X Seminario y Escuela Nacional de Aprendizaje e Inteligencia Computacional

Evolutionary Computation in Python: *Frameworks* de Código Abierto

M.C. José Alejandro Cornejo Acosta

M.C. Cosijopii García García

Dra. Alicia Morales Reyes

CCC-INAOE

Septiembre 2022

GOBIERNO DE MÉXICO | CONACYT Consejo Nacional de Ciencia y Tecnología | INAOE

Outline

1. Introduction to Evolutionary Computation
   - What is Evolutionary Computation
   - Positioning in Computer Sciences
   - Inspiration from nature
   - Target problems
   - On the good design
2. Open source and alternative for Genetic Algorithms
   - What is open source?
   - Software for C/C++ and Java
   - Software for Python
3. Hands-On Genetic Algorithms
   - Single-objective optimization problem
   - Knapsack problem
   - Traveling Salesman Problem and n-queens
   - The Frequency Assignment Problem (FAP)
   - The Multiobjective Optimization Problem
4. Final remarks

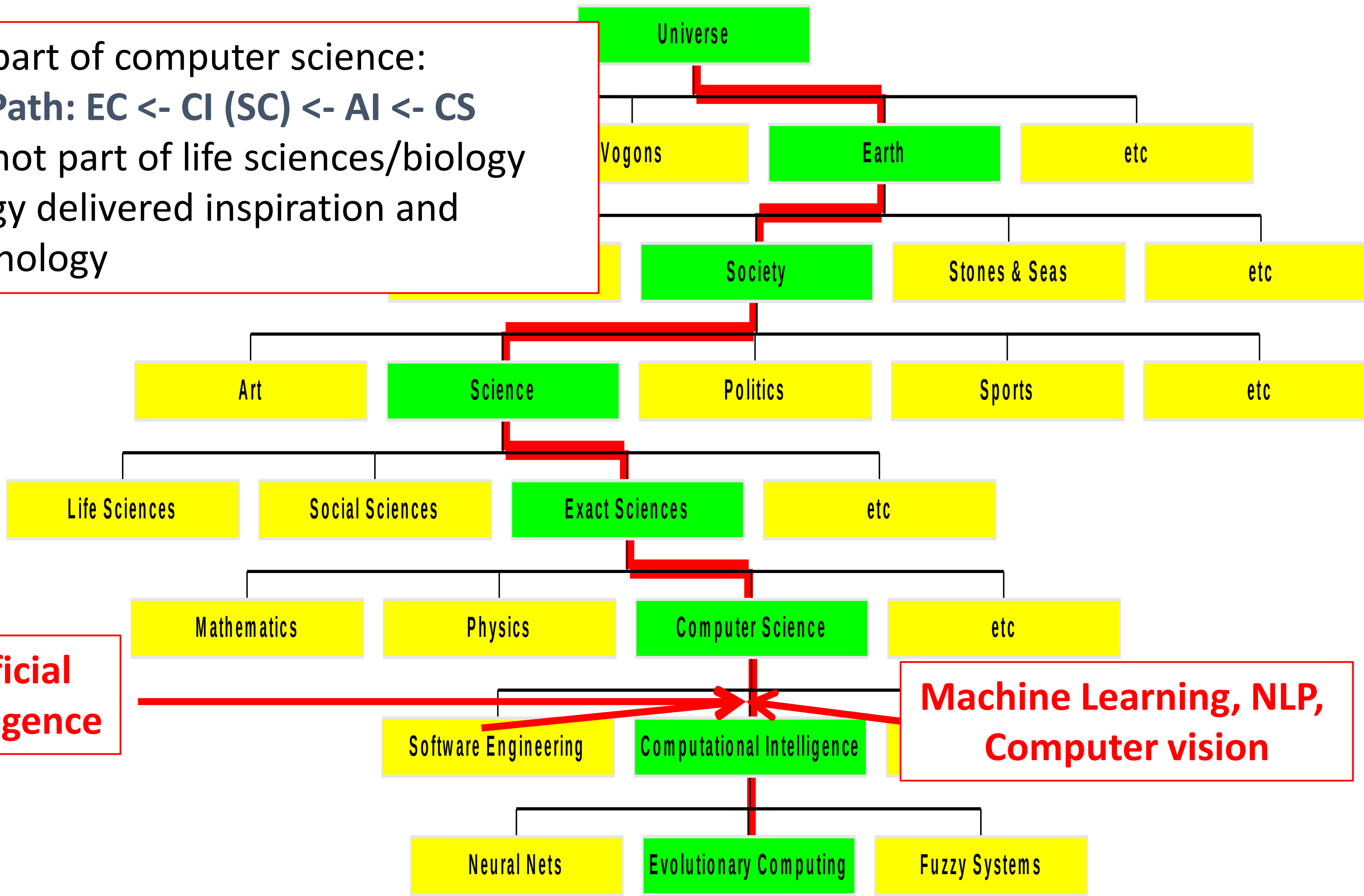
Positioning of Evolutionary Computation

EC is part of computer science:

Path: EC <- CI (SC) <- AI <- CS

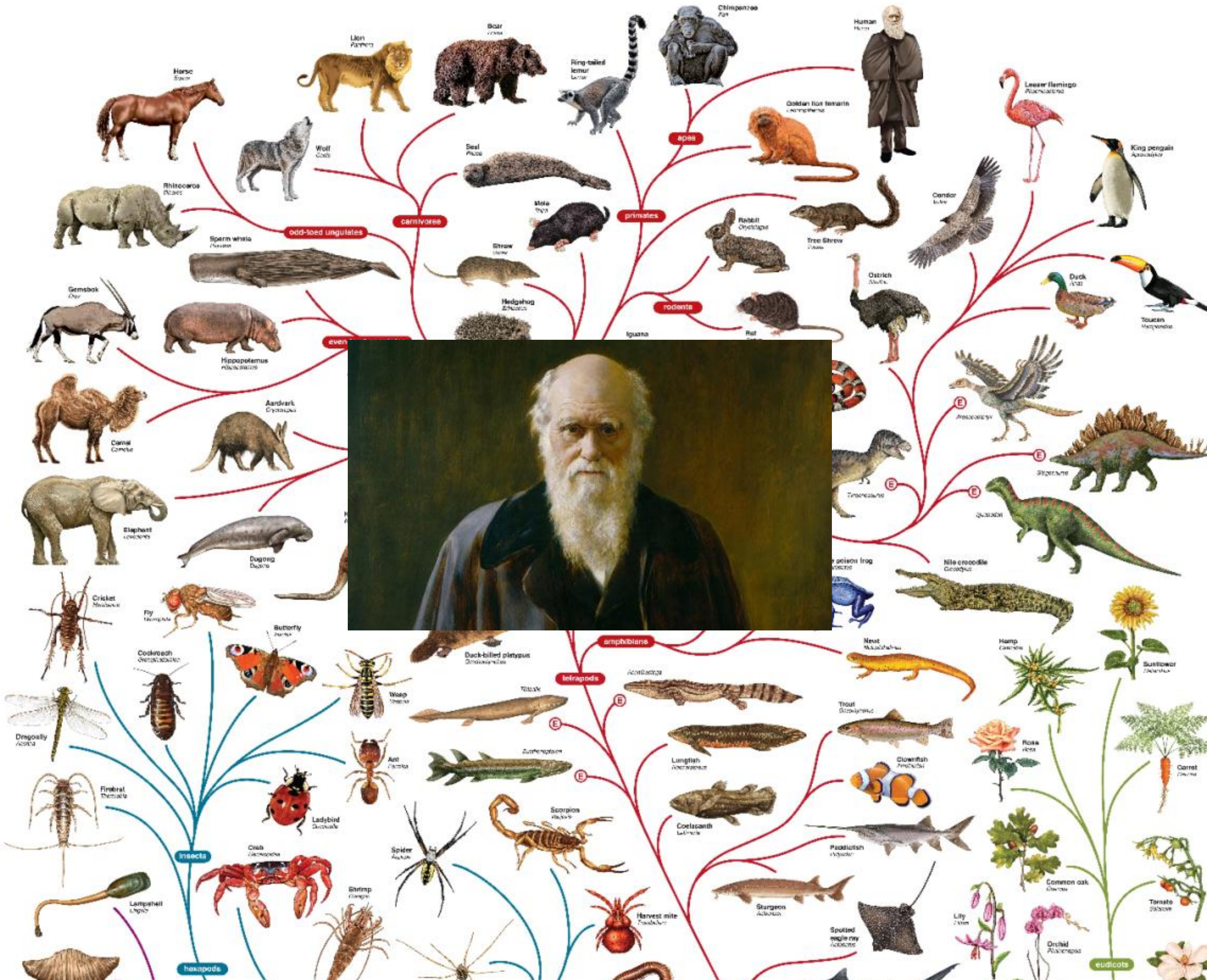
EC is not part of life sciences/biology

Biology delivered inspiration and terminology



Artificial Intelligence

Machine Learning, NLP, Computer vision



Horse
Lion
Bear
Ring-tailed lemur
Chimpanzee
Human
Lesser flamingo
King penguin
Golden lion tamarin
Wolf
Seal
Rhinoceros
Mole
apes
primates
carnivores
odd-toed ungulates
Sperm whale
Shrew
Rabbit
Tree Shrew
Condor
Ostrich
Duck
Toucan
Gemsbok
Hedgehog
rodents
Iguana
Hippopotamus
Aardvark
Camel
Elephant
Dugong
Nile crocodile
Cricket
Fly
Butterfly
Cockroach
Wasp
Duck-billed platypus
amphibians
tetrapods
Newt
Hemp
Sunflower
Dragonfly
Trout
Lungfish
Rose
Carrot
Ant
Clownfish
Firebrat
Scorpion
Coelacanth
Ladybird
Paddlefish
Common oak
Insecta
Crab
Spider
Shrimp
Harvest mite
Sturgeon
Tomato
Lampshell
Lily
Spotted eagle ray
Orchid
hexapods
eudicots

Evolutionary Computing

Optimization

- EC Is the collective name for a range of **problem-solving** techniques based on principles of biological evolution, such as **natural selection** and **genetic inheritance**. These techniques are being increasingly and widely applied to a variety of problems, ranging from practical applications in industry and commerce to leading-edge scientific research.

Evolutionary Computing

- Trial and error problem solving approach:
  - While not_satisfied_with_solution
    1. Generate candidate solution(s) for the problem at hand
    2. Evaluate the quality of candidate solution(s)
  - Return best_solution_found

EC techniques generate new solutions according to (rough) analogies with biological evolution principles

The Main Evolutionary Computing Metaphor

A given environment is filled with a population of individuals that strive for survival and reproduction

A problem is to be solved via stochastic trial-and-error, using as starting point a set of candidate solutions

EVOLUTION		PROBLEM SOLVING
Environment	⟷	Problem (search space)
Individual	⟷	Candidate Solution
Fitness	⟷	Quality of the solution

Fitness → chances for survival and reproduction

Quality → chance for seeding new solutions

Brief History 1: the ancestors

- 1948, Turing:
  proposes "genetical or evolutionary search"
- 1962, Bremermann
  optimization through evolution and recombination
- 1964, Rechenberg
  introduces evolution strategies
- 1965, L. Fogel, Owens and Walsh
  introduce evolutionary programming
- 1975, Holland
  introduces genetic algorithms
- 1992, Koza
  introduces genetic programming

Evolutionary computing

Brief History 2: The rise of EC

- 1985: first international conference (ICGA)

- 1990: first international conference in Europe (PPSN – Parallel Problem Solving from Nature)

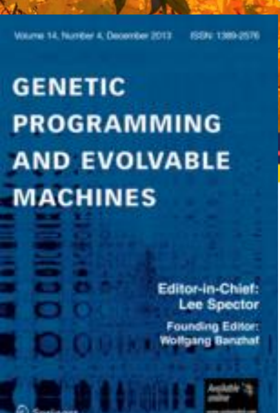
- 1993: first scientific EC journal (MIT Press)

- 1997: launch of European EC Research Network EvoNet

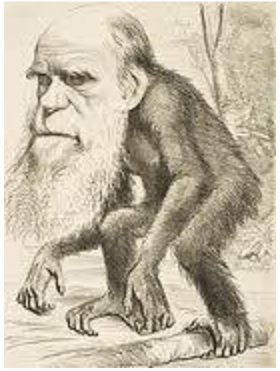
Brief History 3: The rise of EC

- 3 major EC conferences, about 10 small related ones (GECCO, CEC, PPSN)

- 3 scientific core EC journals (EC-MIT, TEC-IEEE, GPEM-Springer)



Darwinian Evolution 1: Survival of the fittest

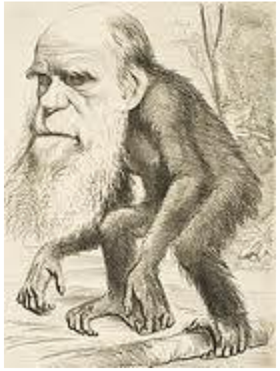


- All environments have finite resources

  (i.e., can only support a limited number of individuals)

- Lifeforms have basic instinct/ lifecycles geared towards reproduction
- Therefore, some kind of selection is inevitable
- Those individuals that compete for resources most effectively have increased chance of reproduction

Darwinian Evolution 2: Diversity drives change



- Phenotypic traits:
  - Behaviour / physical differences that affect response to environment
  - Partly determined by inheritance, partly by factors during development
  - Unique to each individual, partly as a result of random changes
- If phenotypic traits:
  - Lead to higher chances of reproduction
  - Can be inherited

  then they will tend to increase in subsequent generations,
- leading to new combinations of traits …

Darwinian Evolution: Summary

- Population consists of diverse set of individuals
- Combinations of traits that are better adapted tend to increase representation in population

Individuals are “units of selection”

- Variations occur through random changes yielding constant source of diversity, coupled with selection means that:

Population is the “unit of evolution”

- Note the absence of “guiding force”

Adaptive landscape metaphor (Wright, 1932)

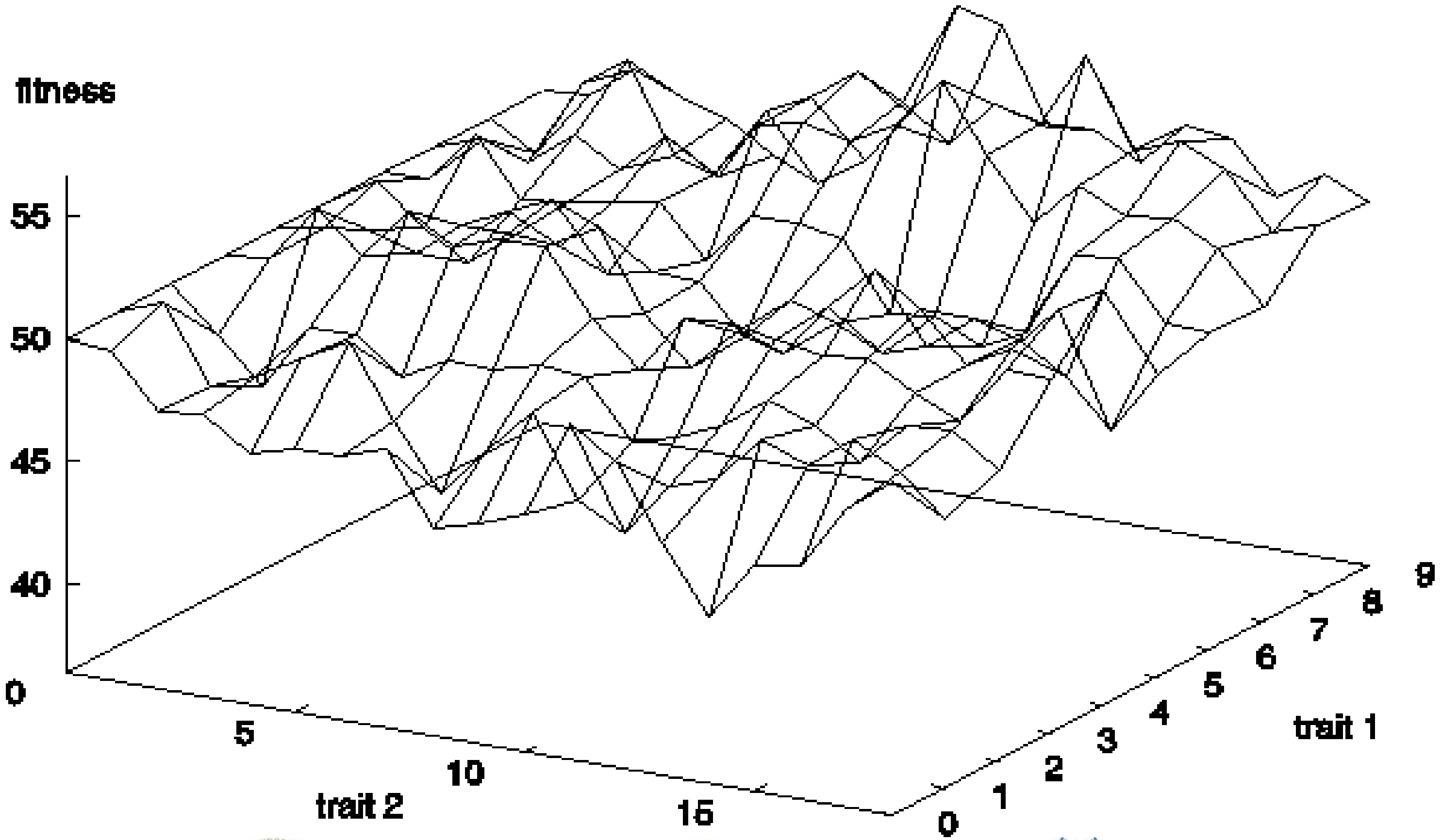
- Can envisage population with *n* traits as existing in a *n+1*-dimensional space (landscape) with height corresponding to fitness
- Each different individual (phenotype) represents a single point on the landscape
- Population is therefore a "cloud" of points, moving on the landscape over time as it evolves - adaptation

Adaptive landscape metaphor (cont'd)

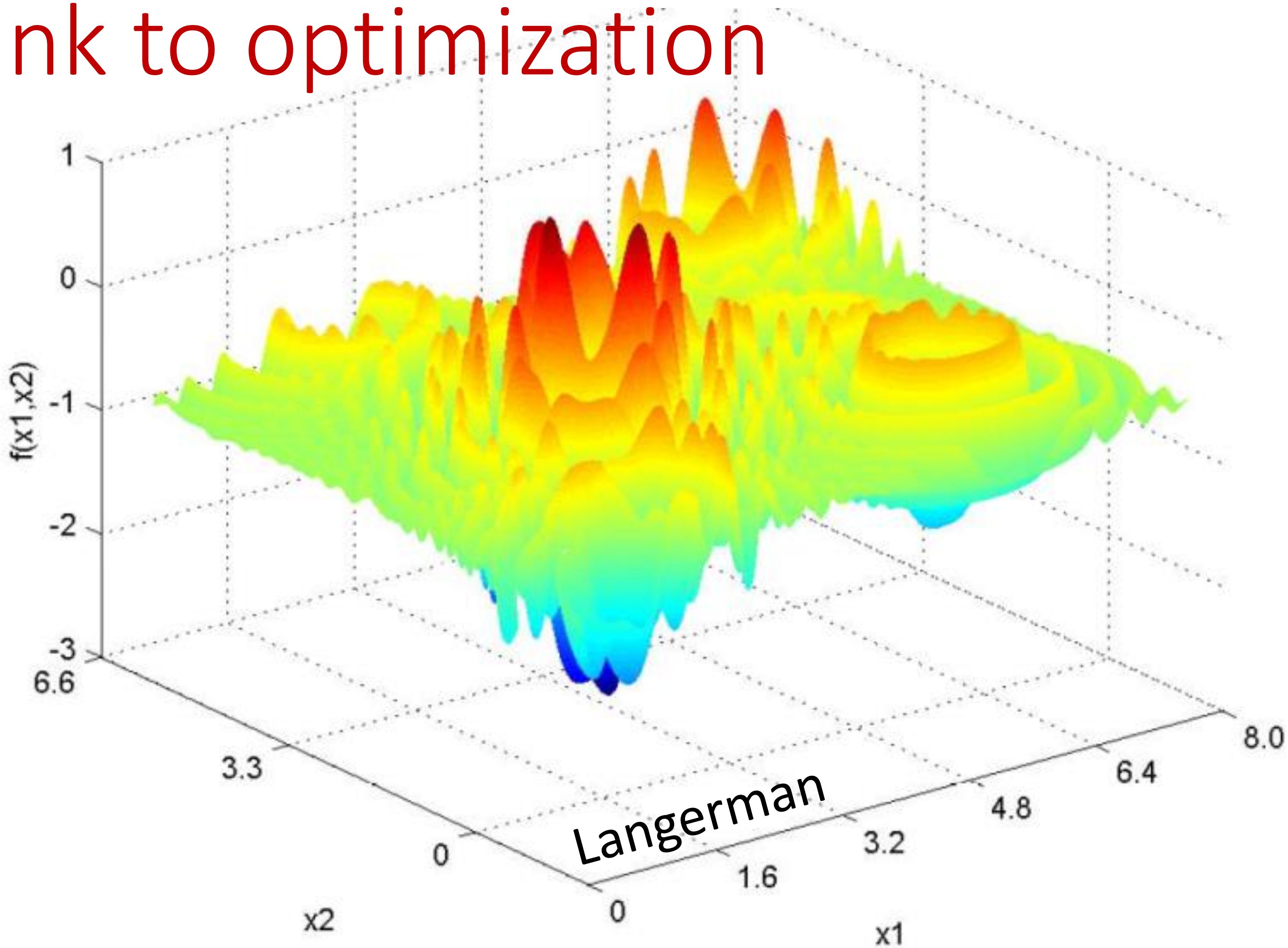
- Selection “pushes” population up the landscape

- Genetic drift:
  - random variations in feature distribution
    (+ or -) arising from sampling error
  - can cause the population “melt down” hills, thus crossing valleys and leaving local optima

Example with two traits



Link to optimization



Motivations for EC: 1

- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solver known in nature is:
  - the (human) brain that created “the wheel, New York, wars and so on” (after Douglas Adams’ Hitch-Hikers Guide)
  - the evolution mechanism that created the human brain (after Darwin’s Origin of Species)

Motivations for EC: 2

- Developing, analyzing, applying problem solving methods a.k.a. algorithms is a central theme in mathematics and computer science
- Time for thorough problem analysis decreases
- Complexity of problems to be solved increases
- Consequence:
  Robust problem solving technology needed

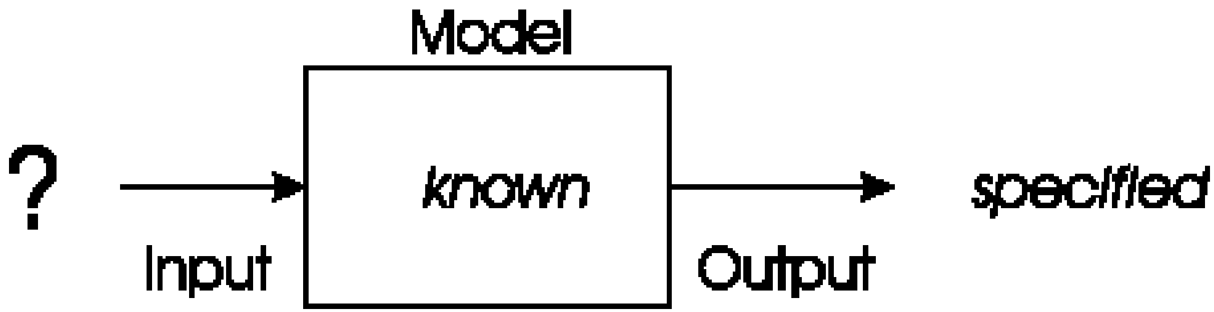
Adequate problems for EC

- (Hard) combinatorial optimization
- Multi-objective optimization
- Constrained optimization
- Tough (differentiable) objective functions
- Multimodal objective functions
- …

Domains/applications requiring fast response
NP- problems
Requiring (almost) accurate solutions

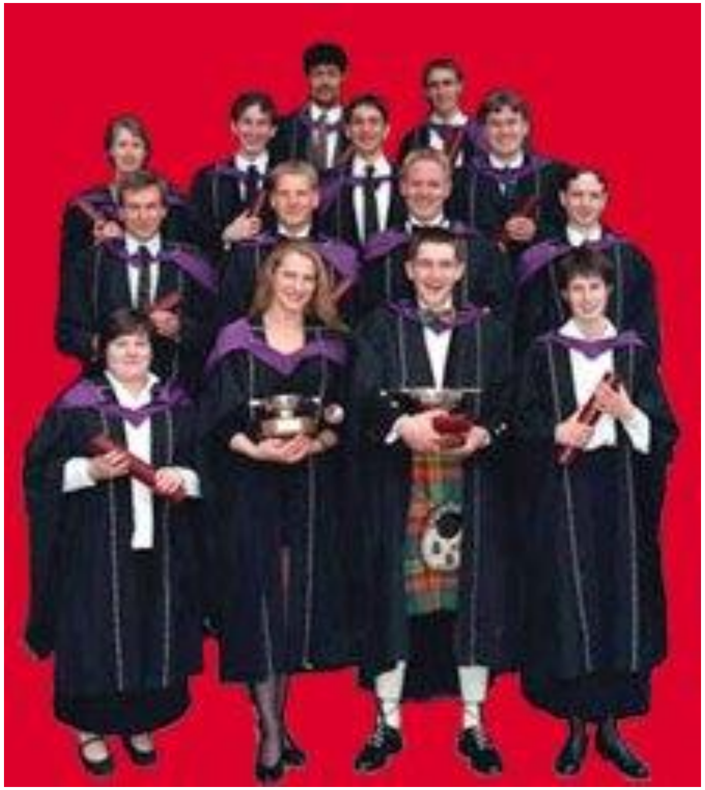
Problem type 1 : Optimization

- We have a model of our system and seek inputs that give us a specified goal



- e.g.
  - time tables for university, call center, or hospital
  - design specifications, etc etc

Optimisation example 1: University timetabling



Enormously big search space

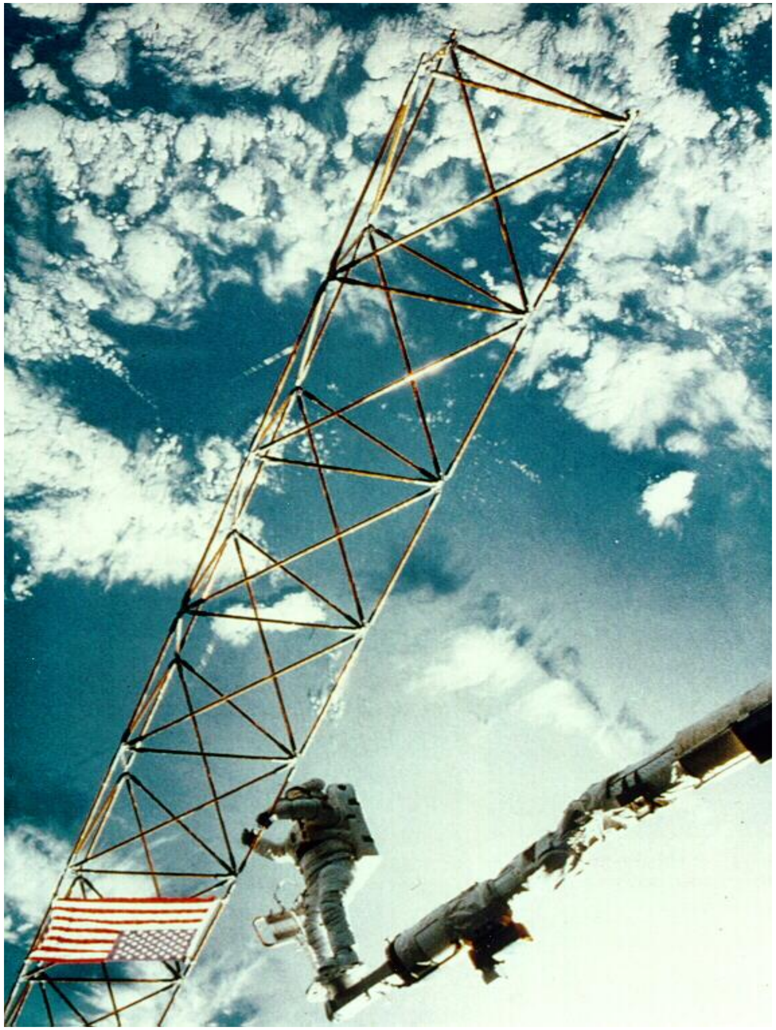
Timetables must be *good*

“Good” is defined by a number of competing criteria

Timetables must be feasible

Vast majority of search space is infeasible

Optimization example 2: Satellite structure



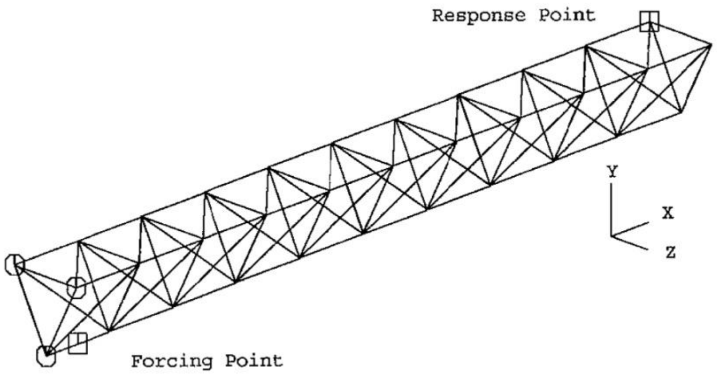
Optimised satellite designs for NASA to maximize vibration isolation

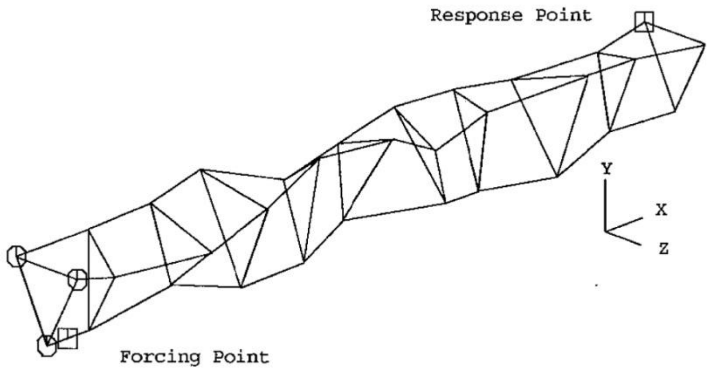
Evolving: design structures

Fitness: vibration resistance

Evolutionary “creativity”

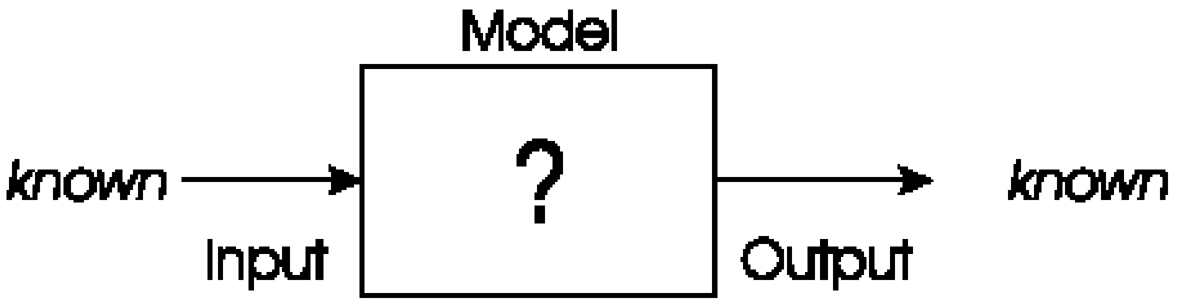
Optimisation example 2: Satellite structure





Problem types 2: Modelling

- We have corresponding sets of inputs & outputs and seek model that delivers correct output for every known input



- Evolutionary machine learning

Modelling example: loan applicant creditibility

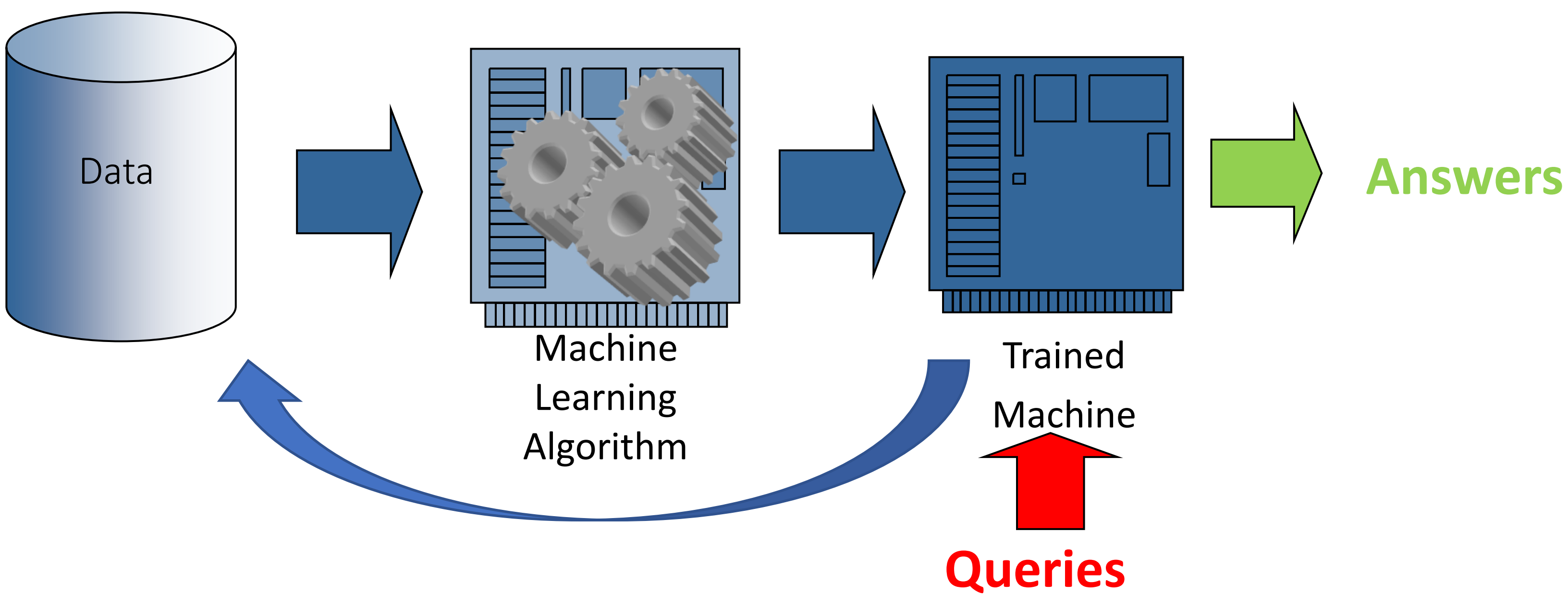
British bank evolved creditability model to predict loan paying behavior of new applicants

Evolving: prediction models

Fitness: model accuracy on historical data

Machine Learning

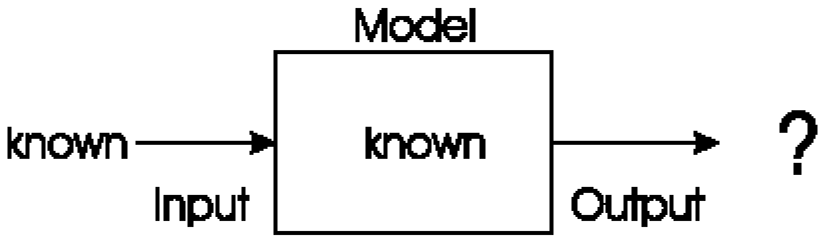
Learning = representation + evaluation + optimization



Isabelle Guyon. **A Practical Guide to Model Selection.** In Jeremie Marie, editor, Machine Learning Summer School 2008, Springer Texts in Statistics, 2011. (slide from I.Guyon's)

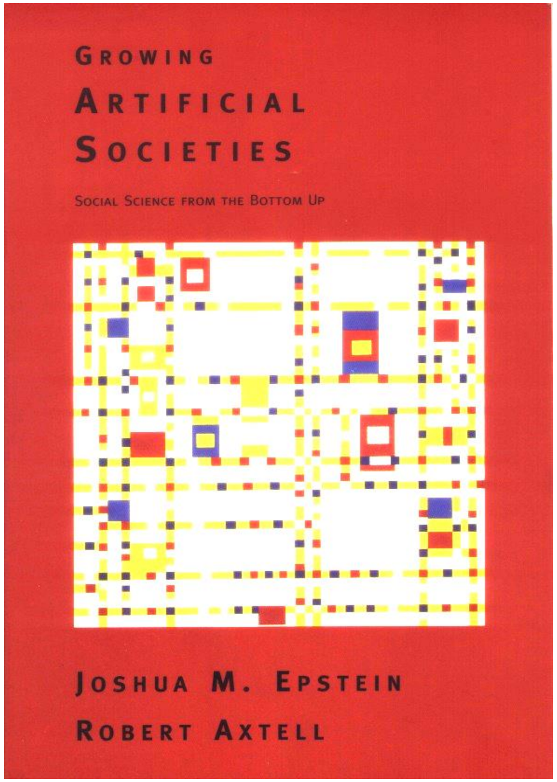
Problem type 3: Simulation

- We have a given model and wish to know the outputs that arise under different input conditions



- Often used to answer "what-if" questions in evolving dynamic environments
- e.g. Evolutionary economics, Artificial Life

Simulation example: evolving artificial societies



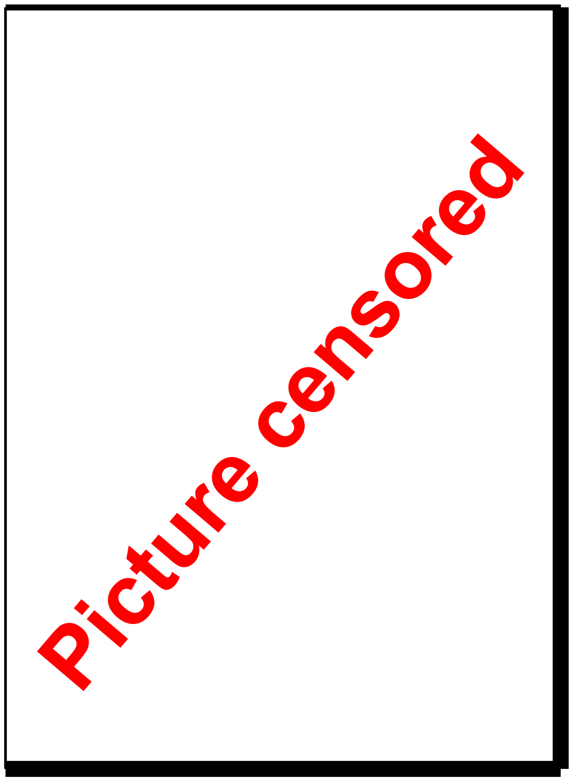
Simulating trade, economic competition, etc. to calibrate models

Use models to optimize strategies and policies

Evolutionary economy

Survival of the fittest is universal (big/small fish)

Simulation example 2: biological interpetations



Incest prevention keeps evolution from rapid degeneration

(we knew this)

Multi-parent reproduction, makes evolution more efficient

(this does not exist on Earth in carbon)

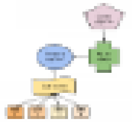
Novel “metaheuristics”(improved)

Frankenstein methods

How not to develop a metaheuristic

1. Take your favorite method
2. Add a component
3. While not working to your satisfaction, go to 2

K. Sorensen. **Metaheuristics – The Methapor Exposed.** 2012 (Draft, tutorial - Euro2012)



The reincarnation algorithm

The Reincarnation Algorithm

submitted to Journal of Heuristics in 2011

"RA utilizes the techniques of
local and population based

Population of commoners gets influenced by the
population of their own community guru(s) and

The system basically works with two sets of human population. One is salient population (called gurus) with higher fitness values and another is common population (called commoners) with low fitness values. The entire population of humans is dispersed evenly into many small subsets of communities. Humans are bound to perform karma or deeds that can upgrade or degrade their souls depending on the types of karma (either good or bad).

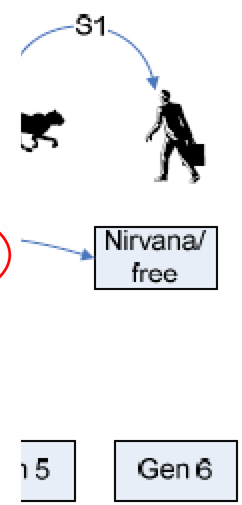


Fig. 1 reincarnation concept of rebirth cycle

A. Sharma, **A new optimizing algorithm using reincarnation concept.** 11th International Symposium on Computational Intelligence and Informatics (CINTI), pp. 281--288, IEEE, 2010

For a Good Metaheuristic Design

Focus on the problem

- Do not develop a method without a problem
- No black-box implementations of existing frameworks
- Study the problem in detail
- Know the literature on the problem and on related problems
- Study the relationship between methods from the literature and the problem
- Use the best parts from existing methods for your problems (e.g., matheuristics)

It's all about optimization!

K. Sorensen. **Metaheuristics – The Methapor Exposed.** 2012 (Draft, tutorial - Euro2012)

For a Good Metaheuristic Design

No new metaphors, please

There are more than enough "novel methods"

- Use the standard optimization vocabulary
- Stay within the existing taxonomy
- No points are awarded for originality, only for performance

A metaphor can serve as inspiration
Never as justification!

K. Sorensen. **Metaheuristics – The Methapor Exposed.** 2012 (Draft, tutorial - Euro2012)

Thank you!