DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Applied Computational Intelligence MEEC/MECD (2022/2023 – 1º Sem)

Evolutionary Computation

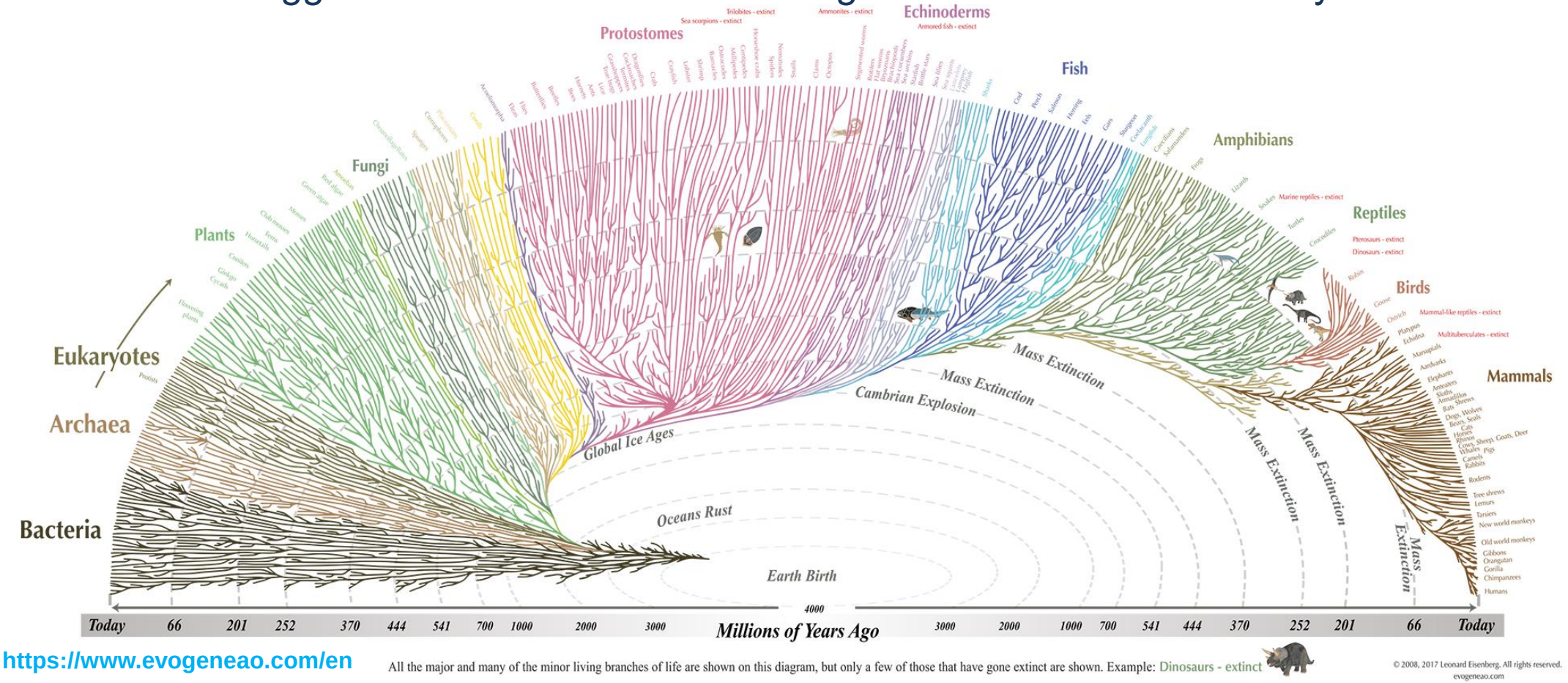
Prof. Nuno Horta

Outline

- Evolutionary Computation
  - Basic Ideas and Fundamentals
  - Evolutionary Algorithms: Generate and Test
  - Representation, Search, and Selection Operators
  - Major Research and Application Areas
  - Bibliography

Basic Ideas and Fundamentals

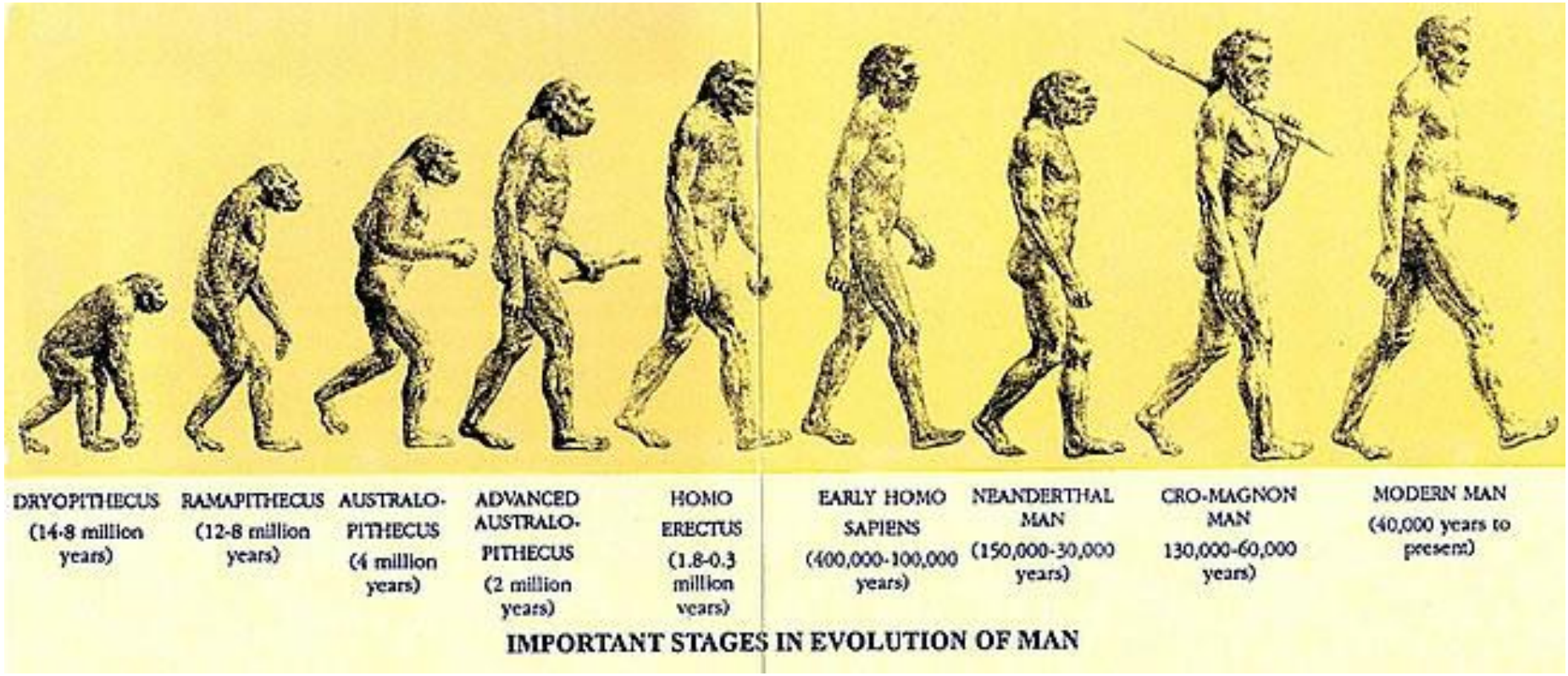
- "Evidence suggests that life has been evolving on Earth for over 3 billion years"



https://www.evogeneao.com/en

Basic Ideas and Fundamentals

- … human evolution, a few million years.



https://thehumanevolutionsite.wordpress.com/human-evolution/

Basic Ideas and Fundamentals

- **Evolution** is the keyword of a success story!

- **Challenge:** Can we mimetize the principles of natural evolution to perform computational tasks, e.g., optimizing, designing, learning, gaming?

- **Motivation**:
  - Automate the iterative process of finding optimal or quasi-optimal solutions for complex problems, e.g., problems without exact analytical description.
  - Automate the process of evolving algorithms (learning) for game playing.
  - etc.

Basic Ideas and Fundamentals

- "**Evolution** is a simple concept, but it was **resisted for three-quarters of a century after first being proposed by Charles Darwin and Alfred Russel Wallace** in 1859"

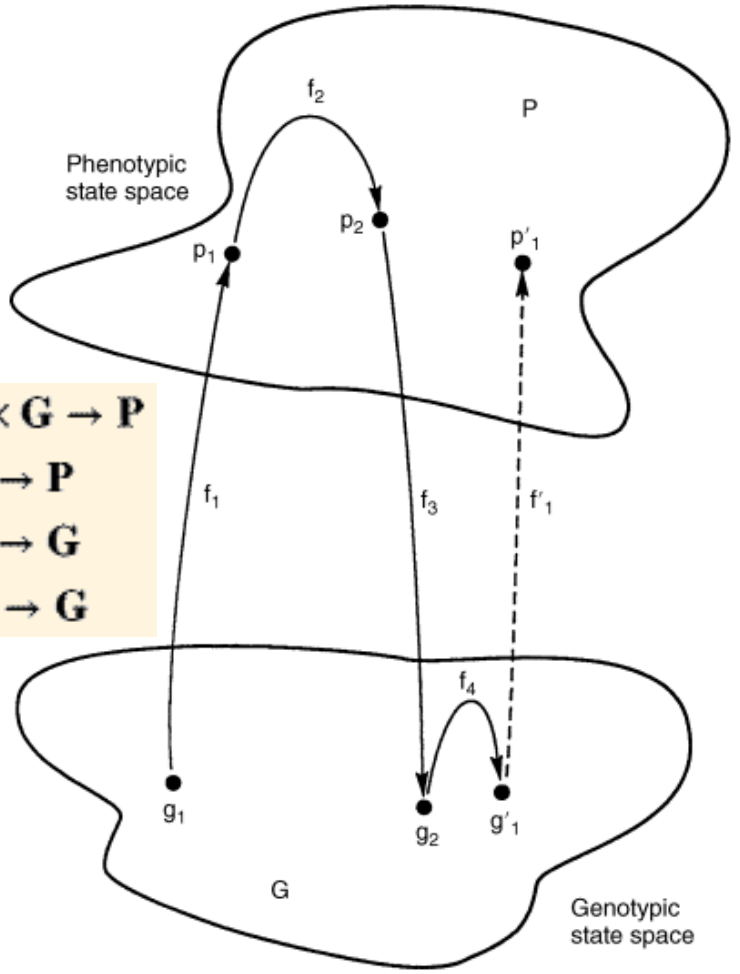
Keep an open mind!

- "Despite the simplicity of the concept, evolution is a complex process ... However, the essence of evolution—**variation and selection**—can be illustrated conveniently and cogently by way of a series of mapping functions that connect genetics with behavior.

Basic Ideas and Fundamentals

- The **genotype–phenotype** distinction is drawn in genetics. "**Genotype" is an organism's full hereditary information**. **"Phenotype" is an organism's actual observed properties, such as morphology, development, or behavior**.
- Living organisms act as a duality of their **genotype** (the underlying genetic coding) and their **phenotype** (the manner of response contained in the behavior, physiology, and morphology of the organism).
- This **genotype–phenotype pairing** may be viewed as a pairing across two state spaces, G and P, that are **informational (genetic) and behavioral (phenotypic),**
- four functions map elements in G and P to each other

$$f_1 : \mathbf{I} \times \mathbf{G} \rightarrow \mathbf{P}$$
$$f_2 : \mathbf{P} \rightarrow \mathbf{P}$$
$$f_3 : \mathbf{P} \rightarrow \mathbf{G}$$
$$f_4 : \mathbf{G} \rightarrow \mathbf{G}$$

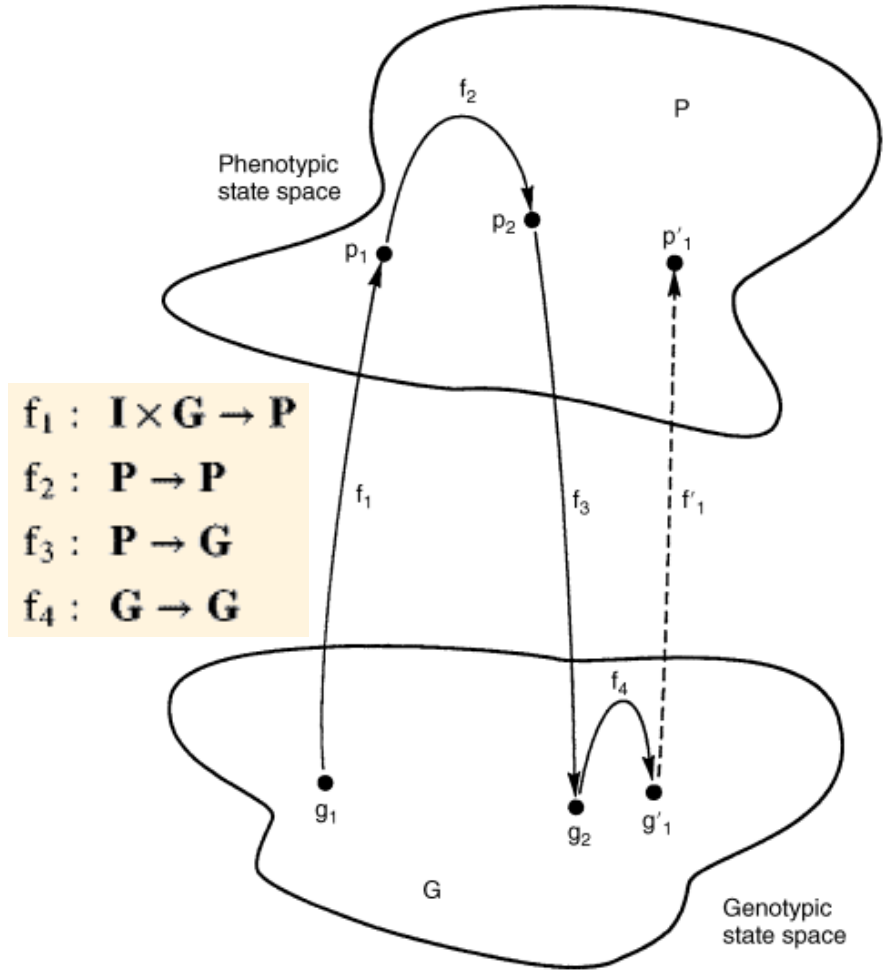


Basic Ideas and Fundamentals

- The evolution of a population within a single generation. **Evolution can be viewed as occurring as a succession of four mapping functions** — epigenesis, selection, genotypic survival, and mutation — relating the genotypic information state space and the phenotypic behavioral state space.

- **Example**:
  - Consider a **Genotypic State Space** defined as a 2-dimensional space defined by x, y $\in$ [-4,4]
  - Consider a collection of 8 elements from that 2-dimensional space, e.g.:

$g_1 = \{(-4,4), (-3,1), (-2,-2),(-2,3) ,(1,1) ,(2,-1) ,(3,2) ,(3,-3)\}$



Basic Ideas and Fundamentals

- The function **f1** is called **epigenesis**. This maps an element **g1** and indexed set of symbols **(i1,…,ik)∈I**, where I is the set of all such environmental sequences into the **phenotypic space P** as a particular **collection of phenotypes p1**, whose development is modified by its environment.

- **Example**:
  - Consider

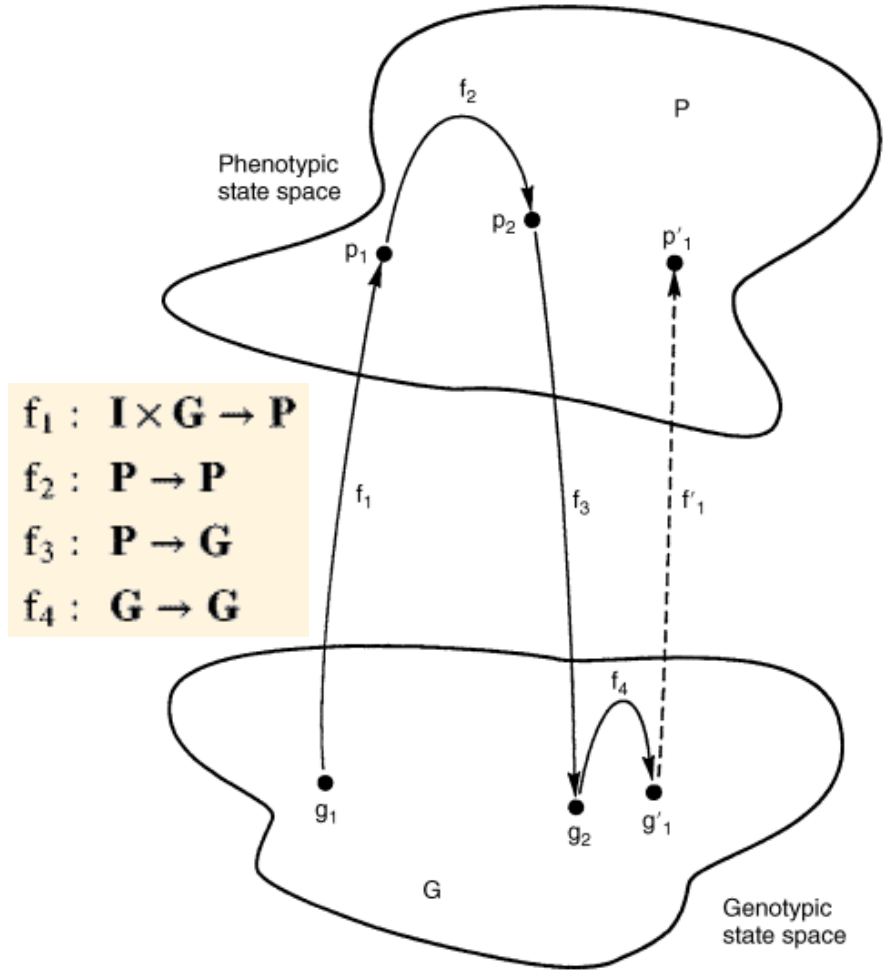
    $g_1 = \{(-4,4), (-3,1), (-2,-2),(-2,3) ,(1,1) ,(2,-1) ,(3,2) ,(3,-3)\}$

  - Consider a **function f1** defined as:

    $f_1(x,y) = x^2 + y^2$

  - The collection of phenotypes would be:

    $p_1 = \{32, 10, 8, 13, 2, 5, 13, 18\}$



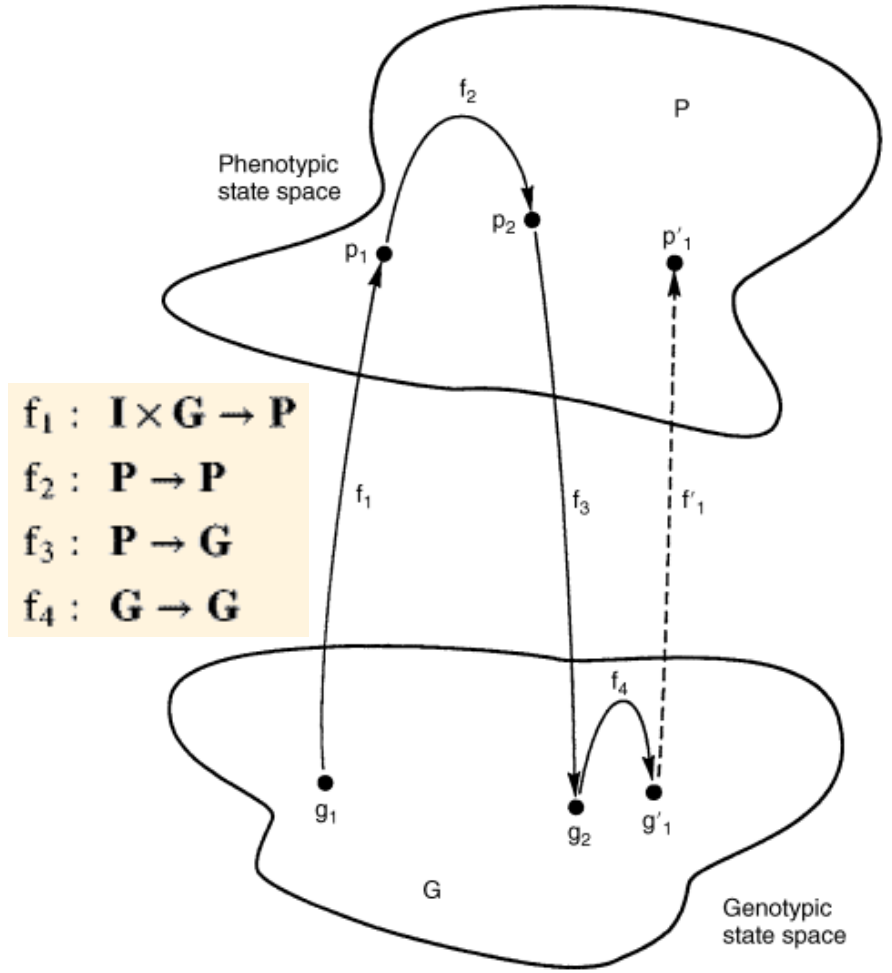
Basic Ideas and Fundamentals

- The function **f2**, **selection**, **maps phenotypes p1 into p2**. As natural selection operates only on the phenotypic expressions of the genotype.

- **Example**:
  - Consider a **function f2** which selects just the best half of the collection of phenotypes (assuming a minimization problem)

  - The selected collection of phenotypes would be:

    $p_2 = \{10, 8, 2, 5\}$



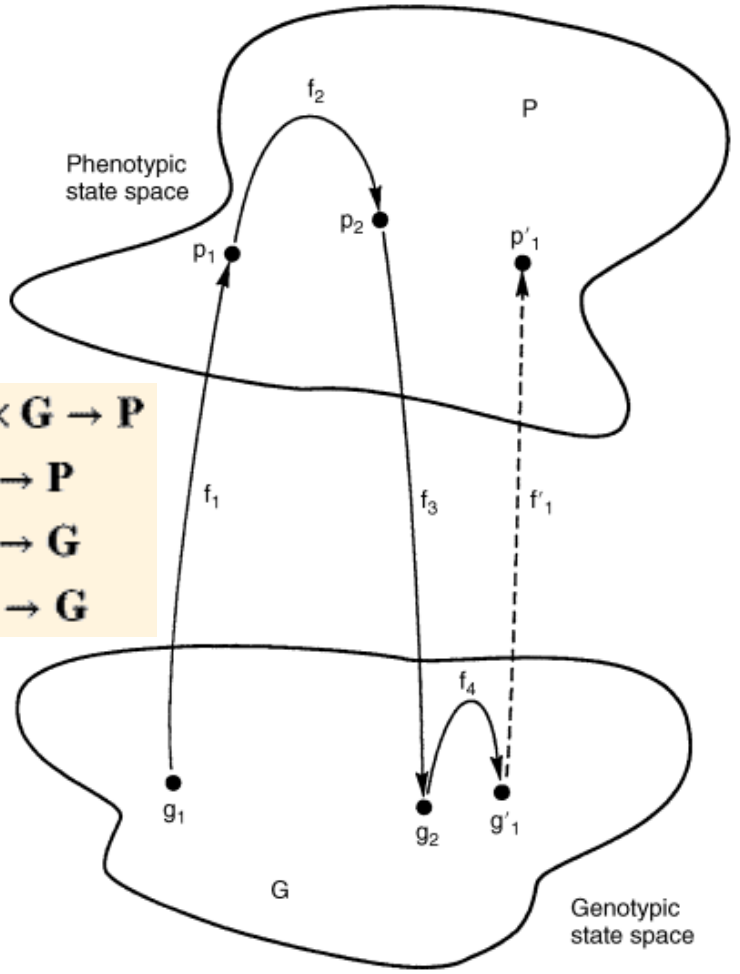
Basic Ideas and Fundamentals

- Function **f3**, **genotypic survival**, describes the effects of selection and migration processes on G. For those phenotypes that survive the selection function f2, **there is a collection of genotypes g2 that correspond to those surviving phenotypes.** (The collection g2 differs from g1 because selection and migration have removed individuals from g1.)

- **Example**:
  - The new collection of genotypes g_1 is given by:

    $g_2 = \{ (-3,1), (-2,-2), (1,1), (2,-1) \}$

$f_1 : \mathbf{I} \times \mathbf{G} \rightarrow \mathbf{P}$
$f_2 : \mathbf{P} \rightarrow \mathbf{P}$
$f_3 : \mathbf{P} \rightarrow \mathbf{G}$
$f_4 : \mathbf{G} \rightarrow \mathbf{G}$

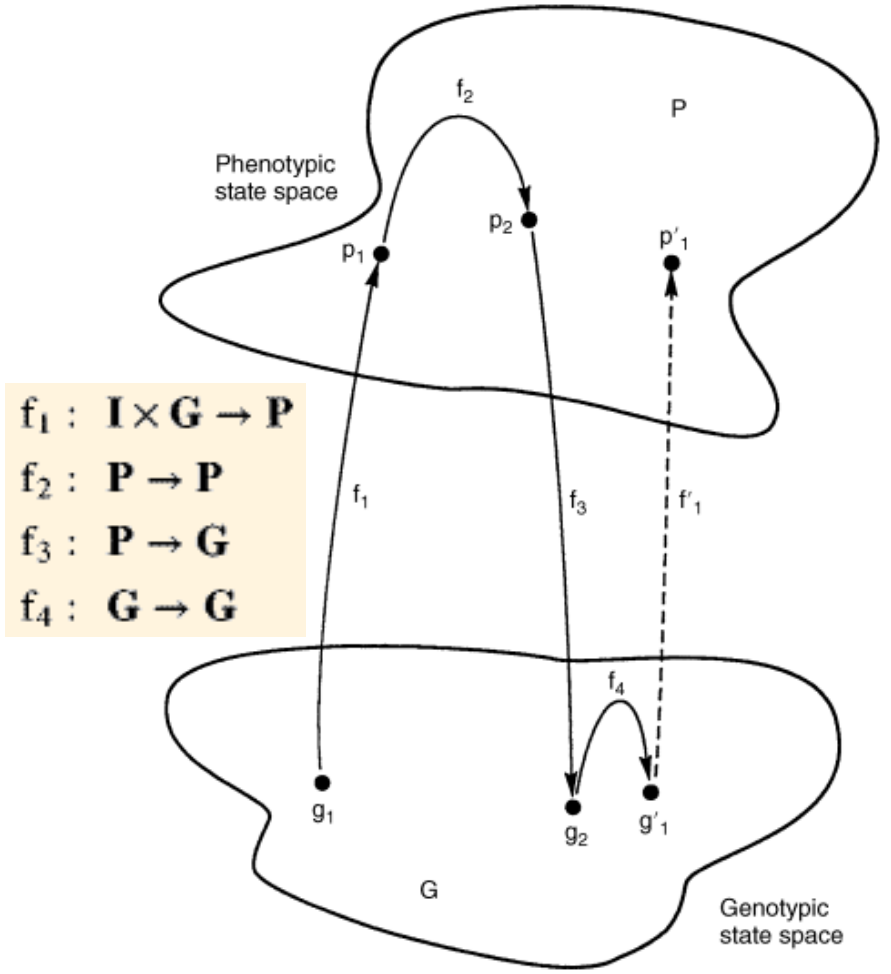


Basic Ideas and Fundamentals

- Function **f4**, **mutation**, which maps the representative codings **g2∈G** to the point **g'1∈G**. This function represents the "rules" of genetic variation—**individual gene mutations, recombination, inversion, and so forth**—that occur in the creation of offspring from parents. With the creation of a new population of genotypes **g'1**, one generation is complete. Evolutionary adaptation occurs over successive iterations of these mappings.

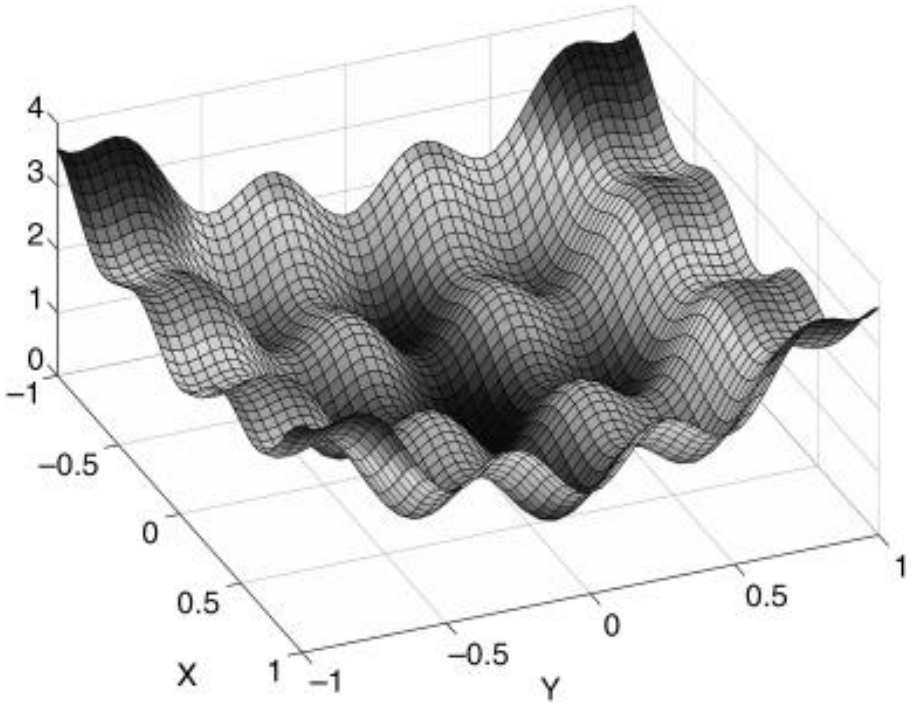
- **Example**:
  - The new collection of genotypes g'_1 is given by:

$g'_1 = \{ (-3,1), (-2,-2), (1,1), (2,-1), (-1,1), (4,-1), (2,3), (-2,-3) \}$



Evolutionary Algorithms: Generate and Test

- The **simplest evolutionary algorithm** can be viewed as a **search procedure that:**
  1. **generates potential solutions to a problem,**
  2. **tests each for suitability,**
  3. **and then generates new solutions**.
- It's important to understand how this process differs from **exhaustive search** or **blind random search**.



Evolutionary Algorithms: Generate and Test

- **Exhaustive Search**

Assume:

S – sample space

s_1 … s_k can be selected from S

where the cardinality of S can be Infinite

Consider the objective of finding the

Maximum of f(s)

The search consists of **evaluating all possible solutions**.

Comment:

the **procedure is certain to find the best solution** but time to find it may be prohibitive.

Exhaustive Search

```
i = 1;
best = i;
bestscore = f(s(i));
repeat
   i = i + 1;
   if f(s(i)) > bestscore then
   begin
      best = i;
      bestscore = f(s(i));
   end
until (i == k);
```

Evolutionary Algorithms: Generate and Test

- **Exhaustive Search (2)**

"… many practical problems pose search spaces that are **transcomputationally large**, meaning that the size of the space is greater than 10^{100}.

There are only about 10^{18} s in the history of the universe, so in these cases you could examine 10^{80} solutions every second, and still not be done in over 13 billion years! …

after another 13 billion years, **no one would be around to applaud you for finding the solution** 😎

Exhaustive Search

```
i = 1;
best = i;
bestscore = f(s(i));
repeat
   i = i + 1;
   if f(s(i)) > bestscore then
   begin
      best = i;
      bestscore = f(s(i));
   end
until (i == k);
```

Evolutionary Algorithms: Generate and Test

- **Exhaustive Search (3)**

A transcomputational problem is a problem that requires processing of more than 10^{93} bits of information. Any number greater than 10^{93} is called a transcomputational number.

Example: A system of **n variables**, each of which can take **k different states**, can have k^n possible system states. The problem becomes transcomputational when $k^n > 10^{93}$. This happens for the following values of k and n:

k	2	3	4	5	6	7	8	9	10
n	308	194	154	133	119	110	102	97	93

Exhaustive Search

```
i = 1;

best = i;

bestscore = f(s(i));

repeat

   i = i + 1;

   if f(s(i)) > bestscore then

   begin

      best = i;

      bestscore = f(s(i));

   end

until (i == k);
```

Evolutionary Algorithms: Generate and Test

- **Blind Random Search**

Consider searching at random, blindly choosing a series of possible solutions $s_1,\ldots, s_n$ from S, where the sequence of length $n \leq k$ is chosen uniformly without replacement.

Mathematically, this means that for the first sample s_1, each solution in S has a 1/k probability of being selected. After s_1 is selected, every remaining solution in S has a 1/(k – 1) probability of being selected as s_2, and so forth.

Comment: The procedure can be performed in a reasonable time by selecting n appropriately. However, it usually performs poorly on real-world problems.

Note: Both Exhaustive and **Blind Random** search **choose the next solution without regard to what has been chosen previously**.

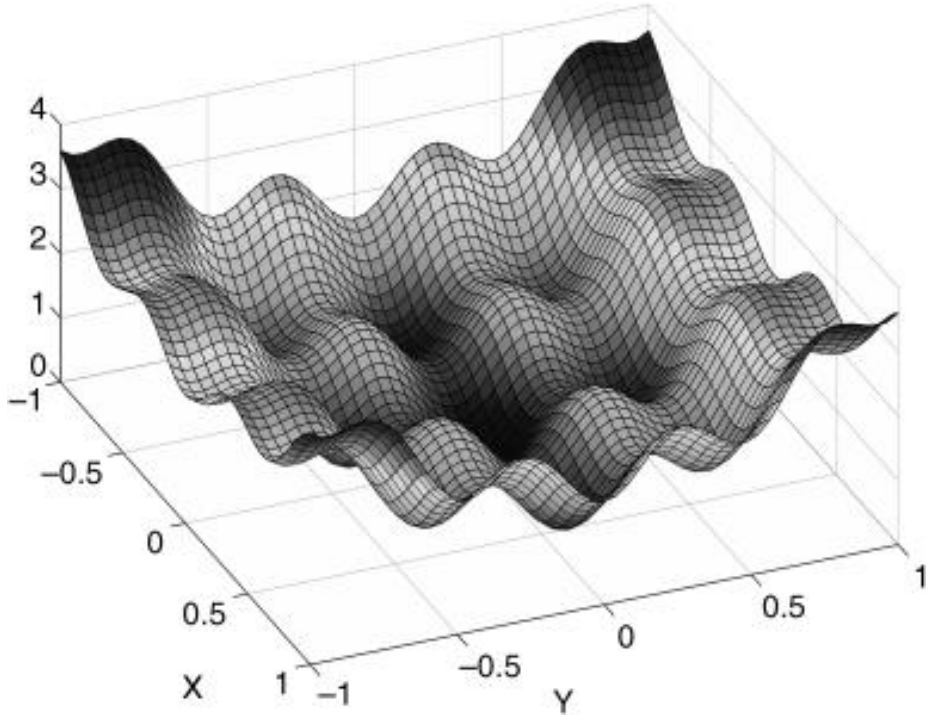
Blind Random Search

```
i = 1;
selected = U(1, k); //chosen uniformly
best = selected;
bestscore = f(s(selected));
repeat
   i = i + 1;
   remove s(selected) from S;
   selected = U(1, k - i + 1); //chosen
   if f(s(selected)) > bestscore then
   begin
      best = selected;
      bestscore = f(s(selected));
   end
until (i == n);
```

Evolutionary Algorithms: Generate and Test

- **Traditional Search Procedures**

- **Traditional search procedures** described in mathematics **are not blind**
- They explore for a maximum (or minimum) by utilizing **information from the function being searched**
- Often this **involves use of the gradient or higher order statistics that allow a search algorithm to move rapidly in a direction that is presumed to be beneficial**



Comment: These procedures can **converge quickly on a maximum or minimum**, but run the risks of stalling at a **saddle point**—where the gradient is zero—or becoming trapped in **maxima or minima that are only optimal locally**.

Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm** vs **Traditional Gradient Methods**

Evolutionary algorithms **operate in two ways that are fundamentally different from traditional gradient methods**.

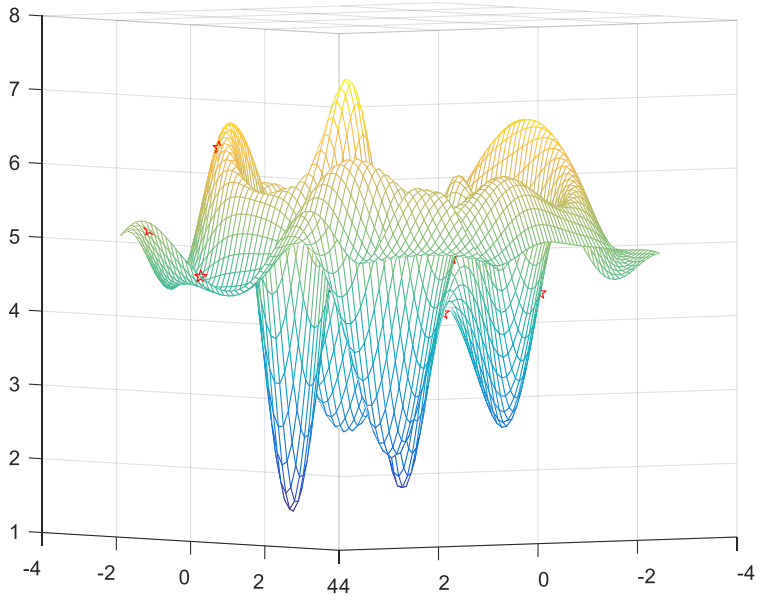
- First, rather than executing a point-to-point search, they incorporate a **population of solutions, each individual solution competing for survival.**
- Second, instead of utilizing gradient information from the response surface being searched, they utilize **random variation to explore for new solutions**.

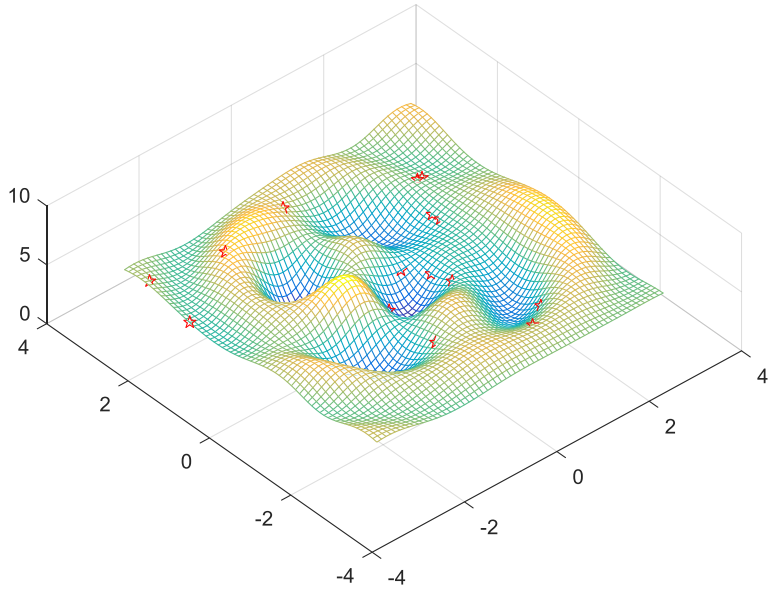
Evolutionary Algorithm

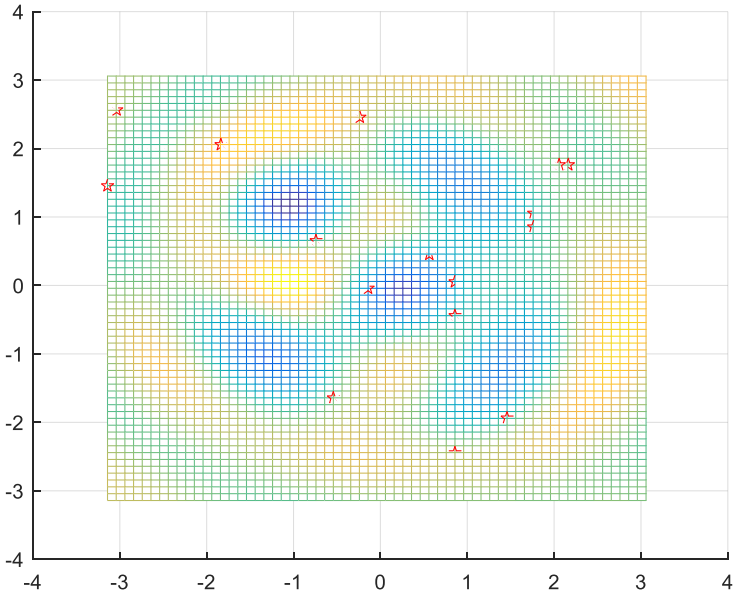
```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

E01: From a Random Initial Population to a Global Minimum Solution

- 3 different views of the initial Population

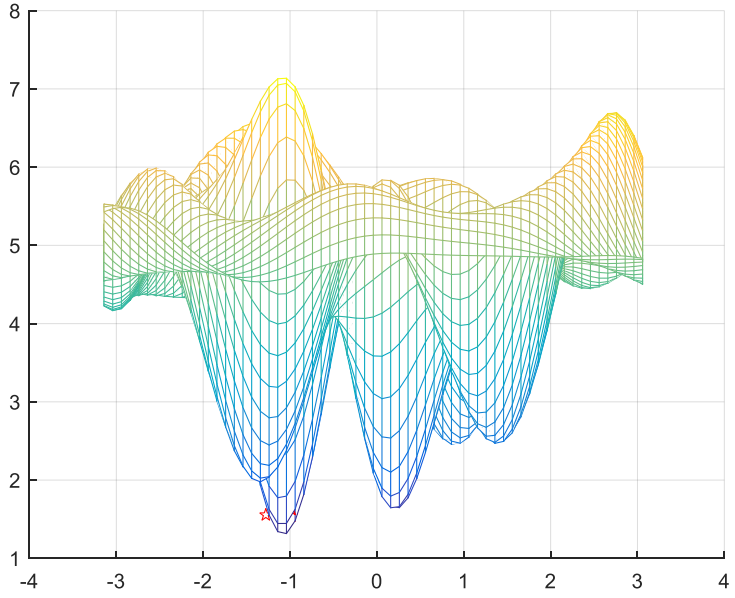


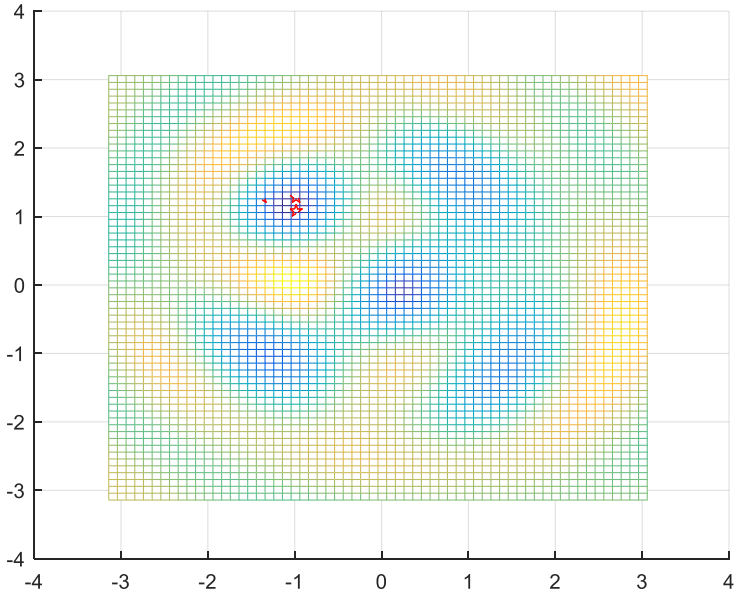


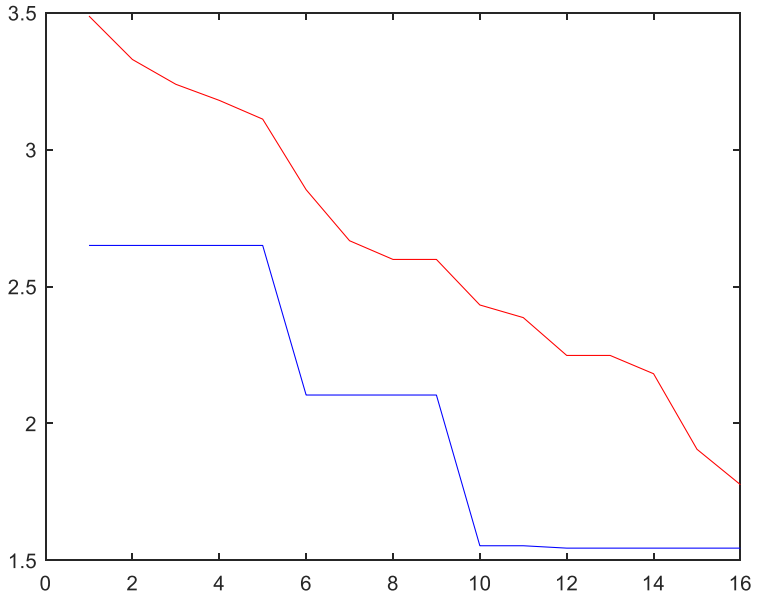


E01: From a Random Initial Population to a Global Minimum Solution

- Appropriate operators (**to be discussed later**) are used to evolve the population in the desired direction.
- Results after 16 generations ... Population moved towards global minimum

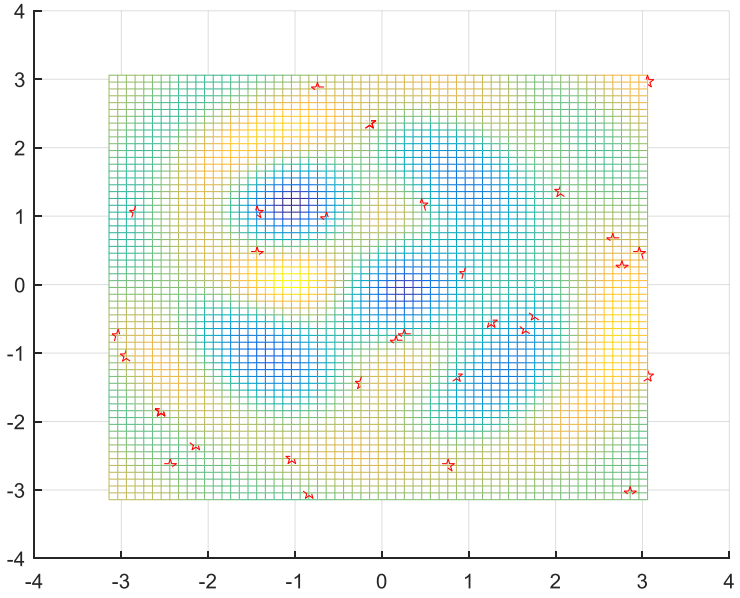


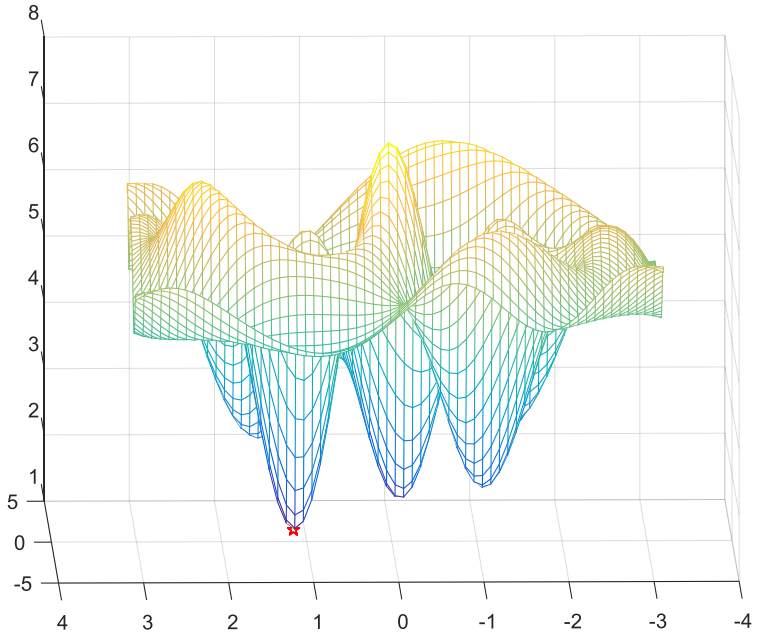


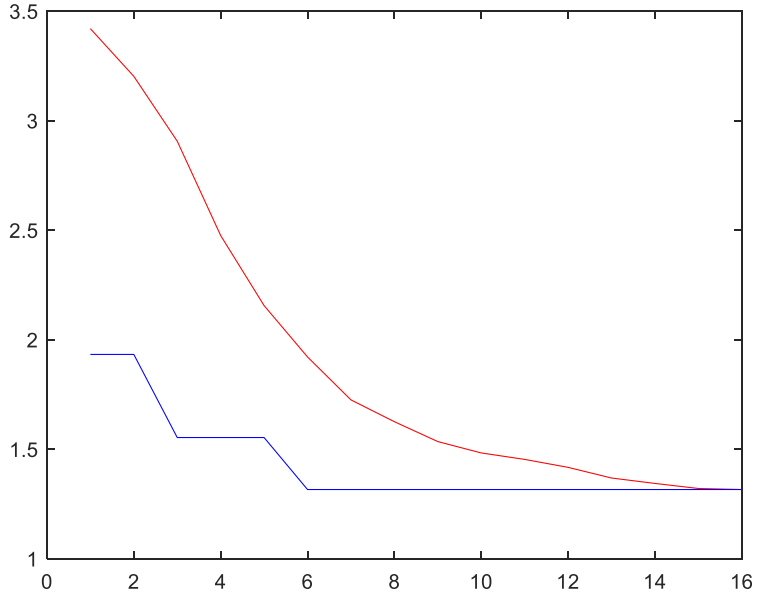


E01: From a Random Initial Population to a Global Minimum Solution

- Larger population increases the change of reaching the minimum in the same number of generations but the computational cost is higher … particularly, when the evaluation requires time consuming simulations (**to be discussed later**)







Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm**

The relationship between parent and offspring defines **inheritance**. **If there is no inheritance, then the procedure is essentially like a blind random search** conducted multiple samples at a time.

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

Evolutionary Algorithm

Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm**

Suppose the value of **an offspring is related to its parent as follows**:

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

where x' is the offspring of x and N(0, σ) is a Gaussian random variable with zero mean and standard deviation σ.

- If $\sigma \to \infty$ then the search is again like a **blind random search**.
- If the $\sigma \to 0$ then there is **complete inheritance** from parent to offspring and there's no search at all because **each offspring merely replicates its parent**.
- But with $0 < \sigma < \infty$ the procedure becomes a stochastic parallel search with many interesting mathematical properties **(to be discussed later).**

Evolutionary Algorithms: Generate and Test

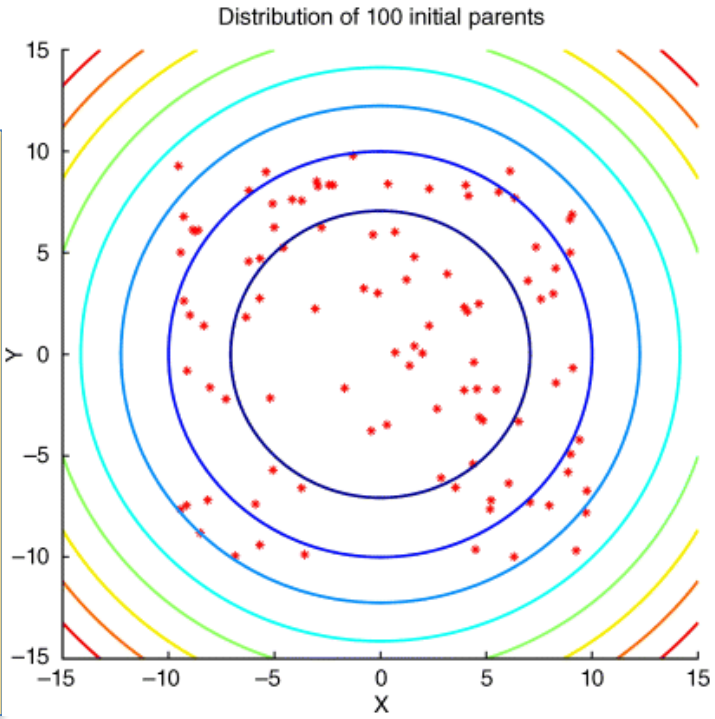
- **Evolutionary Algorithm**

The function is convex and continuous, thus gradient methods would be more appropriate (the Newton–Gauss method would be most appropriate). But the function will serve for instructional purposes.

$$f(x, y) = x^2 + y^2$$

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

Evolutionary Algorithm



Evolutionary Algorithms: Generate and Test

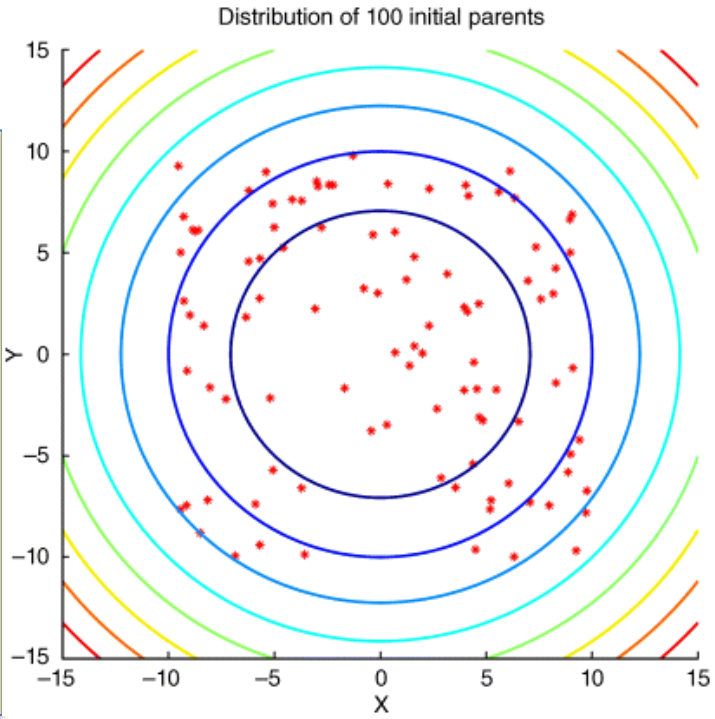
- **Evolutionary Algorithm**

Start with a population of 100 candidate solutions that chosen uniformly at random from the range:

$$(x, y) \in (-10, 10)^2.$$

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

Evolutionary Algorithm



Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm**

In this very basic approach, we'll have **each parent generate one offspring by mutating the (x,y)** coordinates. **Mutation** is accomplished by adding a **Gaussian random variable with zero mean and a fixed standard deviation of 0.01** to the x- and y-coordinates of the parent.

After the creation of 100 offspring, we **choose 100 best solutions** from among the parents and offspring to be parents for the next generation.

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

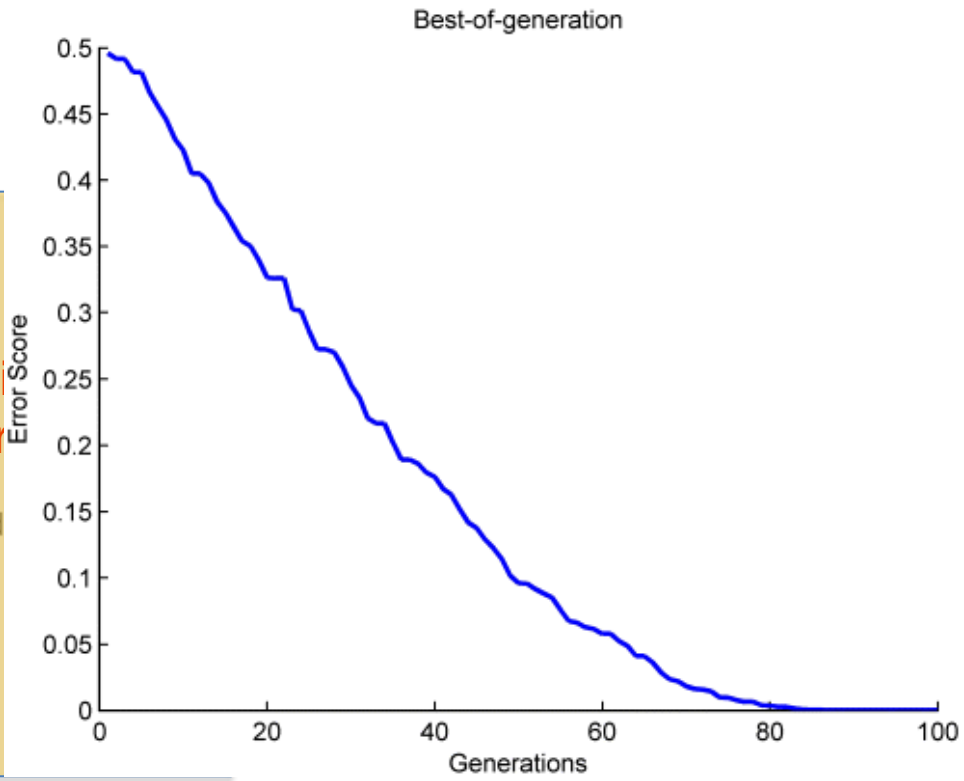
Evolutionary Algorithm

Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm**

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of ind
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

Evolut
Algor



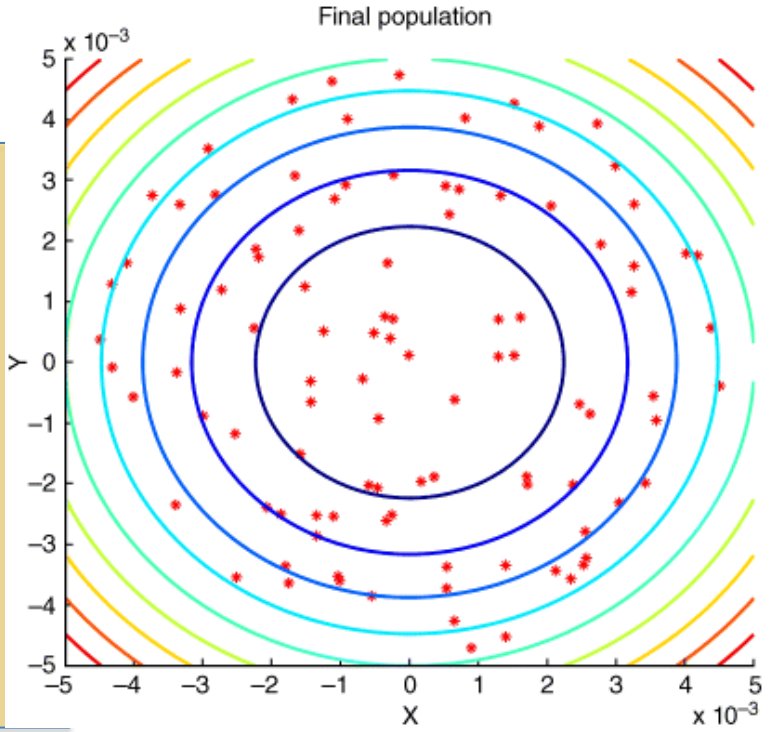
Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm**

Solutions are distributed from −0.005 to 0.005 in each dimension, confined in a box that's 0.01 units on each side. A box that is 0.01 on a side has an area that is 2.5×10^{-7} of the area of the initial box we started with, which was 20 units on a side.

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

Evolutionary Algorithm



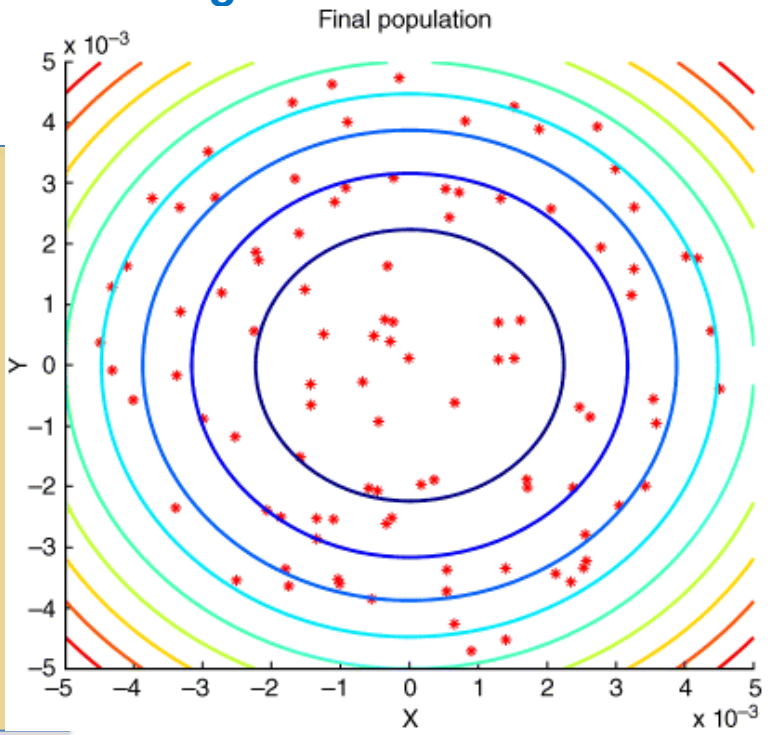
Evolutionary Algorithms: Generate and Test

- **Evolutionary Algorithm**

One way of judging the improvement that the evolutionary algorithm offers in this case is that it that **narrows the area of the best answer by a factor of over 1 million in 100 generations**

```
g = 0; // g is for generation number
choose initial population; //often this is selected at random from S
repeat
    evaluate population; //assign a score to each individual
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; //based on the parents, create offspring
    g = g + 1;
until (g == max)
```

Evolutionary Algorithm



Representation, Search, and Selection Operators

- **Evolutionary algorithms** are often viewed in terms of:
  - the **data structures** that are used **to represent solutions**,
  - **the search operators** that are **applied to those data structures**,
  - and **the selection operators** that **determine which solutions** in a population will influence the creation of the next generation.

- **Representation**:
  - Choosing a **representation** for a problem **must be done in light of what search operators** (i.e., variation operators) **are going to be applied to that representation**

Representation, Search, and Selection Operators

- **Selection**:
  - **Selection can vary in effect** from **strong to weak**. **Strong selection ensures that only the very best solutions in a population will serve as parents** for the next generation. **Weak selection offers nonzero probability that weaker solutions may also serve as parents.**
- **Search**:
  - Some of the more common **search operators** include **mutation functions** that apply to a single parent, and **recombination functions** that apply to two or more parents. One special case of recombination is crossover, which takes contiguous segments and swaps them between two parents.

- **Comment**: **Search and selection are intertwined in effects**. A very **narrow search can be made effectively broader by using weak selection**. This would allow subpar solutions to become parents, thus expanding the application of the narrow search operation. **Similarly, a broad search can be made effectively narrower to some extent by using strong selection**.

Major Research and Application Areas

- **Optimization**
  - The most common application of evolutionary algorithms comes in optimization. Applications range in diversity, including biomedical pattern classification, industrial scheduling, financial forecasting, video game character control, and many others, e.g., Computational Finance (optimizing an investment strategy)

- **Design**
  - Design is often connected intrinsically to optimization. Suppose there is an electronic circuit that must be designed to fulfill a particular function, e.g., design of analog IC designs with AIDAsoft (www.aidasoft.com).

- **Learning and Games**
  - Learning often refers to the case of adjusting strategies for accomplishing a task based on feedback about the quality of performance being generated. For example, creating new investment strategies or new circuit topologies. Evolutionary algorithms can be used to adjust the weights between nodes in a neural network as well as the topology of the network simultaneously

Basic Ideas and Fundamentals

- **Applied EC Examples: (AIDAsoft - Analog IC Design)**

AIDAsoft
Analog IC Design Automation

Home | Solutions | Publications | Opportunities | About Us

:: AIDA Framework

- Fully automatic design or semi-automatic allowing designer intervention.
- Supported System: Cent OS 6&7, Linux RedHatEnterprise 5&6

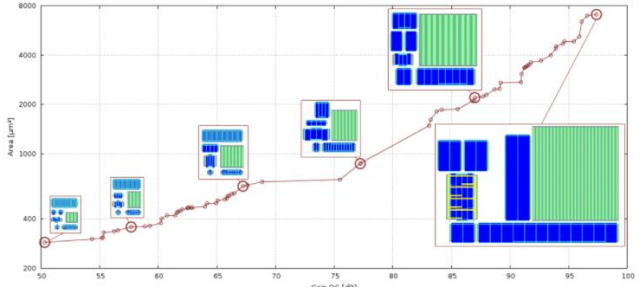
:: AIDA-C

- Uses standard electrical circuit simulators as evaluation engines.
- Handles multiple circuit test-benches simultaneously.
- Performs geometric and parasitic layout-aware circuit-level multi-objective

Solutions

The **AIDA framework** implements an automatic analog IC design flow from a circuit-level specification to a physical layout description. The circuit-level synthesis is done by **AIDA-C**, and after the circuit-level design, **AIDA-L** takes the device sizes and the best floorplan, and generates the complete layout, which is then saved as a GDSII stream format.

AIDA-C is a circuit-level synthesizer supported by state-of-the-art multi-objective optimization kernels, where the robustness of the solutions is attained by considering user-defined worst case corners, that account for process variations and(or) PVT corners. The circuit's performance is measured using **Spectre®**, **Eldo®** or **HSPICE®** electrical circuit simulators.



Pareto optimal solution set for Area vs. DC Gain optimization with superimposed circuit layouts.

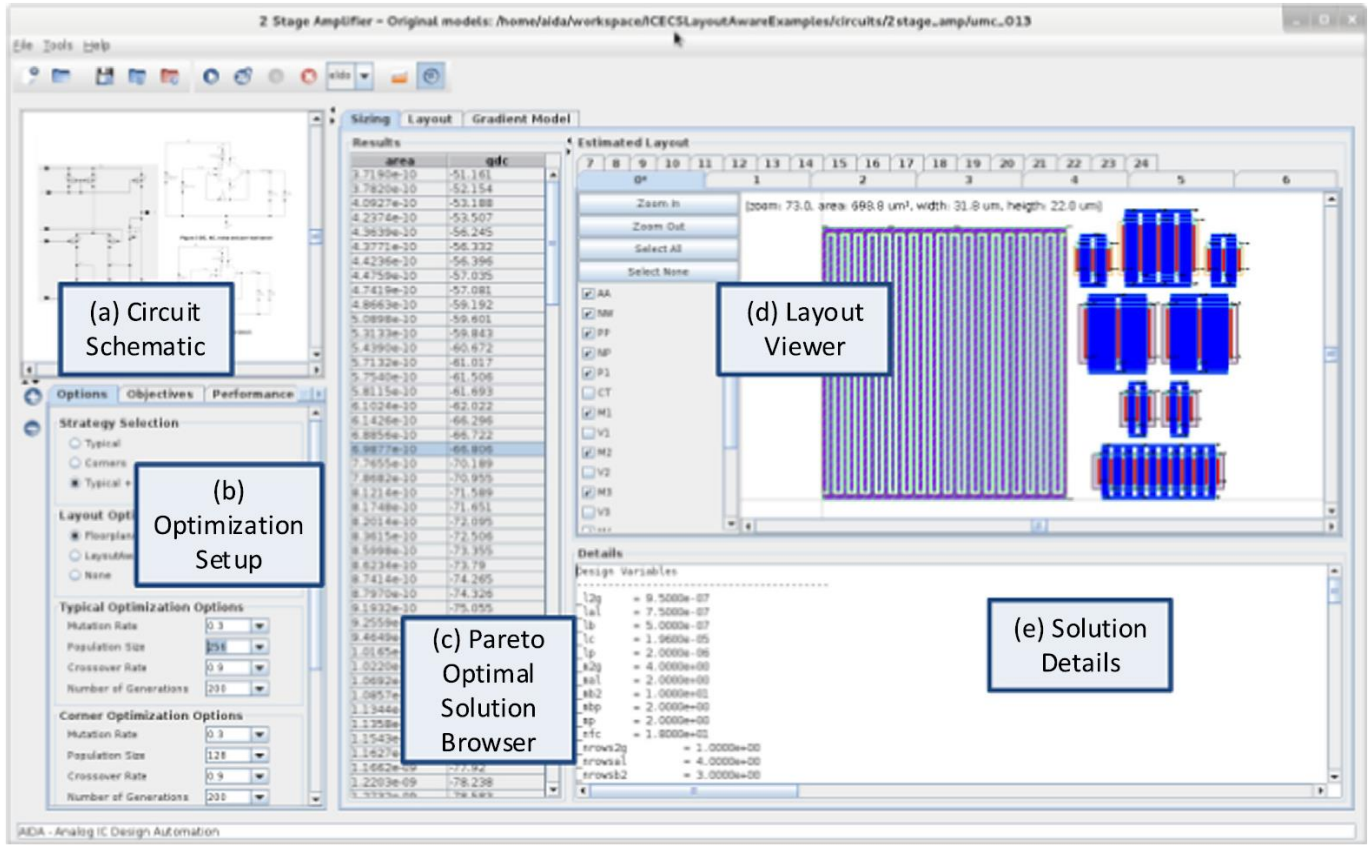
- **EC** for Integrated Circuit Optimizal Sizing
- **EC** for Integrated Circuit Layout
- **AIDAsoft** fully developed at IT and with industrial licensing.

Available MSc. Thesis proposals on the topic, supervised by:
N. Lourenço, R. Martins, N. Horta

http://www.aidasoft.com

Basic Ideas and Fundamentals

- **Applied EC Examples: (AIDAsoft - Analog IC Design)**



http://www.aidasoft.com/Home

- **EC** for Integrated Circuit Optimizal Sizing
- **EC** for Integrated Circuit Layout
- **AIDAsoft** fully developed at Instituto de Telecomunicações (IT) and with industrial licensing.

Available MSc. Thesis proposals on the topic, supervised by:
N. Lourenço, R. Martins, N. Horta

Basic Ideas and Fundamentals

- **Applied EC Examples: (AIDAsoft - Analog IC Design)**

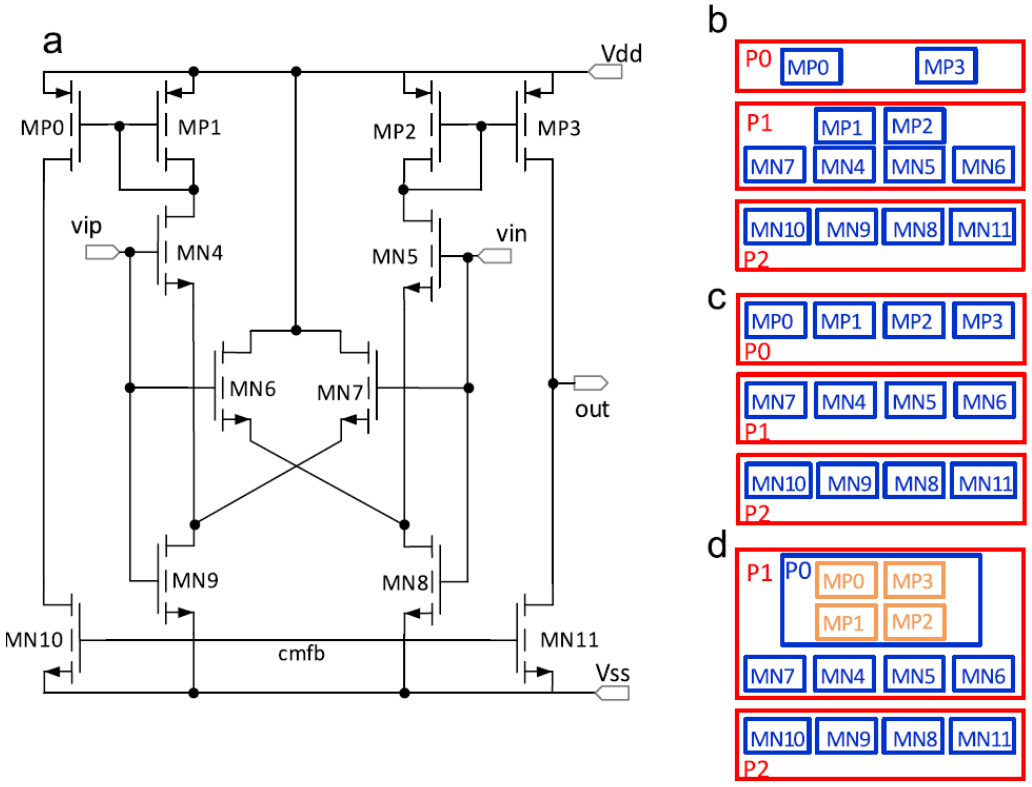


Fig. 17. Single stage amplifier with gain enhancement using voltage combiners.

Search Space Definition

Table 4
Single stage amplifier with gain enhancement using voltage combiners variable and ranges.

Variable (Unit)	Min.	Grid resolution	Max.
l0, l1, l4, l6, l8, l10 [nm]	120	10	1000
w0 w1 w4 w6 w8 w10 [μm]	1	0.1	10
nf0, nf1, nf4, nf6, nf8, nf10	1	2	8

The variables l0, w0, and nf0 are dimensions of MP0 and MP3; l1, w1, and nf1 of MP1 and MP2; l4, w4 and nf4 of MN4 and MN5; l6, w6 and nf6 of MN6 and MN7; l8, w8, and nf8 of MN8 and MN9; l10 w10 and nf10 of MN10 and MN11.

Objectives and Constraints

Table 5
Single stage amplifier with gain enhancement using voltage combiners' specifications.

Specifications	Measure	Target	Units	Description
Objectives	Idd	Minimize	μA	Current consumption
	Area	Minimize	μm^2	Area
	GBW	Maximize	MHz	Unity gain frequency
Constraints	Idd	≤ 350	μA	Current consumption
	GDC	≥ 50	dB	Low-frequency gain
	GBW	≥ 30	MHz	Unity gain frequency
	PM	≥ 60	°	Phase margin
	FOM	≥ 1000	MHz.pF/mA	$FOM = \frac{Gbw \times C_{Load}}{Idd}$
	ov[a]	≥ 50	mV	Overdrive voltages ($V_{GS}-V_{TH}$)[b]
	ov[c]	≥ 100	mV	Overdrive voltages ($V_{GS}-V_{TH}$)[c]
	d[a,b]	≥ 50	mV	Saturation margin of the PMOS device ($V_{DS}-V_{DSat}$)[c]

[a] Applies to: MP0, MP1, MP2 and MP3.
[b] Applies to: MN4, MN5, MN6, MN7, MN8, MN9, MN10 and MN11.
[c] For PMOS devices the overdrive is Vt – Vgs and delta is Vdsat – Vds.

Basic Ideas and Fundamentals

- **Applied EC Examples: (AIDAsoft - Analog IC Design)**

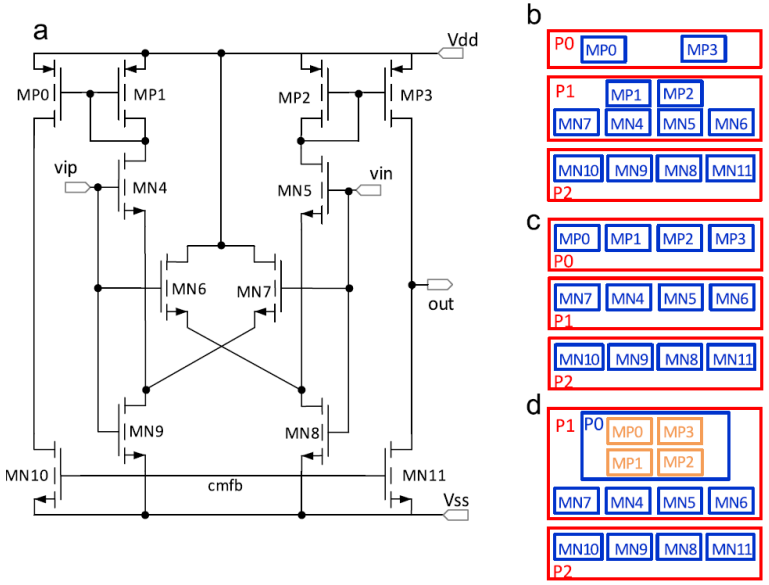


Fig. 17. Single stage amplifier with gain enhancement using voltage combiners.

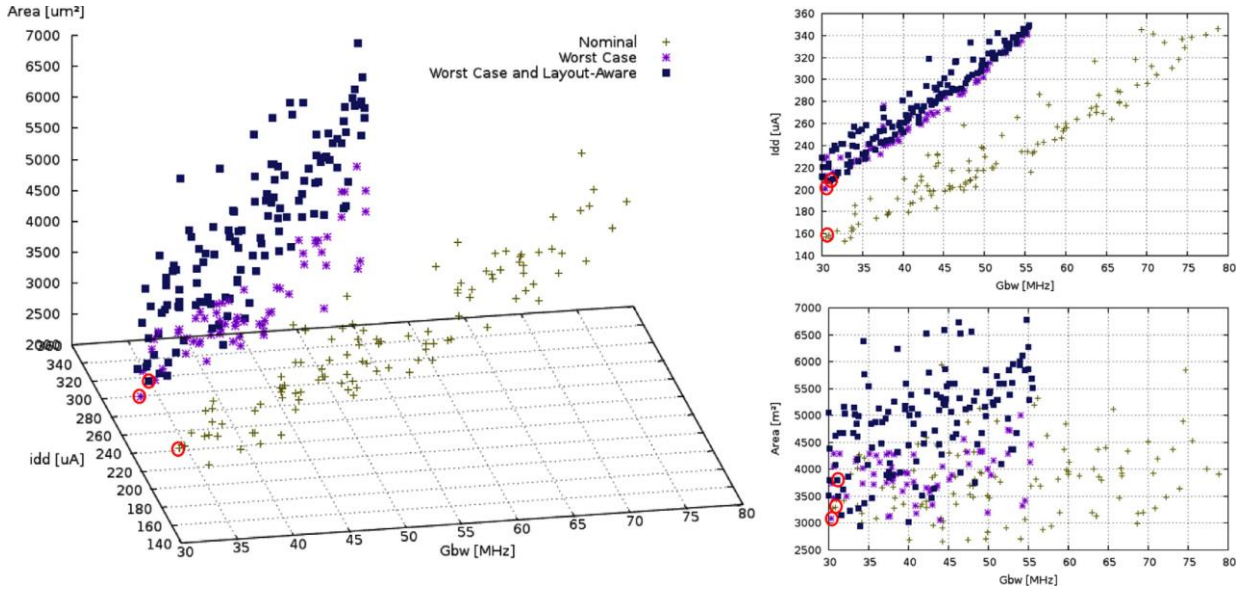


Table 6
Performance of the single stage amplifier for solution with lowest power of each optimization stage.

Specs.		Nominal			Worst case		Layout-aware
		Nominal	Performance after worst-case	Performance after WC and LA	Worst-case	Performance after WC and LA	WC and LA
Idd	min ≤ 350 µA	156.8	188.2	188.3	200.8	201.1	208.1
GBW	max ≥ 30 MHz	30.6	**25.3***	**24.7***	30.4	**29.7***	31.1
Area	min µm²	3274			3089		3807
GDC	≥ 50 dB	54.8	54.2	54.1	51.1	51.1	51.1
PM	≥ 60° dB	79.6	77.1	73.8	81.8	78.6	83.5
FOM	≥ 1000 MHz pF/mA	1168	1143	1107	1234	1199	1295

* Constraints violated, all layouts considered were generated using AIDA-L;

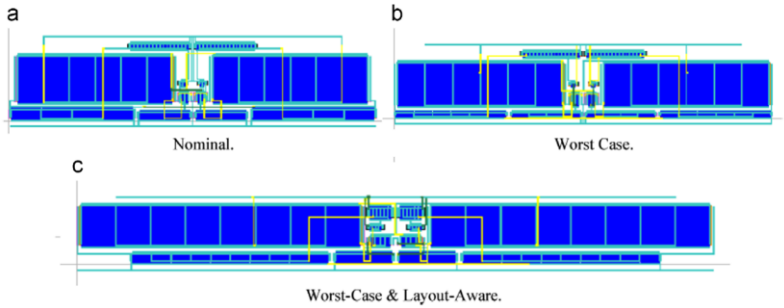


Fig. 19. Layout for the single stage amplifier solutions shown in Table 6.

Basic Ideas and Fundamentals

- **Applied EC Examples: (Computational Finance)**

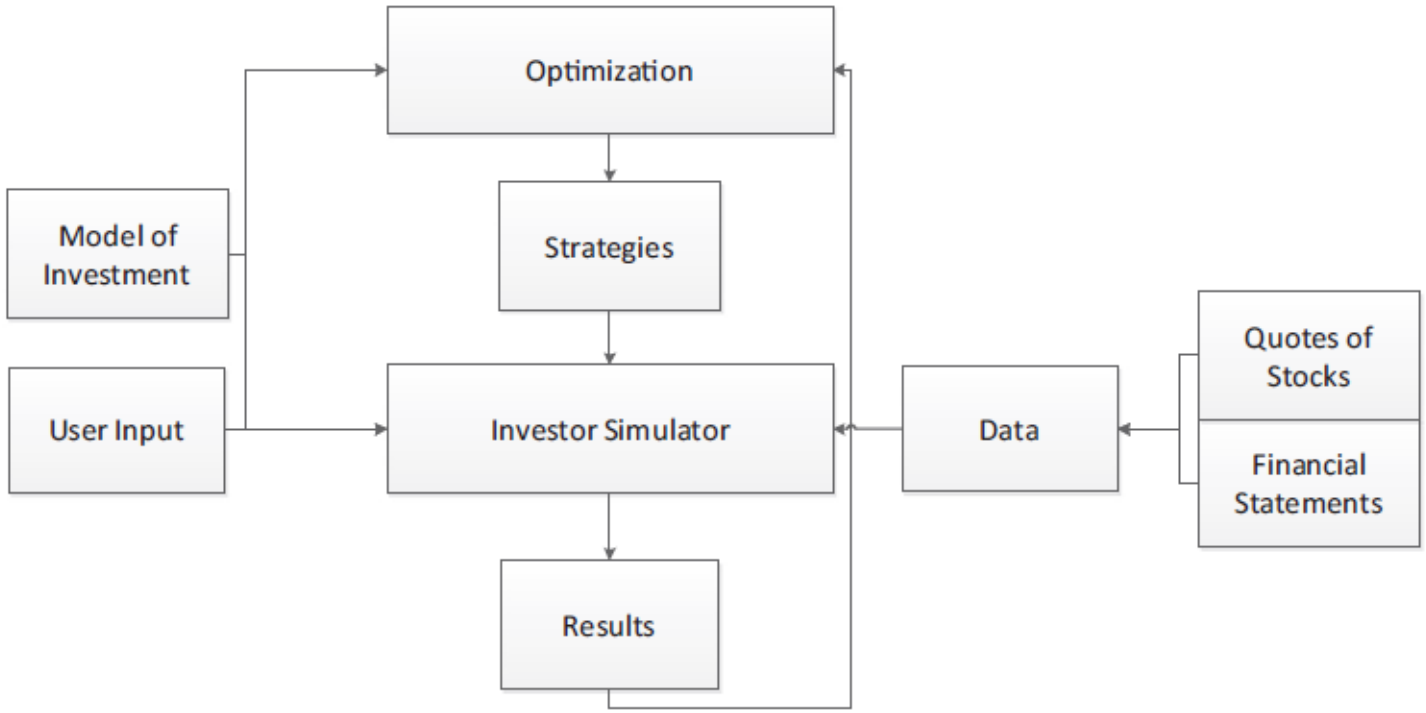


Fig. 1. System architecture.

- **EC** for Computational Finance
- Investiment Strategies, Fund Composition, Portfolio Management, etc.

Available MSc. Thesis proposals on the topic, supervised by:
R. Neves, N. Horta

Basic Ideas and Fundamentals

- **Applied EC Examples: (Computational Finance)**

Table 4
Results of the real test.

	Return (%)	Variance (%)
SP&500	25.55	10.82
Chromosome ID		
2	36.4	8.02
10	27.59	7.18
23	18.31	4.58
29	11.84	2.25
40	8.89	1.16

Table 5
Genes of the Chromosomes.

Genes	C2	C10	C23	C29	C0
Debt ratio	0,000006	0,000002	0,017	0,0003	0,000002
ROE	0,00015	0,00015	0,31	0,0029	0,31
Profit margin	0,3578	0,0059	0,02617	0,2156	0,00286
PER	1,11	1,11	2,099	0,8637	2.099
Δ of Rev.	0,0739	0,0739	0.8549	1.0786	0,01089
Δ Common stock out	1,3989	1.3983	1.9047	1.0967	1.5
Δ Net income	2.9453	1,8516	0.331	0,043	0,638
Stocks number	8	7	9	6	4
Position size	0,18	0,18	0,063855	0,0638	0,0638

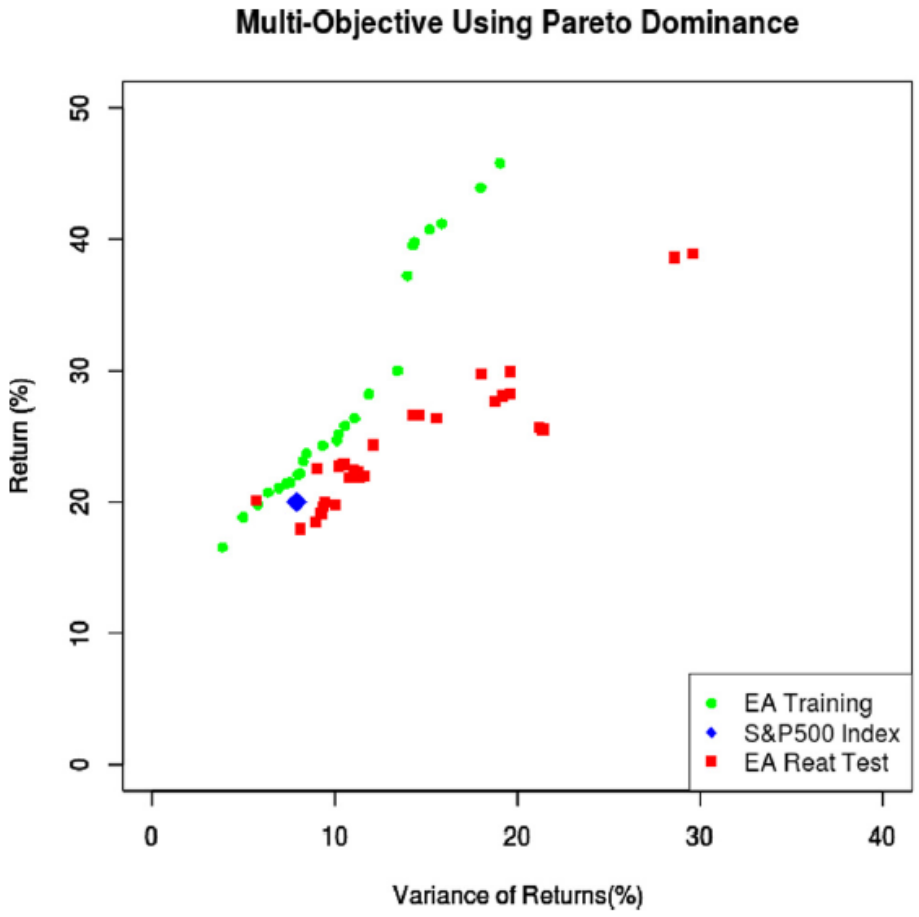
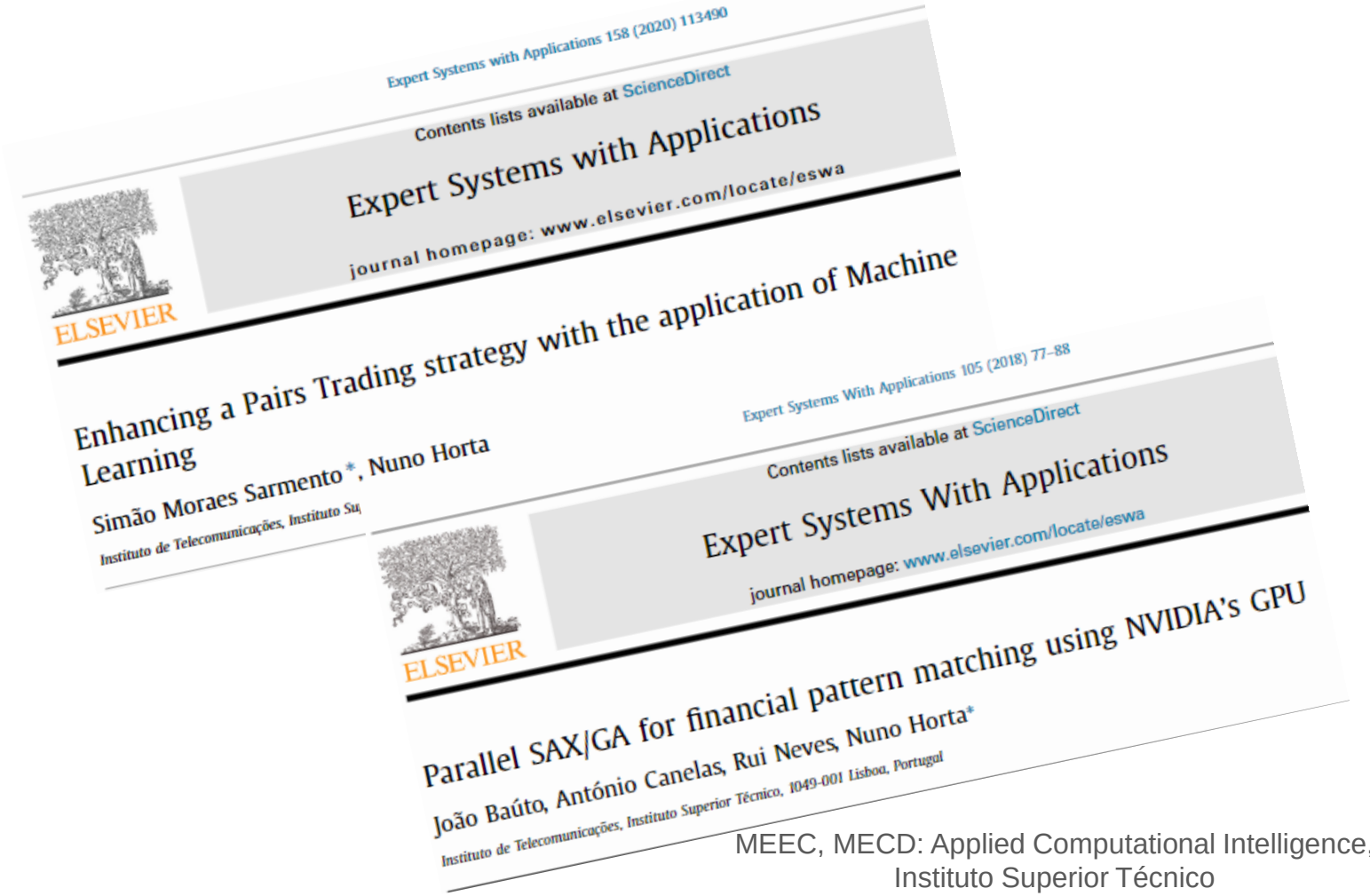


Fig. 21. Result of the real test using the third model.

Basic Ideas and Fundamentals

- **Applied EC Examples: (Computational Finance)**



- **EC** for Computational Finance
- Investiment Strategies, Fund Composition, Portfolio Management, etc.

Available MSc. Thesis proposals on the topic, supervised by:
R. Neves, N. Horta

Basic Ideas and Fundamentals

- **Applied EC Examples: (Bridge Design)**

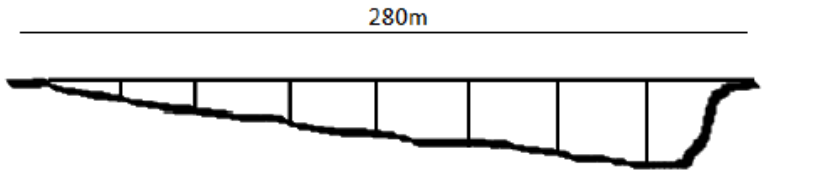


Figure 4.4 Longitudinal profile of the case study bridge.

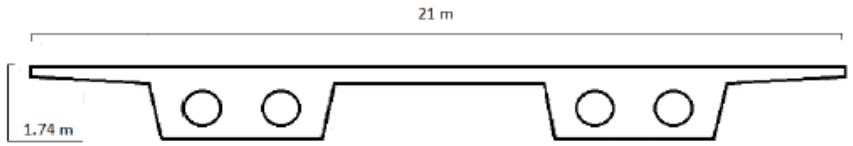
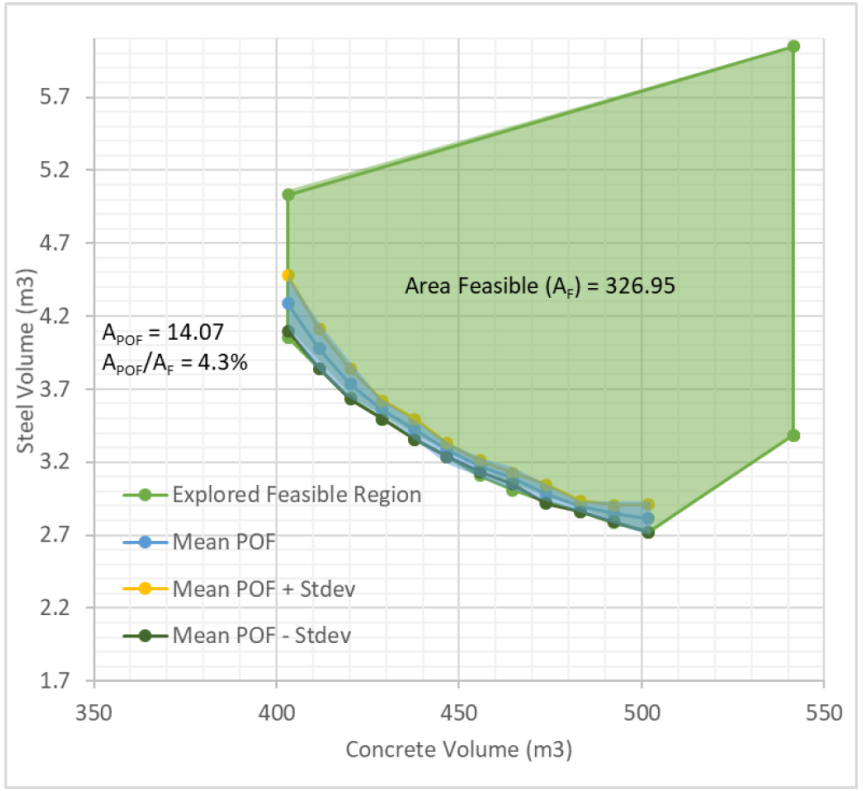


Figure 4.5 Cross-section of the case study's deck.



- **EC** applied to Civil Engineering
- Optimizing robustness and cost

Available MSc. Thesis proposals on the topic, co-supervised by:
N. Horta

Basic Ideas and Fundamentals

- **Applied EC Examples: (Optical Transport Networks - OTN)**

Applied Soft Computing Journal 83 (2019) 105608

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Multi-objective framework for cost-effective OTN switch placement using NSGA-II with embedded domain knowledge

Daniela Moniz [a,b,*], João Pedro [a,b], Nuno Horta [b], João Pires [b]

[a] Infinera Unipessoal Lda, Rua da Garagem 1, 2790-078 Oeiras, Portugal
[b] Instituto de Telecomunicações, Instituto Superior Técnico, Avenida Rovisco Pais 1, 1049-001, Lisboa, Portugal

HIGHLIGHTS

- A multi-objective framework for routing services in optical transport networks.
- With the goal of minimizing the key resources impacting capital expenditures.
- The framework relies on the NSGA-II algorithm with embedded domain knowledge.
- A study is conducted comparing the NSGA-II with the embedding knowledge algorithm.

- **EC** applied to ECE
- Optical Transport Networks (OTN)

Available MSc. Thesis proposals on the topic, co-supervised by:
N. Horta, R. Neves

Basic Ideas and Fundamentals

- **Applied EC Examples: (Wireless Sensor Networks)**

Applied Soft Computing 30 (2015) 104–112

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A multi-objective routing algorithm for Wireless Multimedia Sensor Networks

Naércio Magaia[a,*], Nuno Horta[b], Rui Neves[b], Paulo Rogério Pereira[a], Miguel Correia[a]

[a] INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, no 9, 1000-029 Lisboa, Portugal
[b] Instituto de Telecomunicações/Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, no 1, Torre Norte, Piso 10, 1049-001 Lisboa, Portugal

ARTICLE INFO

Article history:
Received 8 May 2014
Received in revised form 22 January 2015
Accepted 22 January 2015
Available online 2 February 2015

Keywords:
Multi-objective optimization
Wireless Multimedia Sensor Networks
Strength Pareto Evolutionary Algorithm
Quality of Service

ABSTRACT

In this paper, a new multi-objective approach for the routing problem in Wireless Multimedia Sensor Networks (WMSNs) is proposed. It takes into account Quality of Service (QoS) requirements such as delay and the Expected Transmission Count (ETX). Classical approximations optimize a single objective or QoS parameter, not taking into account the conflicting nature of these parameters which leads to sub-optimal solutions. The case studies applying the proposed approach show clear improvements on the QoS routing solutions. For example, in terms of delay, the approximate mean improvement ratios obtained for scenarios 1 and 2 were of 15 and 28 times, respectively.

- **EC** applied to ECE
- Wireless Sensor Networks

Available MSc. Thesis proposals on the topic, co-supervised by:
N. Horta, R. Neves

Bibliographic References

- **Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation: : J. Keller, D. Liu, and D. Fogel, 2016 Wiley**

- **Multiobjective Optimization: Interactive and Evolutionary Approaches: J. Branke, K. Deb, K. Miettinen, and R. Słowiński, 2008 Springer**

- **Lecture Slides, "Applied Computational Intelligence", N. Horta**

next chapter