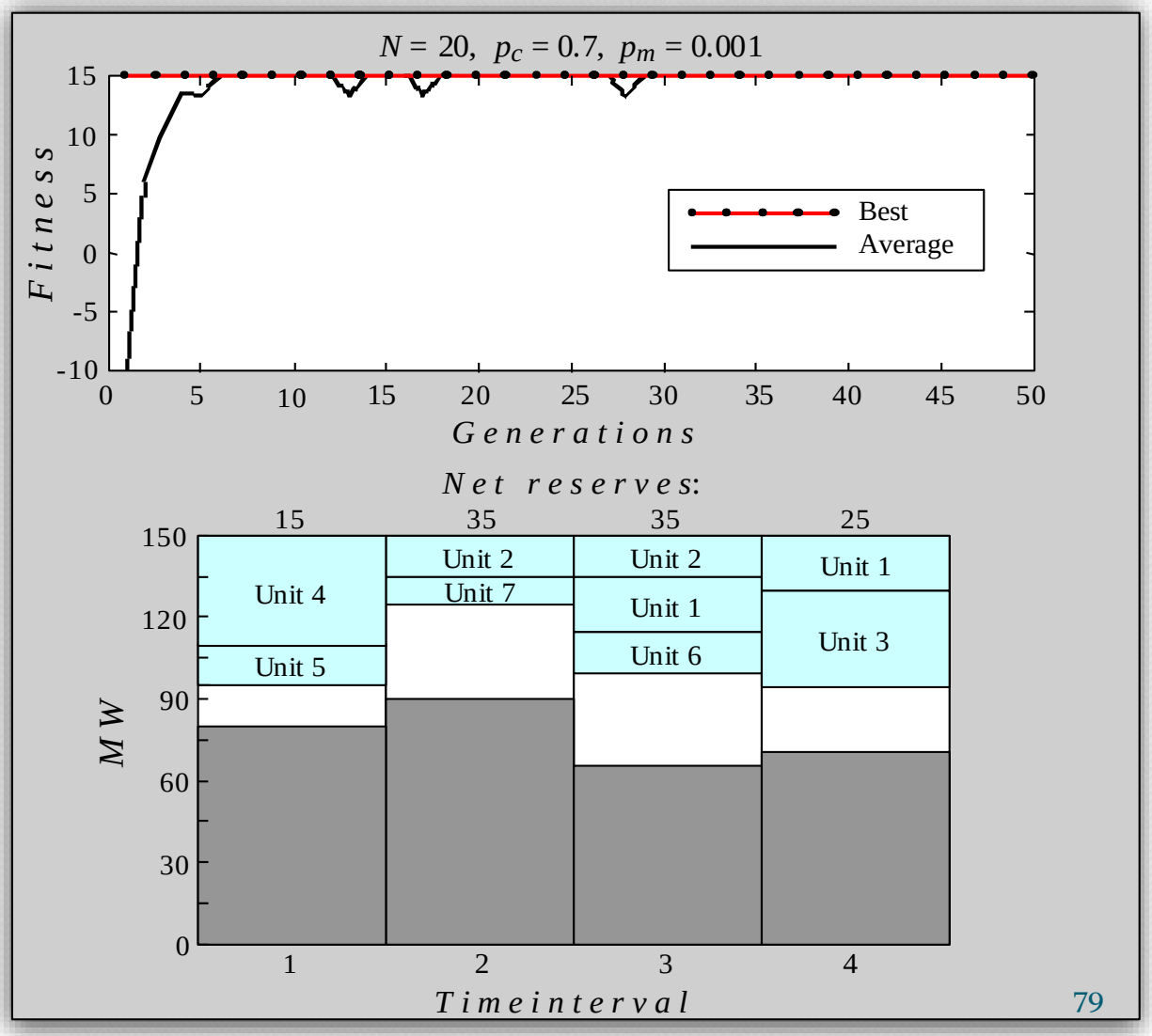


The mutation operator



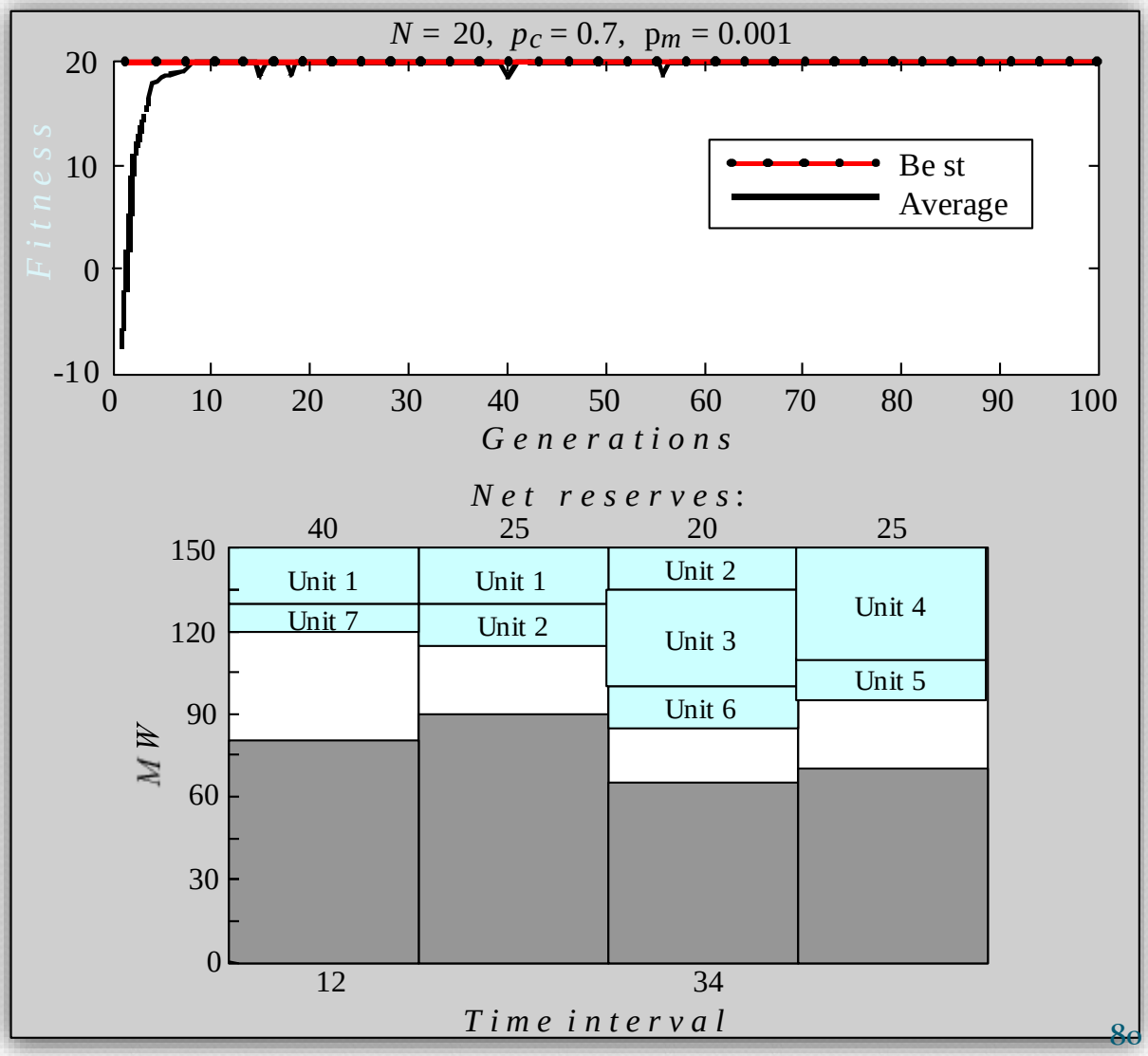
Performance graphs and the best maintenance schedules created in a population of 20 chromosomes

(*a*) 50 generations



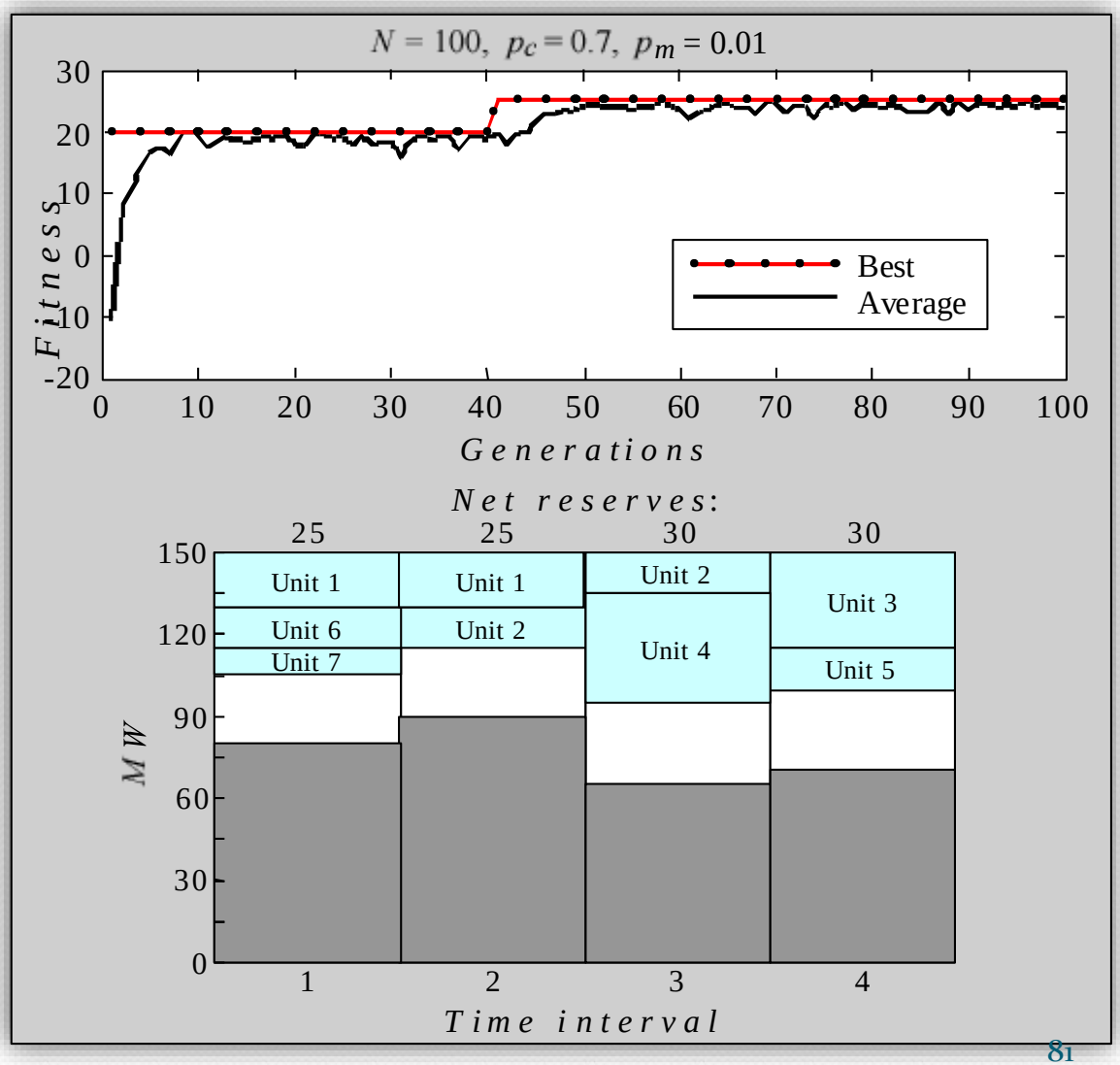
Performance graphs and the best maintenance schedules created in a population of 20 chromosomes

(*b*) 100 generations



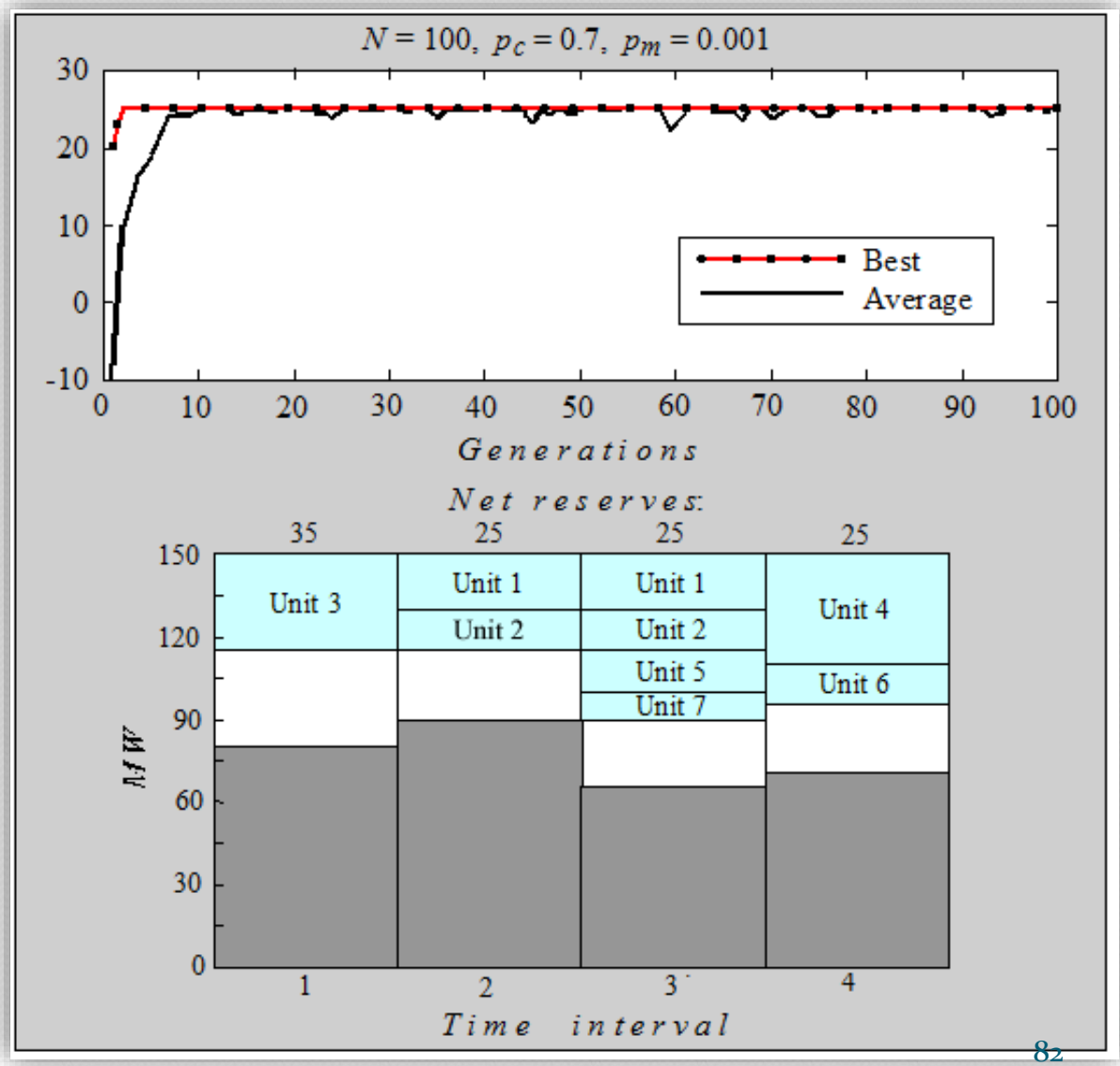
Performance graphs and the best maintenance schedules created in a population of 100 chromosomes

(*b*) Mutation rate is 0.01



Performance graphs and the best maintenance schedules created in a population of 100 chromosomes

(*a*) Mutation rate is 0.001



Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7
0 1 1 0	0 0 1 1	0 0 0 1	1 0 0 0	0 1 0 0	0 0 1 0	1 0 0 0

A chromosome for the scheduling problem

Fitness Function

The evaluation of a chromosome starts with the sum of capacities of the units scheduled for maintenance at each interval. For the chromosome shown in Figure 7.9, we obtain:

Interval 1: $0 \times 20 + 0 \times 15 + 0 \times 35 + 1 \times 40 + 0 \times 15 + 0 \times 15 + 1 \times 10 = 50$

Interval 2: $1 \times 20 + 0 \times 15 + 0 \times 35 + 0 \times 40 + 1 \times 15 + 0 \times 15 + 0 \times 10 = 35$

Interval 3: $1 \times 20 + 1 \times 15 + 0 \times 35 + 0 \times 40 + 0 \times 15 + 1 \times 15 + 0 \times 10 = 50$

Interval 4: $0 \times 20 + 1 \times 15 + 1 \times 35 + 0 \times 40 + 0 \times 15 + 0 \times 15 + 0 \times 10 = 50$

Then these values are subtracted from the total installed capacity of the power system (in our case, 150 MW):

Interval 1: $150 - 50 = 100$
Interval 2: $150 - 35 = 115$
Interval 3: $150 - 50 = 100$
Interval 4: $150 - 50 = 100$

And finally, by subtracting the maximum loads expected at each interval, we obtain the respective net reserves:

Interval 1: $100 - 80 = 20$
Interval 2: $115 - 90 = 25$
Interval 3: $100 - 65 = 35$
Interval 4: $100 - 70 = 30$

Since all the results are positive, this particular chromosome does not violate any constraint, and thus represents a legal schedule. The chromosome's fitness is determined as the lowest of the net reserves; in our case it is 20.

Genetic Cars [Java Applet]

Brian storming:

Why genetic algorithms work?

Think intuitively and mathematically.

Japan’s high speed bullet trains (600KM/h), designing the aerodynamic shape of its nose using GA

Neuro-evolution - Car learns to drive [5:52]

Adapting Morphology to Multiple Tasks in Evolved Virtual Creatures [6:02]

Auto adaptable walking robot with genetic algorithm (2:57)

Evolved Virtual Creature [3:36]

Genetic algorithm. Learning to jump over ball [2:53]

Let us practice one hour thinking at home!

Q1. What is my life most important question?

Q2. What is my life most important optimization problem?

Q3. Which research real-world applications are more attractive for me? And why?

Q4. How can I balance my study/work and life?

Q5. Who is/are my academic and life role model(s)?

Q6. How did/do I select my research direction(s)?

Let us practice one hour thinking at home! (Cont.)

Q7. Do I target academia or industry in future?

Q8. How do I measure successes in my life?

Q9. What are the main constraints in my life?

Q10. How can I have a better relationship with my supervisor(s)?

Q11. What can be my highest life achievement?

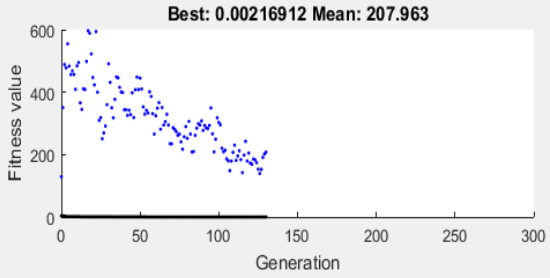
Q12. How can I make my life more happier?

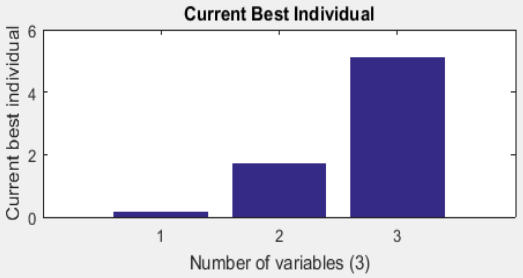
Running GA in Matlab: An Example

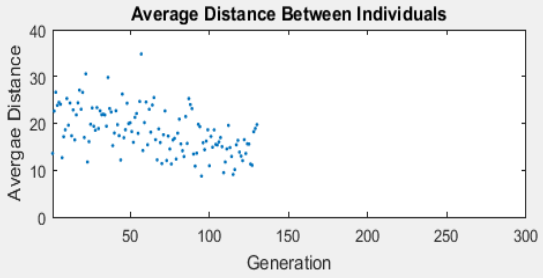
- Run in Matlab: optimtool('ga')

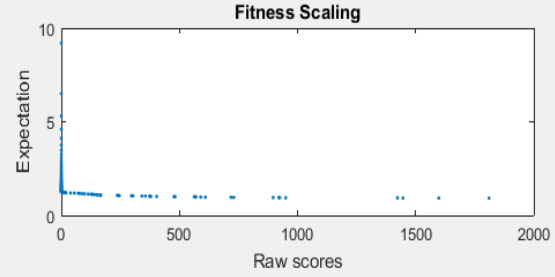
Objective function:

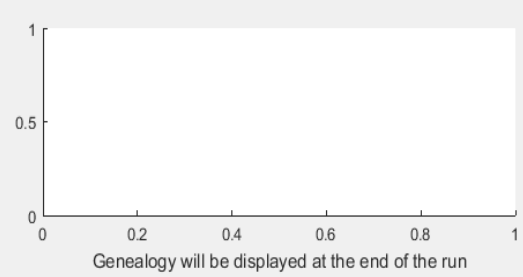
@(x) (x(1) - 0.2)^2 + (x(2) - 1.7)^2 + (x(3) -5.1)^2

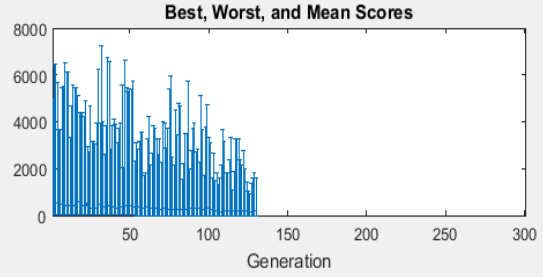

Best: 0.00216912 Mean: 207.963
Fitness value
Generation

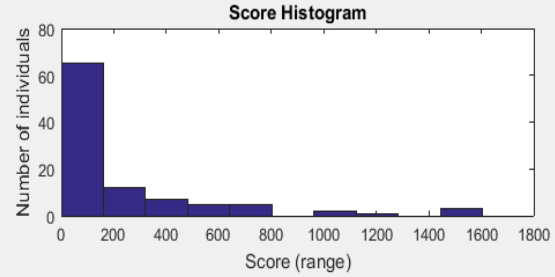

Current Best Individual
Current best individual
Number of variables (3)

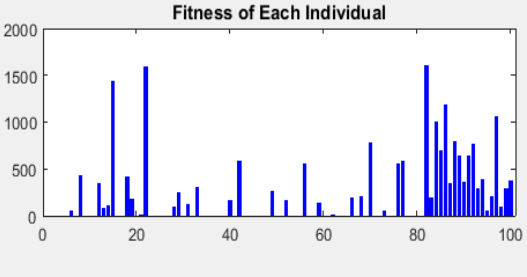

Average Distance Between Individuals
Avergae Distance
Generation

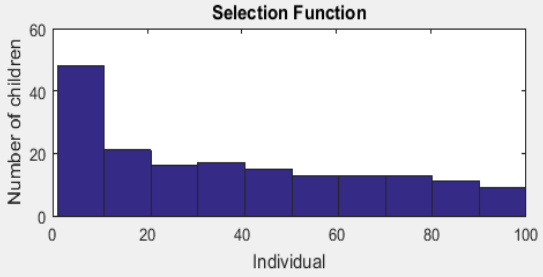

Fitness Scaling
Expectation
Raw scores

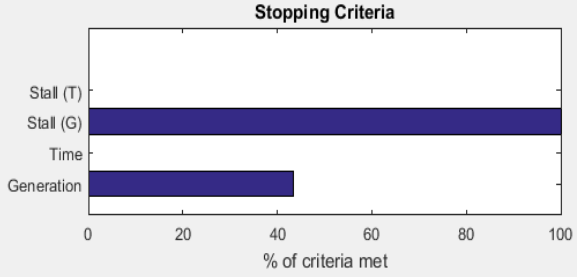

Genealogy will be displayed at the end of the run

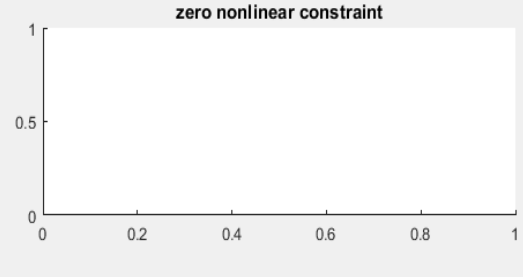

Best, Worst, and Mean Scores
Generation


Score Histogram
Number of individuals
Score (range)


Fitness of Each Individual


Selection Function
Number of children
Individual


Stopping Criteria
Stall (T)
Stall (G)
Time
Generation
% of criteria met


zero nonlinear constraint

Sample GA application

- Programming Architecture - Genetic Algorithms - Structural Optimization Of Free Form Grid Shells [2:11]

Radiation Therapy Planning by Genetic Algorithm [4:33]

Reducing the searching space by using well-designed representation

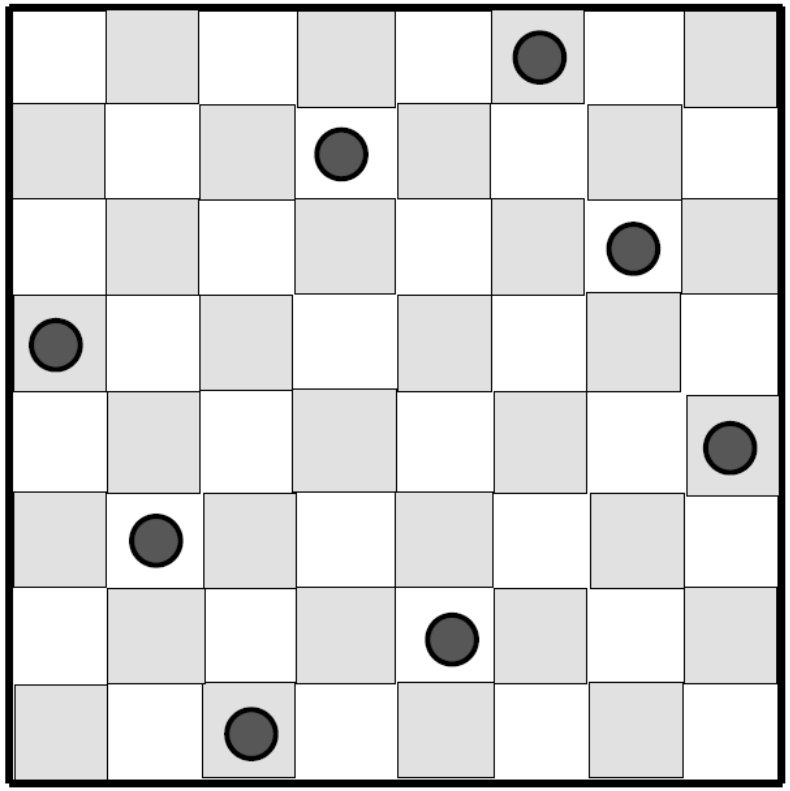


FIGURE 1.17 A solution for the 8-Queens problem represented by the permutation (6,4,7,1,8,2,5,3).

8-Queen Problem

- Method 1) Encode a solution by a vector of eight Cartesian positions x = (p1, p2, . . . , p8) where pi = (xi, yi) represents the Cartesian position of the queen i: The number of possibilities (size of the search space)= 64^8 that is over 4 billion candidate solutions.
- Method 2) If we prohibit more than one queen per row, so that each queen is assigned to a separate row: The number of possibilities = 8^8 candidate solutions which is over 16 million possibilities (~0.4% of method 1).
- Method 3) if we forbid two queens to be both in the same column or row, the encoding will be reduced to a permutation of the n queens: The number of possibilities = n! candidate solutions, which is only 40,320 possibilities for the 8-Queens problem (~0.001% of method 1).

Important Conclusion

Do your best to reduce the search space size for optimization, don't put an unnecessarily heavy task on shoulder of an optimizer by just doing a fast modeling

Bonus Question

For which of following cases, does the search space grow faster? Why? Show mathematically.

a) Increasing range of the variables by factor of k

b) Increasing dimension of the search space by factor of k

Bonus task:

Design a GA (chromosome, crossover, mutation, fitness function) to solve 8-queen problem [1%]

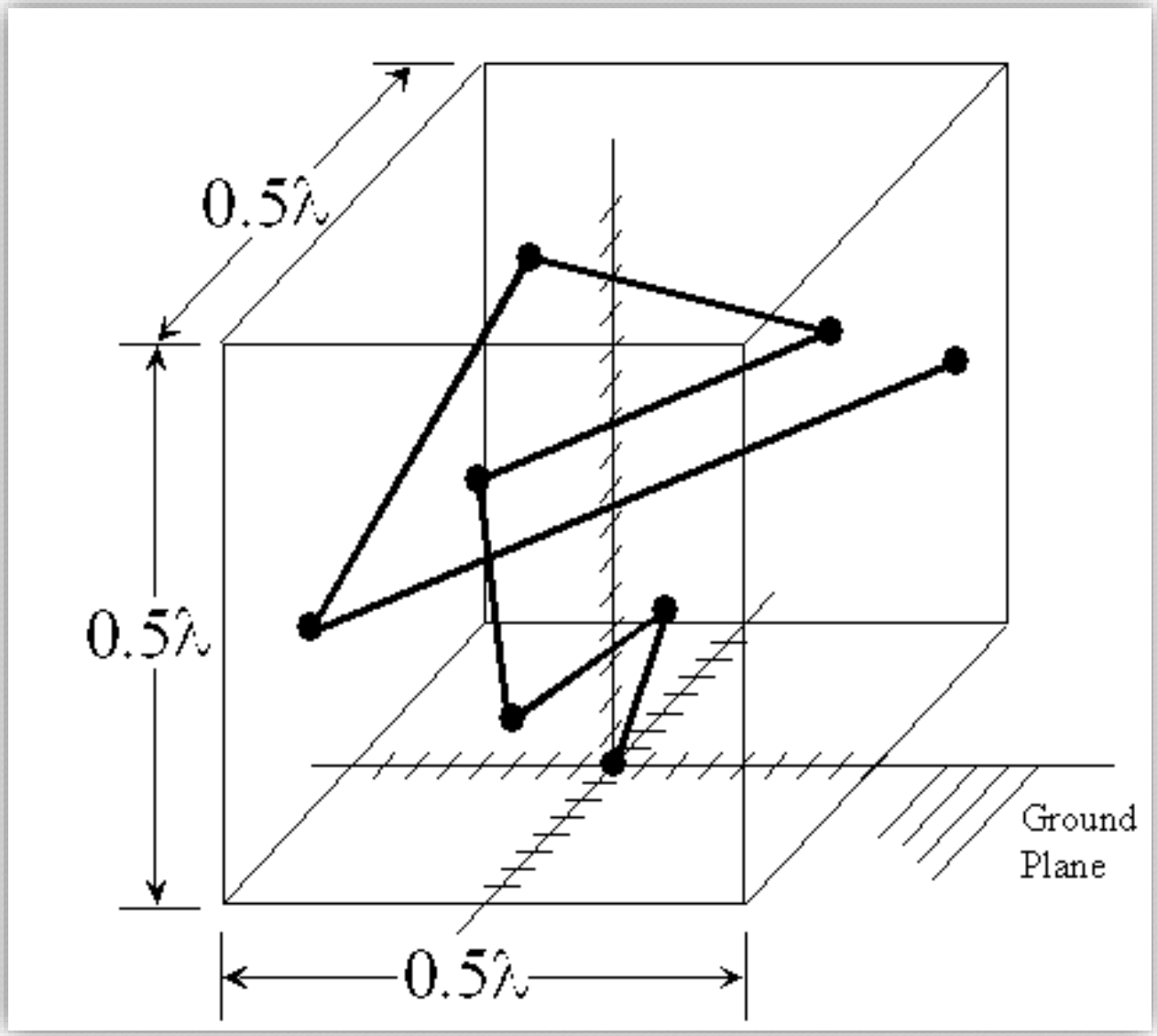
("N")	Total Solutions	Unique Solutions
1	1	1
2	0	0
3	0	0
4	2	1
5	10	2
6	4	1
7	40	6
8	92	12
9	352	46
10	724	92
11	2,680	341
12	14,200	1,787
13	73,712	9,233
14	365,596	45,752
15	2,279,184	285,053
16	14,772,512	1,846,955
17	95,815,104	11,977,939
18	666,090,624	83,263,591
19	4,968,057,848	621,012,754
20	39,029,188,884	4,878,666,808
21	314,666,222,712	39,333,324,973
22	2,691,008,701,644	336,376,244,042
23	24,233,937,684,440	3,029,242,658,210
24	227,514,171,973,736	28,439,272,956,934
25	2,207,893,435,808,352	275,986,683,743,434
26	22,317,699,616,364,044	2,789,712,466,510,289

BQ

What is the most difficult two dimensional problem, which even the super computers are not able to solve it?

Evolution is a Blind Watchmaker [9:48]

U. S. PATENT 5,719,794



Antenna Design

- The problem (Altshuler and Linden 1998) is to determine the *x*-*y*-*z* coordinates of the 3-dimensional position of the ends (*X1, Y1, Z1, X2, Y2, Z2,… , X7, Y7, Z7*) of 7 straight wires so that the resulting 7-wire antenna satisfies certain performance requirements
- The first wire starts at feed point (0, 0, 0) in the middle of the ground plane
- The antenna must fit inside the 0.5λ cube

Antenna Design

X_1	Y_1	Z_1	X_2	Y_2	Z_2	…
+0010	**-1110**	**+0001**	**+0011**	**-1011**	**+0011**	**…**

- 105-bit chromosome
- Each *x*-*y*-*z* coordinate is represented by 5 bits (4-bit granularity for data plus a sign bit)
- Total chromosome size is 3 × 7 × 5 = 105 bits

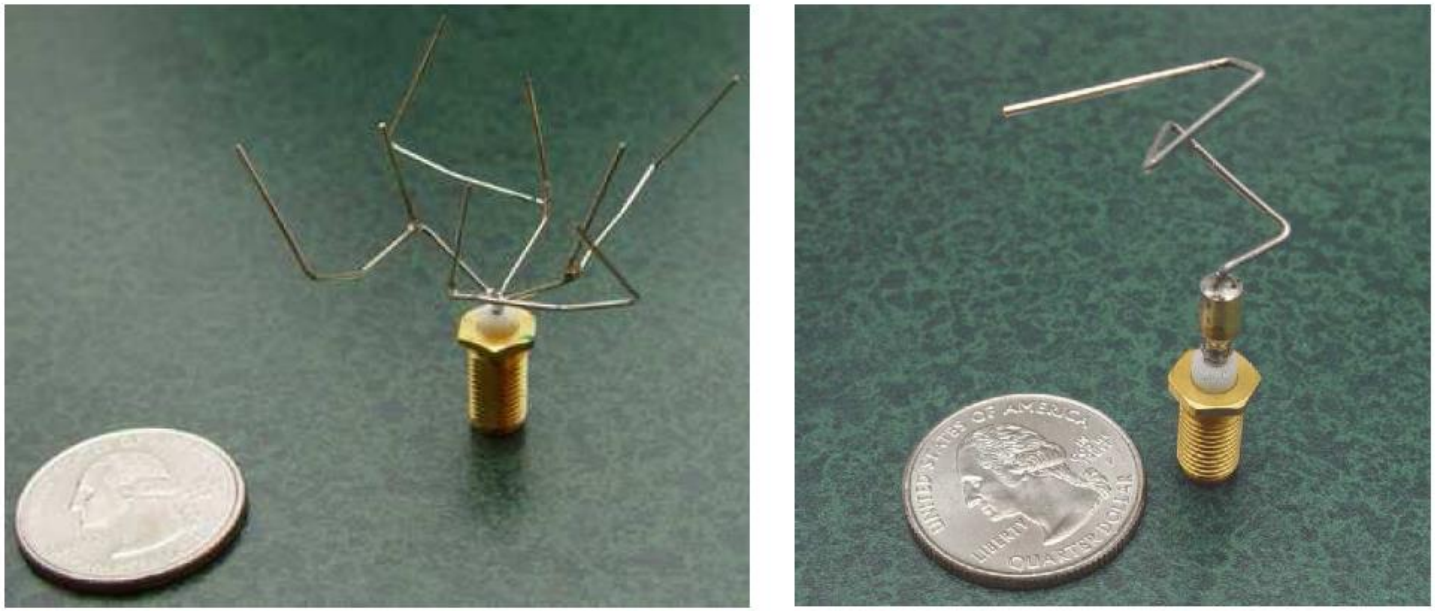
Antenna Fitness

- Antenna is for ground-to-satellite communications for cars and handsets
- We desire near-uniform gain pattern 10° above the horizon
- Fitness is measured based on the antenna's radiation pattern. The radiation pattern is simulated by National Electromagnetics Code (NEC)

Antenna Fitness

- Fitness is sum of the squares of the difference between the average gain and the antenna's gain
- Sum is taken for angles Θ between -90° and +90° and all azimuth angles Φ from 0° to 180°
- The smaller the value of fitness, the better

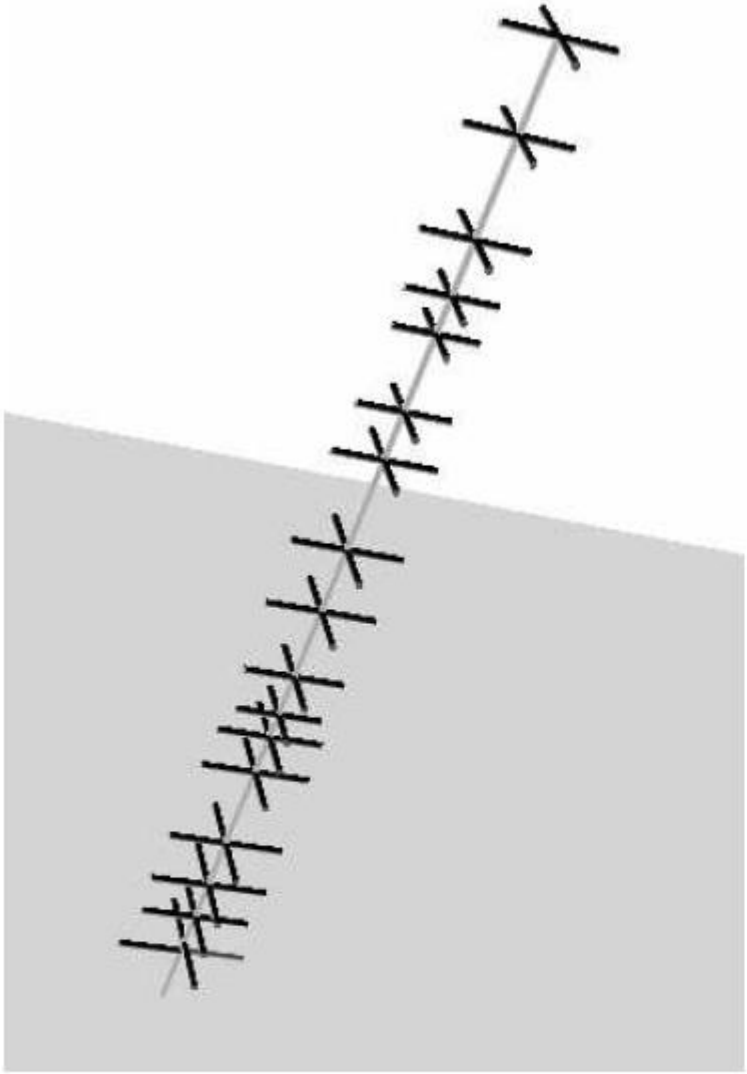
NASA EVOLVED ANTENNA



(a) (b)

Photographs of prototype evolved antennas: (a) the best evolved antenna for the initial gain pattern requirement, ST5-3-10; (b) the best evolved antenna for the revised specifications, ST5-33-142-7.

Utilized in the satellite launched in 2006

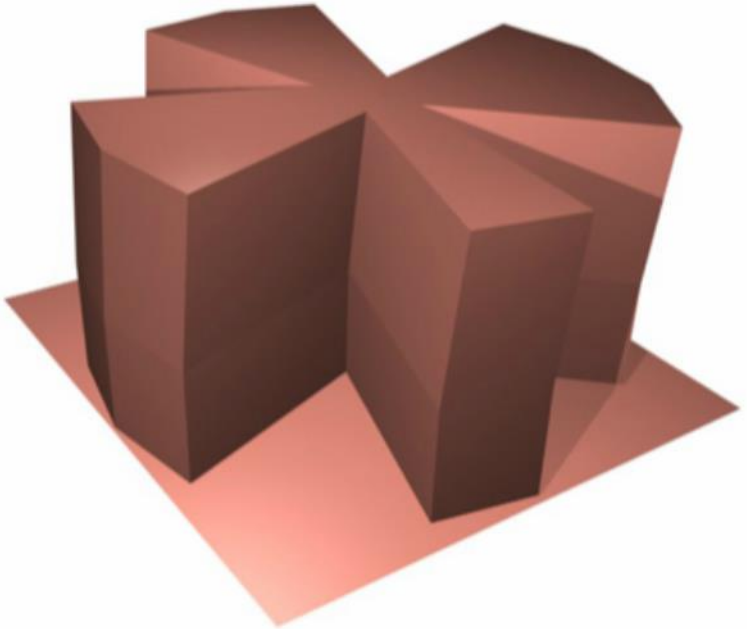


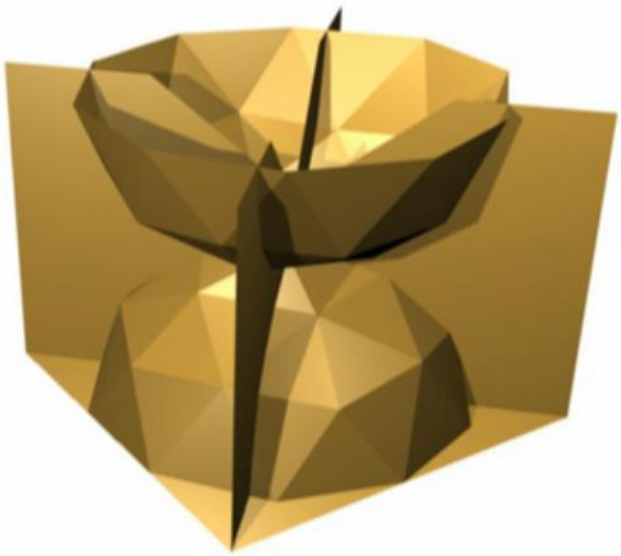
(a)

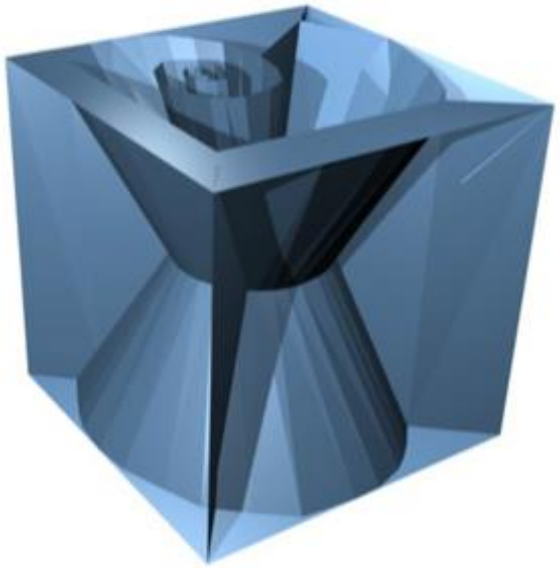
(b)

Best evolved TDRS-C antenna: (a) simulation and (b) fabricated. 111

Designing 3D solar cells (MIT)





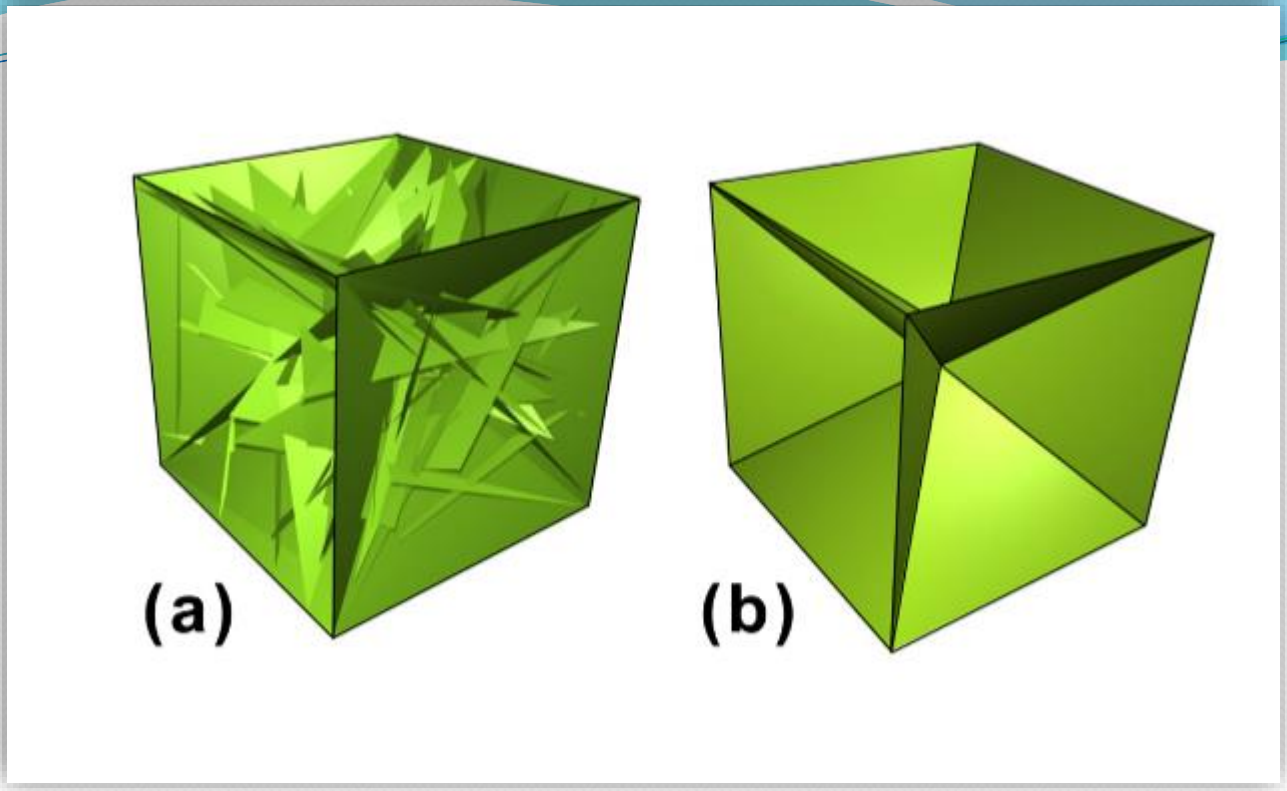


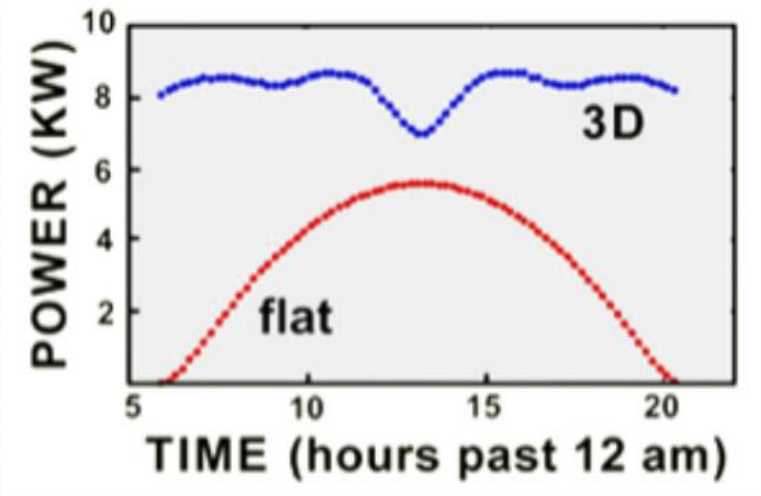
Genetic Algorithm
Millions of Evolutions

Fitness function: energy produced in a day

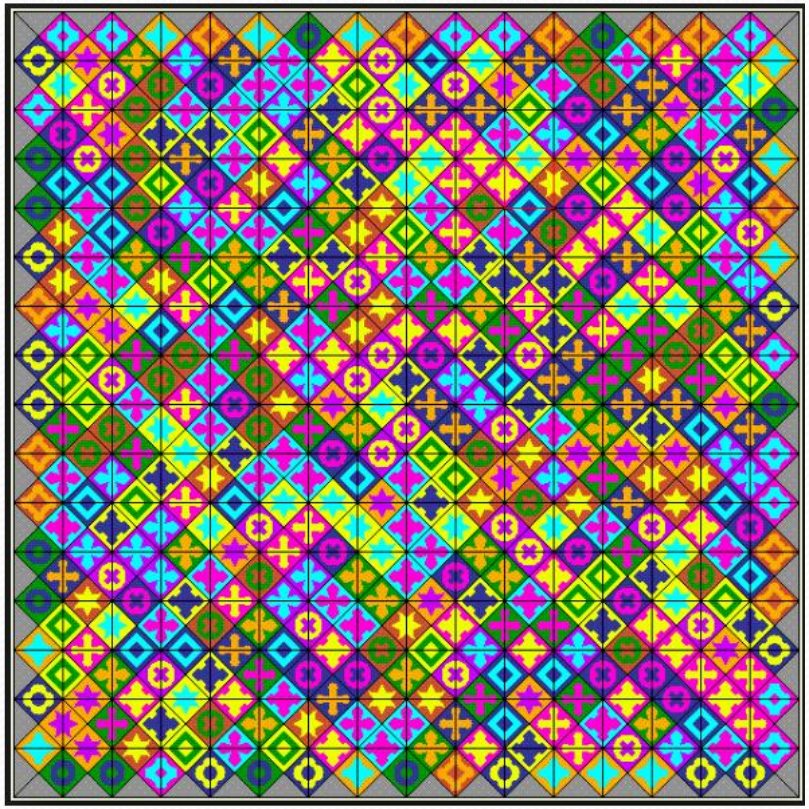
Initial Structure: Random

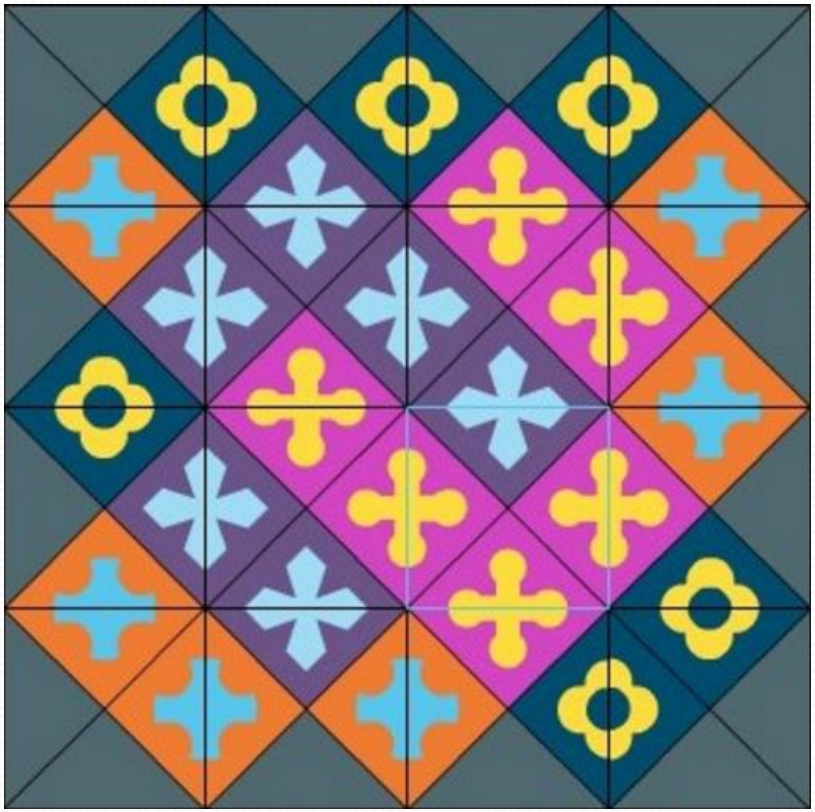
Final Structure: Emergent Order


(a)
(b)


POWER (KW)
10
8
6
4
2
3D
flat
5
10
15
20
TIME (hours past 12 am)

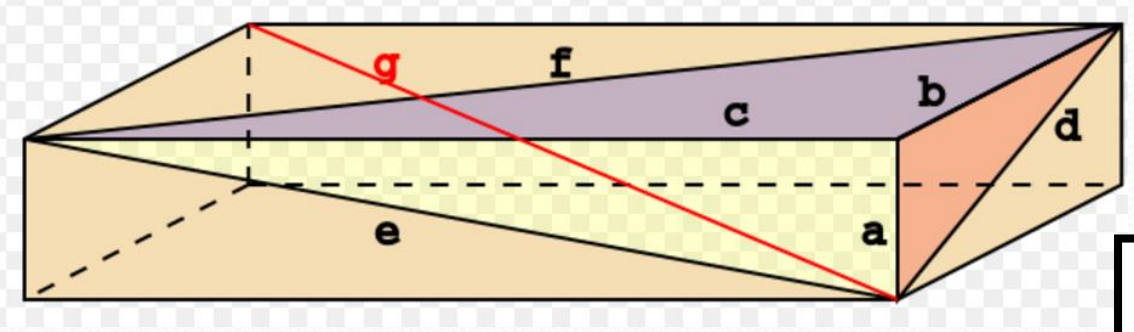
Eternity II (X256), $2M problem





What kind of the optimization problem is this?
Why is that very complex and hard to solve?

Is there any unsolved problem about a cube?

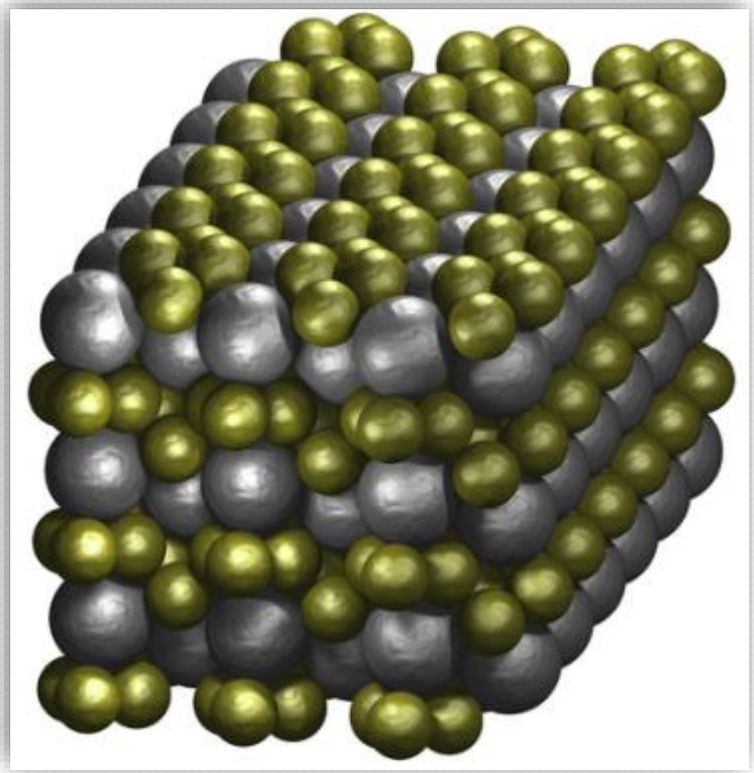


$$\begin{cases} a^2 + b^2 = d^2 \\ a^2 + c^2 = e^2 \\ b^2 + c^2 = f^2 \end{cases}$$

The smallest Euler brick, discovered by Paul Halcke in 1719, has edges (*a*, *b*, *c*) = (44, 117, 240) and face diagonals (125, 244, 267).

A perfect cuboid (also called a perfect box) is an Euler brick whose space diagonal also has integer length.

Is there any unsolved problem related to a sphere?

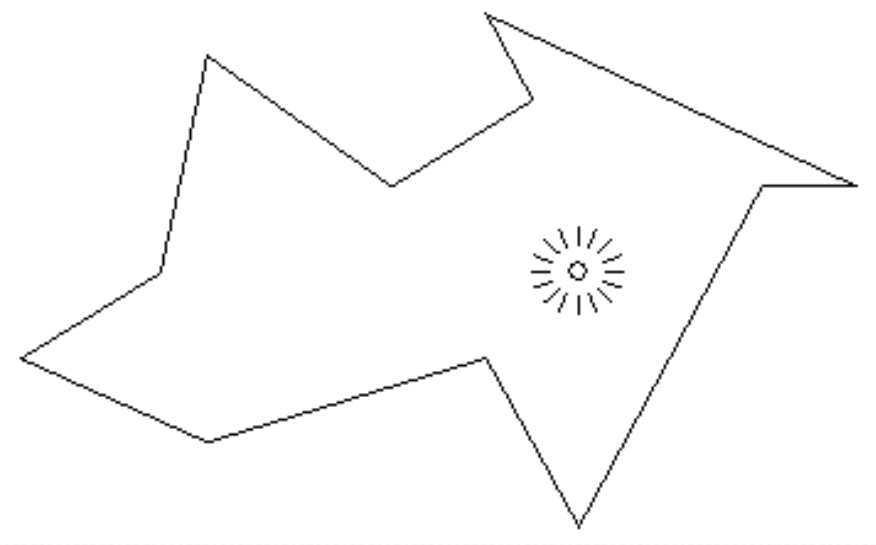


Unequal sphere packing

Is there any unsolved problem related to a polygon?

The Polygonal Illumination Problem

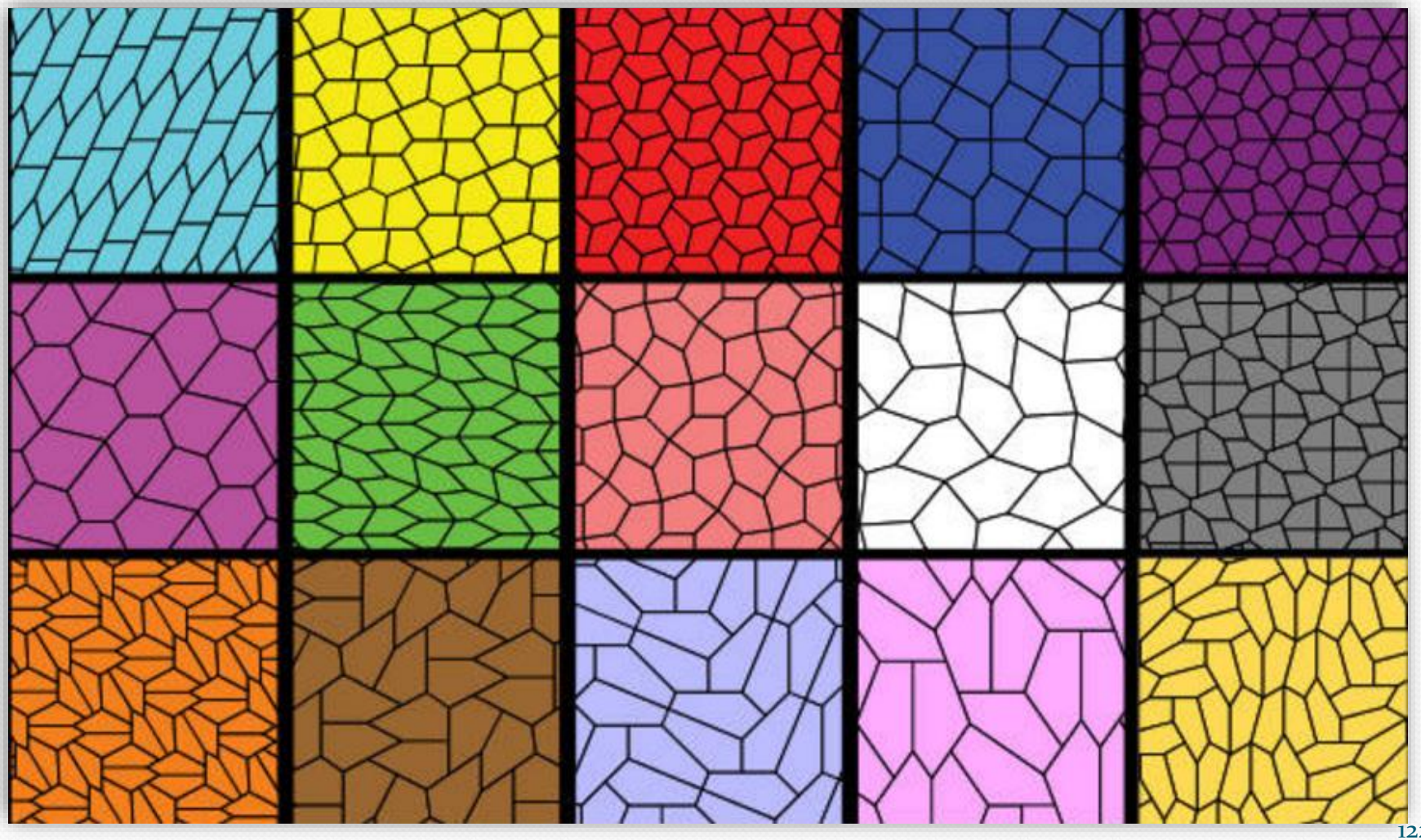
- Given a polygon S constructed with mirrors as sides, and given a point P in the interior of S, we can ask
  if the inside of S
  will be completely
  illuminated by a
  light source at P?



The Polygonal Illumination Problem (Cont.)

- For every S and P, the answer is yes.
- No counterexample is known, but no one has a proof.
- Even this easier problem is open: Does every polygon S have any point P where a light source would illuminate the interior?

Pentagonal Tiling (open problem)



Bonus Activity

- Report three unsolved problems in your major

Discovering Strassen's equations for Matrix multiplication

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

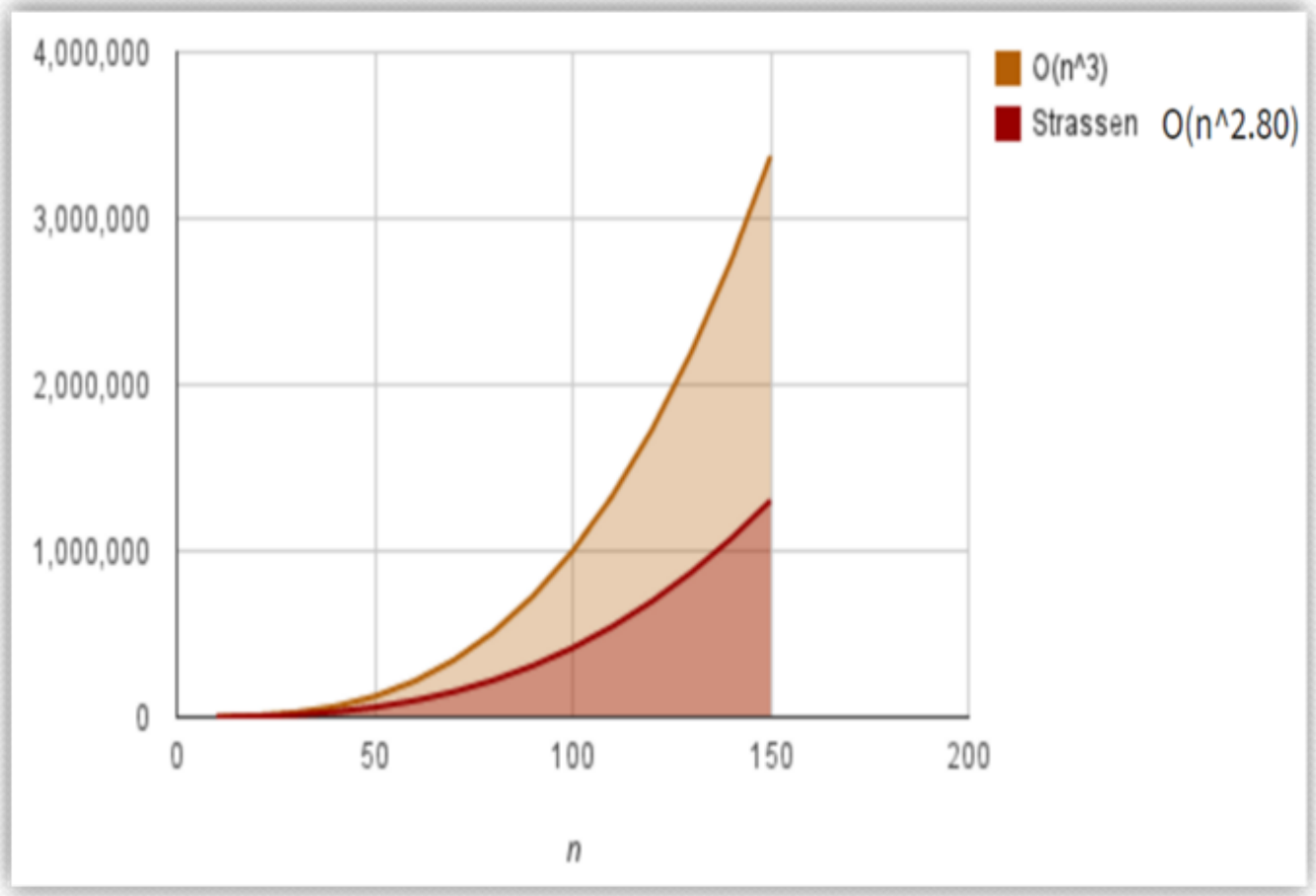
$$\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}$$
$$\mathbf{C}_{1,2} = \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2}$$
$$\mathbf{C}_{2,1} = \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}$$
$$\mathbf{C}_{2,2} = \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}$$

Discovering Strassen's equations for Matrix multiplication (Cont.)

$$
\begin{aligned}
\mathbf{M}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{M}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{M}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{M}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{M}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{M}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{M}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{1,2} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{2,1} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{2,2} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}
$$

Discovering Strassen's equations for Matrix multiplication (Cont.)



For n=100,000

(100000^3)/(100000^2.81)

=

8.9125 times faster

Oh, Seunghyun, and Byung-Ro Moon. "Automatic reproduction of a genius algorithm: Strassen's algorithm revisited by genetic search." IEEE Transactions on Evolutionary Computation 14.2 (2010): 246-251.

- "In 1968, Volker Strassen, a young German mathematician, announced a clever algorithm to reduce the asymptotic complexity of n*n matrix multiplication from the order of n^3 to n^2.81. It soon became one of the most famous scientific discoveries in the 20th century and provoked numerous studies by other mathematicians to improve upon it. …. , **and people are still curious how Strassen developed his algorithm**. …., **we found 608 algorithms that have the same quality as Strassen's**, including Strassen's original algorithm."

BQ:

How can we model the previous problem as an optimization problem?