Optimisation applications in Engineering

Dr Fabio Caraffini



IAI: Institute of Artificial Intelligence

✉ **fabio.caraffini@dmu.ac.uk**
www.cse.dmu.ac.uk/~fcaraf00



23/11/2018 **Universidad EIA, Medellín, Colombia**

Optimisation Problems: a generalised formulation

Any time we pick a decision/make a choice we face an optimisation problem

$$
\begin{array}{lcl}
\textit{Maximise/Minimise} & f_m(\mathbf{x}) & m = 1, 2, \ldots, M \\
\textit{subject} - \textit{to} & g_j(\mathbf{x}) \leqslant 0 & j = 1, 2, \ldots, J \\
& h_k(\mathbf{x}) = 0 & k = 1, 2, \ldots, K \\
& x_i^L \leqslant x_i \leqslant x_i^U & i = 1, 2, \ldots, n
\end{array}
$$

Regardless of their nature, all optimisation problems share the same mathematical formulation, see [3] for details.

Optimisation Problems: a generalised formulation

Any time we pick a decision/make a choice we face an optimisation problem

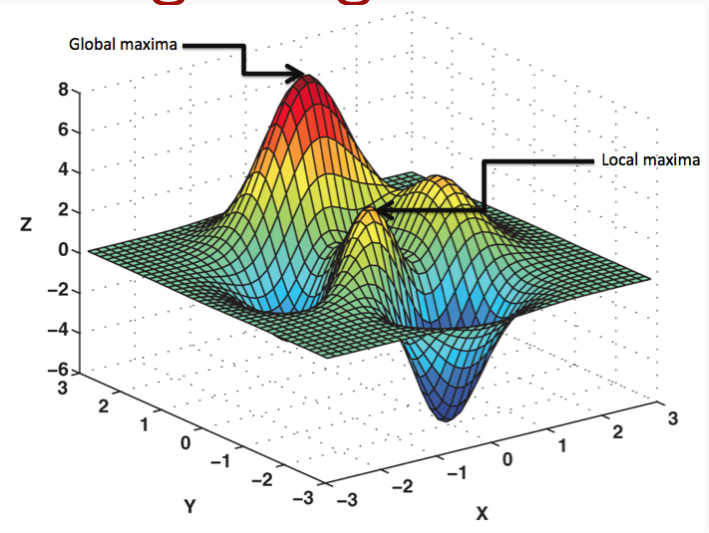
$$\begin{array}{lcl} \textit{Maximise}/\textit{Minimise} & f_m(\mathbf{x}) & m = 1, 2, \ldots, M \\ \textit{subject} - \textit{to} & g_j(\mathbf{x}) \leqslant 0 & j = 1, 2, \ldots, J \\ & h_k(\mathbf{x}) = 0 & k = 1, 2, \ldots, K \\ & x_i^L \leqslant x_i \leqslant x_i^U & i = 1, 2, \ldots, n \end{array}$$

We will focus on real-valued, single-objective and box constrained optimisation:

$$\mathbf{x}^* \equiv \arg\{\min_{\mathbf{x} \in \mathbf{D}} f(\mathbf{x})\} = \{\mathbf{x}^* \in \mathbf{D} \,|\, f(\mathbf{x}^*) \leqslant f(\mathbf{x}) \; \forall \, \mathbf{x} \in \mathbf{D}\}$$

The Mathematical Approach: exploiting the gradient info

There is a number of exact methods that:



return global maxima & minima 👍

if numerical, always converge 👍

may require $f(\mathbf{x}) = \ldots$ 👎

need assumptions (smoothness) 👎

are computationally expensive 👎

straggle on multimodal problems 👎

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

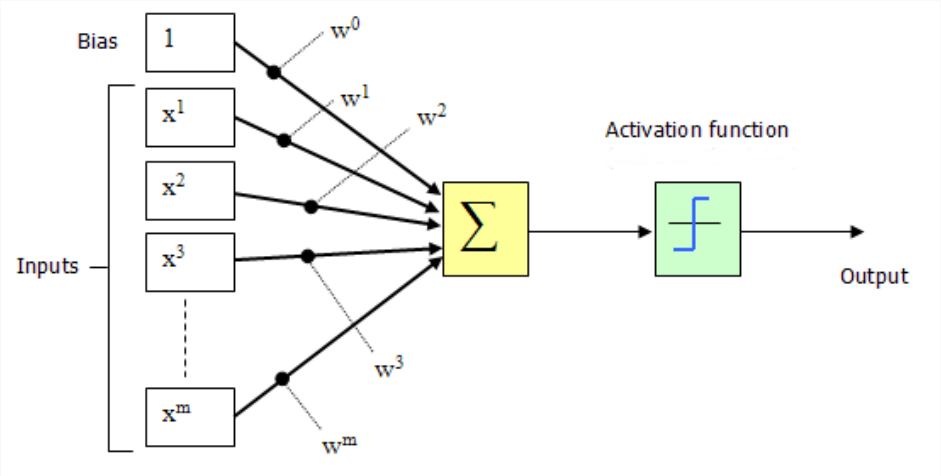
$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

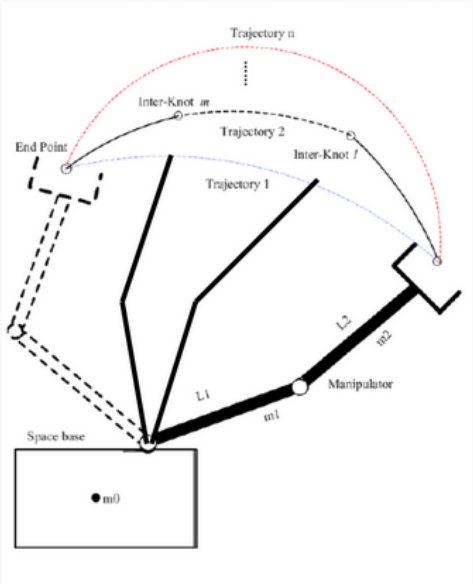
Features of real-world optimisation problems

- When possible the exact approach is preferable but:
  - in Engineering an analytical expression $f(x)$ is rarely available;
  - and it is rarely differentiable.

- Moreover, some features of the problem can be very challenging...

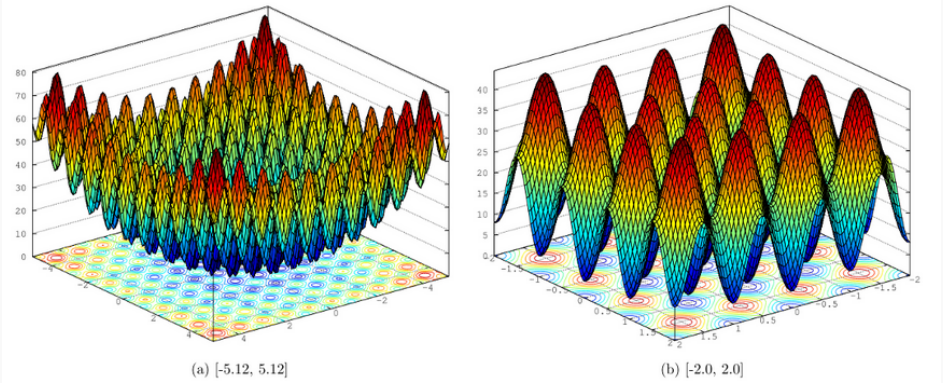
Hard real-life:
HIGH NON-LINEARITY.





- Usually optimisation problems are characterised by nonlinear function. **Optima are not on the bounds**!
- In real world optimisation problems, the physical phenomenon, due to its nature (e.g. in the case of saturation phenomenon or for systems which employ electronic components), cannot be approximated by a linear function neither in some areas of the decision space.

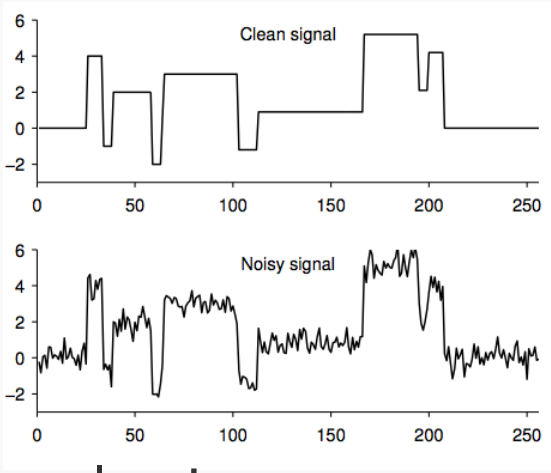
Hard real-life:
HIGH MULTI-MODALITY.



- It often happens that the fitness landscape contains many local optima and that many of these have an unsatisfactory performance (fitness value).
- These fitness landscapes are usually rather difficult to be handled since **the optimisation algorithms which employ gradient** based information in detecting the search direction **could easily converge to a suboptimal basin of attraction**.[1]

[1] Basin of attraction: set of points of the decision space such that ,initial conditions chosen, dynamically evolve to a particular attractor.

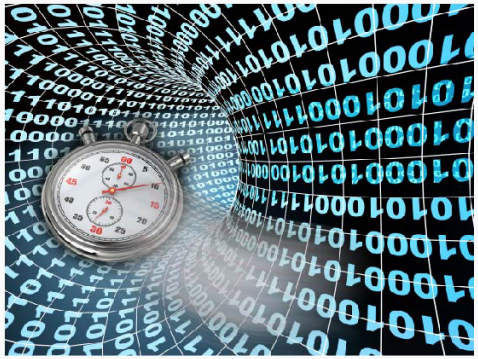
Hard real-life:
OPTIMISATION IN UNCERTAIN ENVIRONMENTS.



- **Noisy fitness function**: noise in fitness evaluations may come from many different sources such as sensory measurement errors or randomized simulations.
- **Approximated fitness function**: when the fitness function is very expensive to evaluate, or an analytical fitness function is not available, approximated fitness functions are often used instead. These approximated models implicitly introduce a noise which is the difference between the approximated value and real fitness value, which is unknown.
- **Robustness**: often, when a solution is implemented, the design variables or the environmental parameters are subject to perturbations or changes (e.g. control problems).

Hard real-life:
COMPUTATIONAL EXPENSIVE PROBLEMS.

Optimisation problems can be computationally expensive because of two reasons:



- large scale problems (needle in a haystack).
- computationally expensive fitness function (e.g. design of aircraft, control on an on-line electric drive).

Hard real-life:
MEMORY/TIME CONSTRAINTS

Many engineering problems are plagued by a modest hardware and stringent time constraints.

This can happen:



- due to cost limitations (e.g. vacuum cleaner robot);
- due to space limitations (e.g. use of minimalistic embedded systems as wearable devices, wireless sensors networks, etc.);
- in real-time systems (e.g. Telecommunication, Video-games, etc.).

Light (and simple) algorithms can be used in a modular way to tackle complex problems: if the hardware is limited an intelligent solution must be found!

Metaheuristic optimisation

*To overcome these issues, a logical choice of tools are **Evolutionary Algorithms**:*

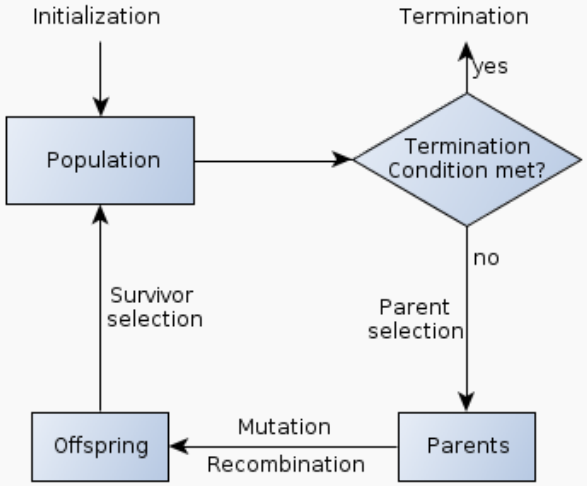
- **Fogel Owens (USA, 1964)**: *Evolutionary Programming (EP)* see [9, 8];
- **Holland (USA, 1975)**: *Genetic Algorithm (GA)* see [10];
- **Rechenberg Schwefel (Germany, 1971)**: *Evolution Strategies (ES)* see [20];
- **Koza (USA, 1990)**: *Genetic Programming (GP)* see [16, 18, 17];
- **Moscato (Argentina-USA 1989)**: *Memetic Algorithms (MA)* see [19].

They all exploit the concept of Darwinian evolution to generate new solutions in the search space

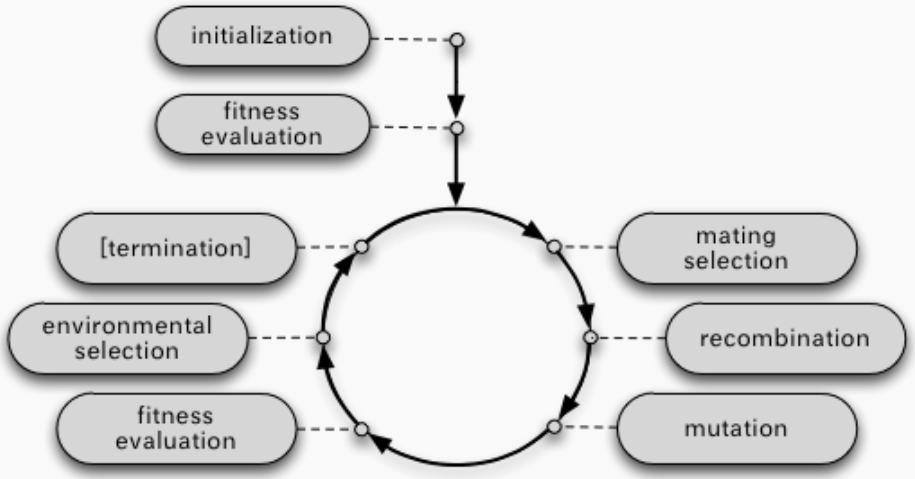
Evolutionary algorithms: general scheme.

Evolutionary Algorithms pseudo-code

INTITIALISE population with random candidate solutions
EVALUATE each solution
while *condition on budget* **do** ▷ or fitness value!
  SELECT parents ▷ 2 or multiple solutions
  RECOMBINE parents ▷ 2 or some of them
  MUTATE the resulting offspring
  EVALUATE new candidates
  SELECT individual for next generation
end while
Output best individual of the final population



Solutions cycle



Selection and variation operators cycle

Simpler yet efficient approaches for embedded systems

- The aforementioned methods do the job, but can...
  - very complex;
  - easily require at least $5000 \times D^2$ iteration to converge.
- One can think of modifyingf Evolutionary Algorithms...
  - to get rid of the population and free memory;
  - reduce the algorithmic overhead;
  - embed them in micro-controllers and perform optimisation on-the-flight.

[2] D is the dimensionality of the problem.

Simpler yet efficient approaches for embedded systems

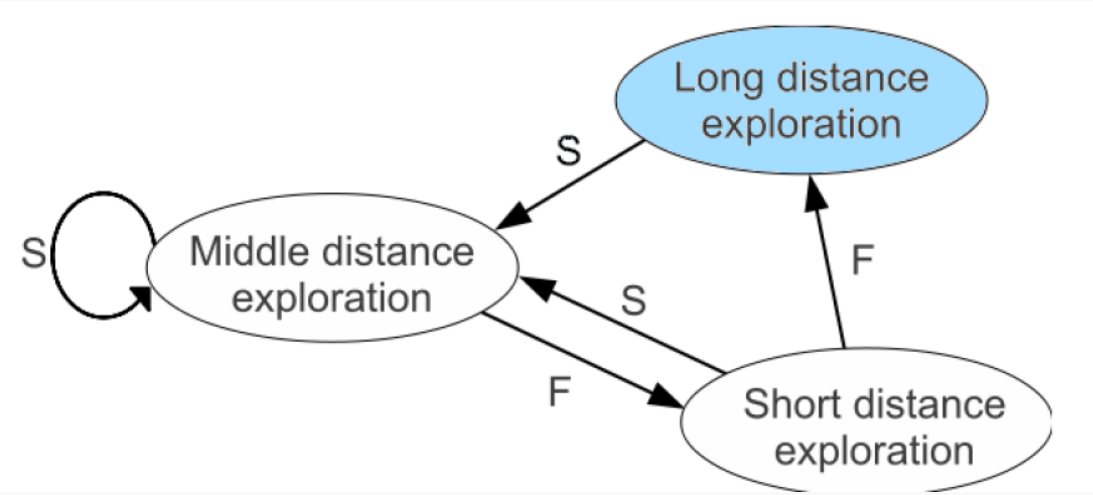
- The aforementioned methods do the job, but can...
  - very complex;
  - easily require at least $5000 \times D$ iteration to converge.
- One can think of modify Evolutionary Algorithms...
  - to get rid of the population and free memory;
  - reduce the algorithmic overhead;
  - embed them in micro-controllers and perform optimisation on-the-flight.

Engineering fields, such as System Control, Robotics and TLC, demand the solution of complex optimisation problems in a reasonable amount of time and by employing the least memory as possible, see e.g. [3];

Three stage optimal memetic exploration [15]

Stage 1: long distance exploration

- A single solution is perturbed;
- initially, via a simple random search (slightly biased via exp crossover with $\mathbf{x_{best}}$).

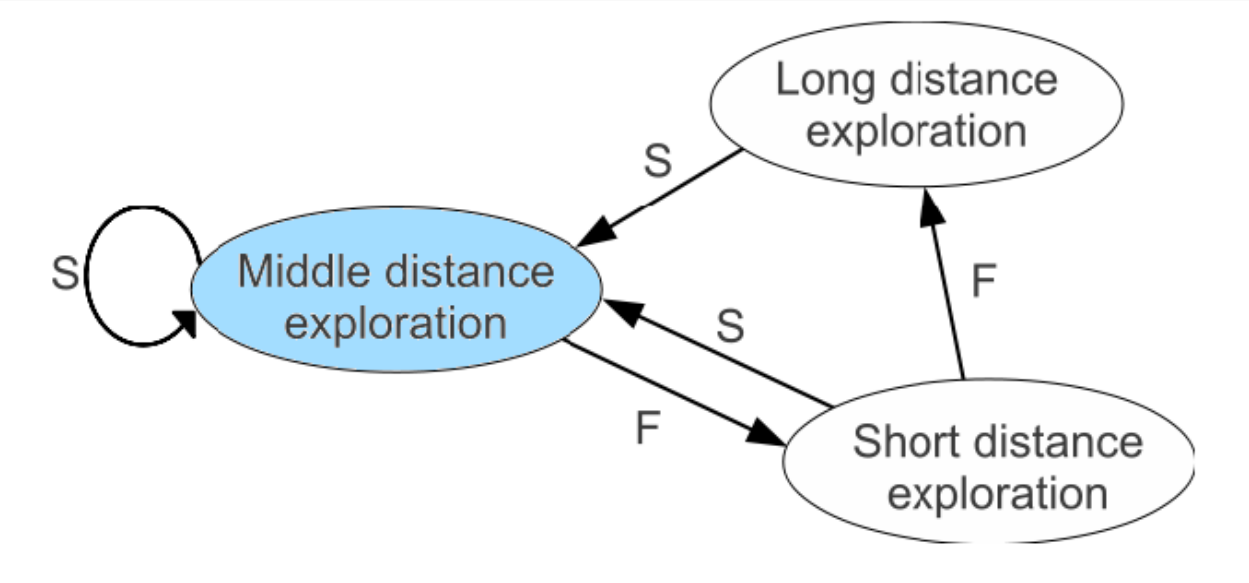


F: failed at improving upon initial solution.
S: succeeded at improving upon initial solution.

Three stage optimal memetic exploration

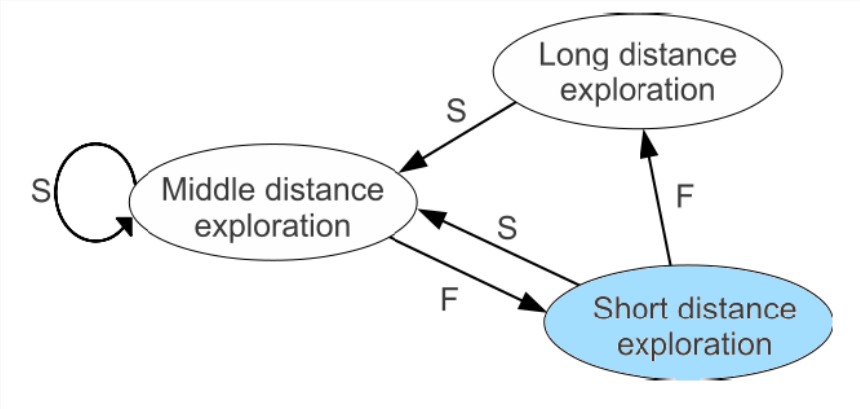
Stage 2: middle distance exploration

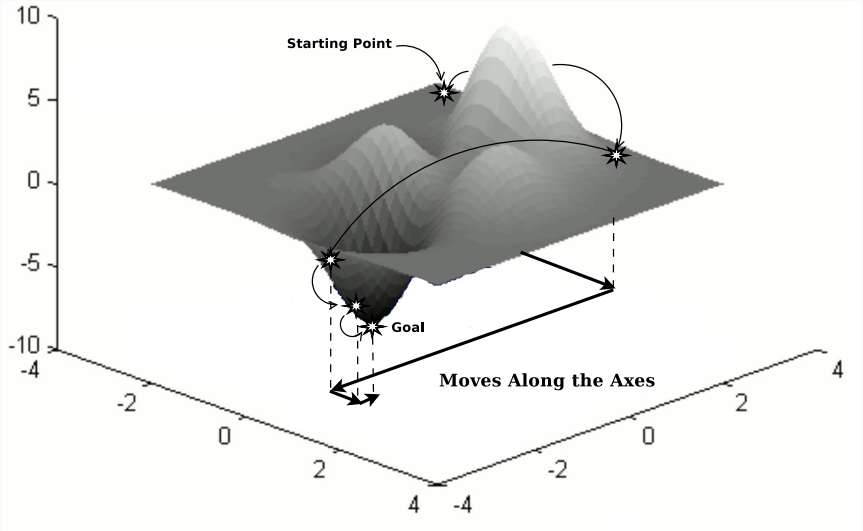
- A hypercube is centred to the current solution;
- a random search takes place inside the new space (strongly biased via exp crossover with $\mathbf{x_{best}}$).

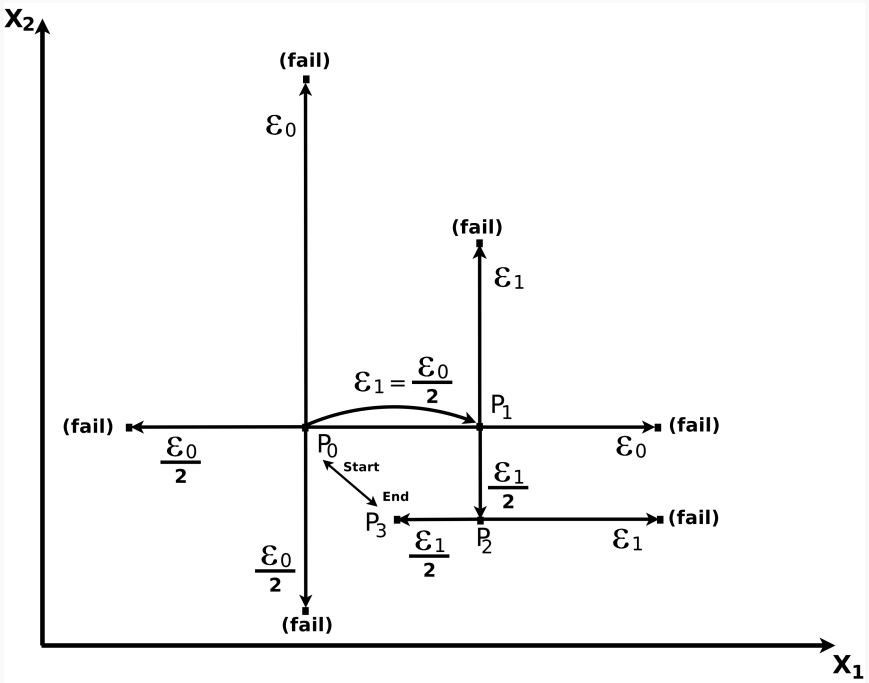


Three stage optimal memetic exploration

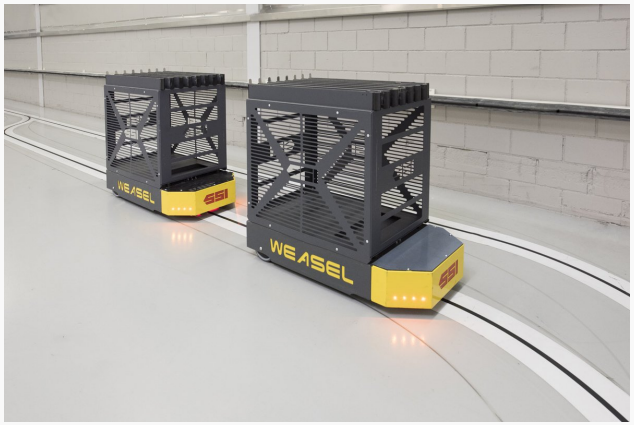
Stage 3: short distance exploration (as in [1, 2, 7, 5, 6, 4])



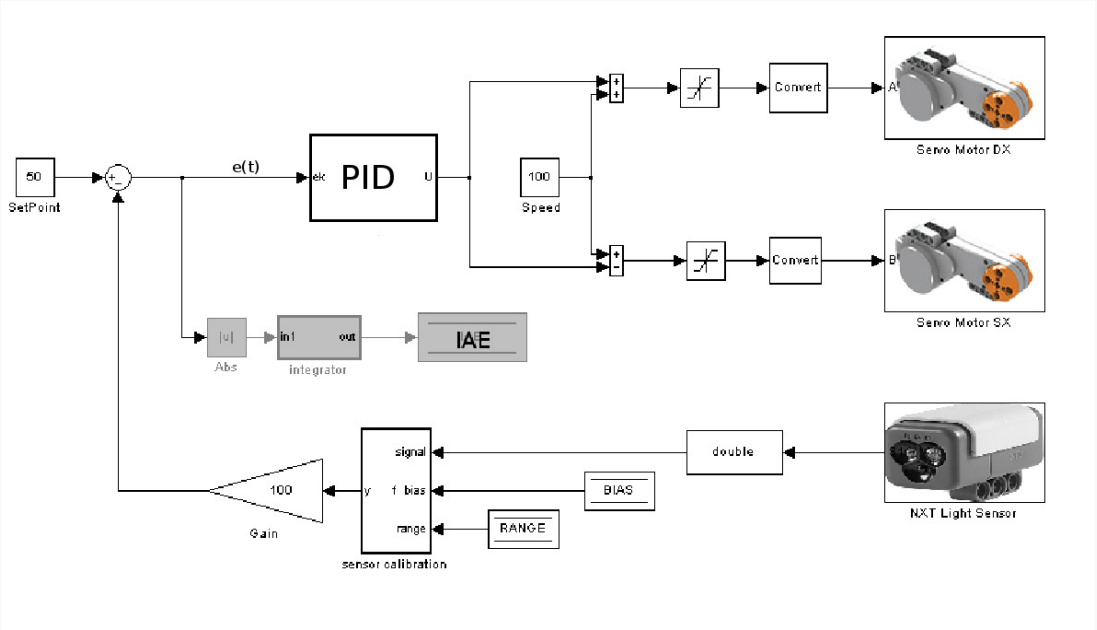




Case study I
Online PID tuning for automated guided vehicle [14]



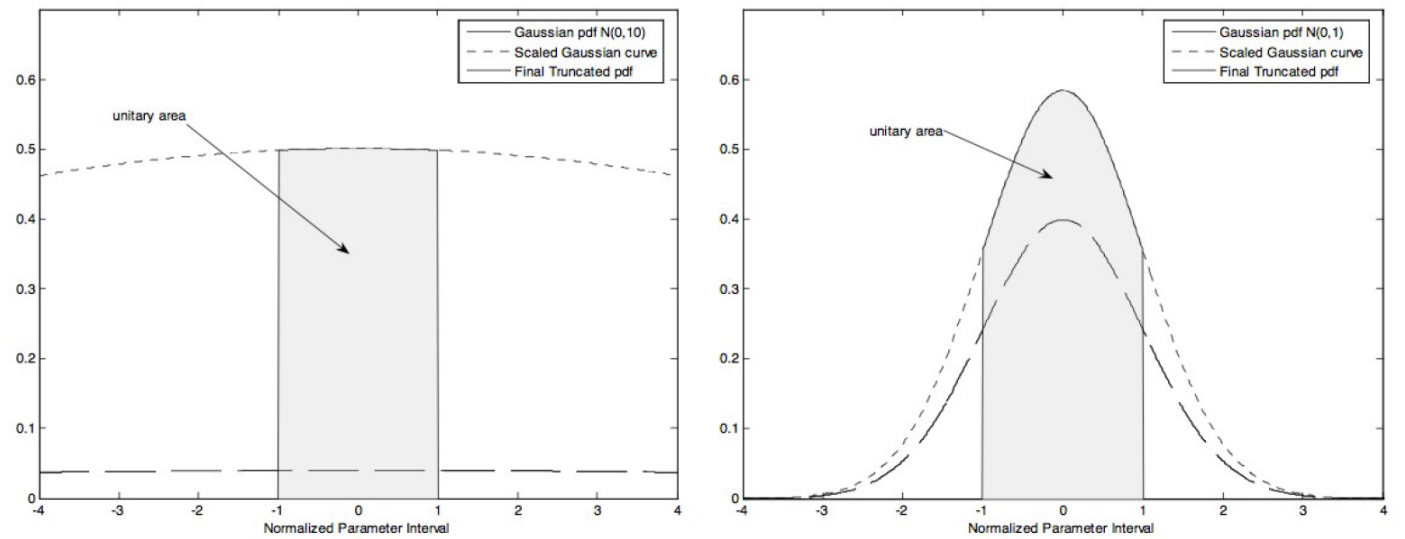
- No analytical expression for f;
- noisy system;
- requires optimisation on-the-fly.



Compact optimisation [12]
(see [13] for details on cDE-light)

Population is replaced with a probabilistic model

(*CLICK HERE FOR DEMO*)



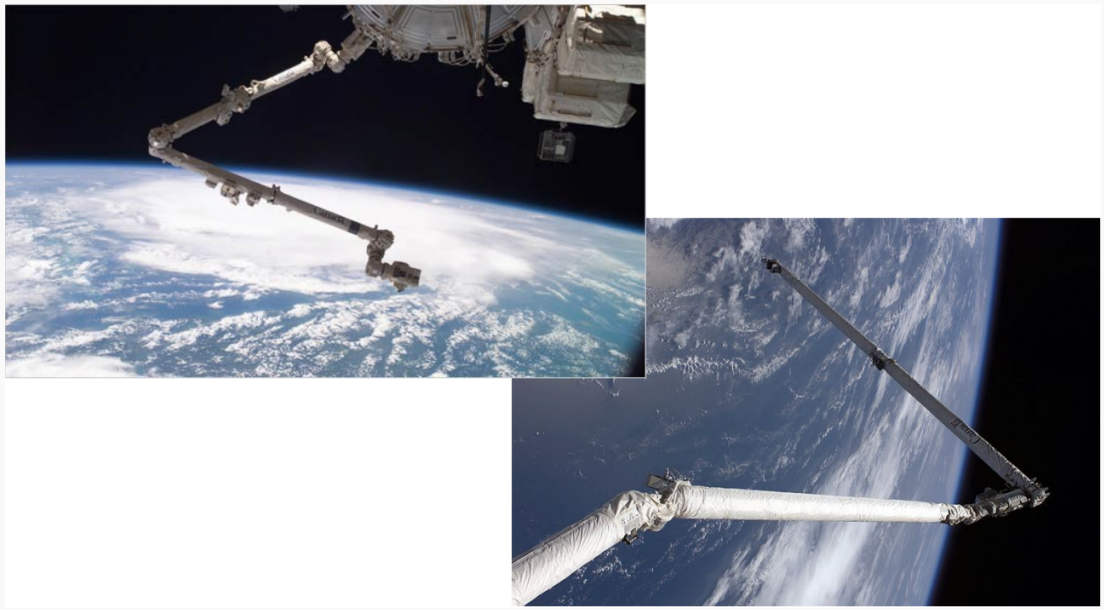
the model is updated after sampling a new individual

$$(\sigma^{t+1})^2 = (\sigma^t)^2 + (\mu^t)^2 - (\mu^{t+1})^2 + \frac{1}{N_p}(winner^2 - loser^2)$$

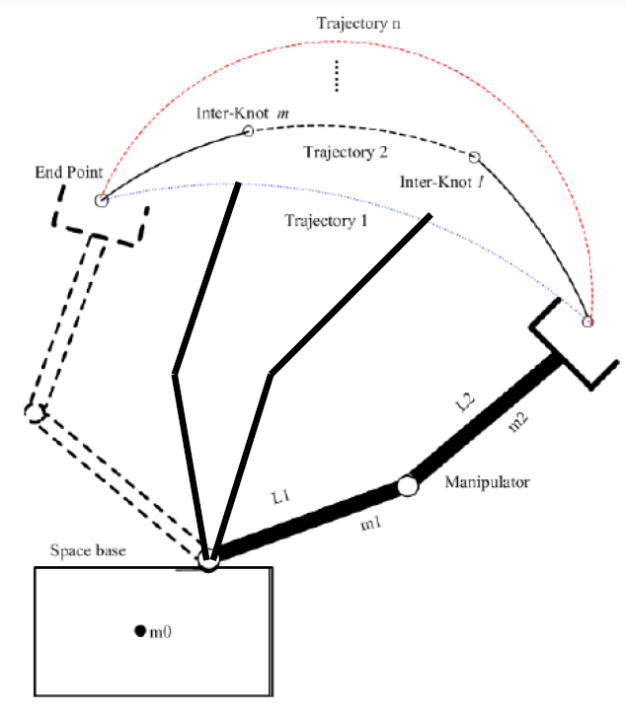
$$\mu^{t+1} = \mu^t + \frac{1}{N_p}(winner - loser)$$

Most algorithms have a compact version, see [12], but we focus on cDE and its "light" variant [13].

Case Study II
Space robot Optimisation [11]



- Same agorithmic issues of Case Study I



- Trajectory must planned not to deviate the spacecraft!;
- trajectory is controlled by imposing $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$ to each joint;
- 3 joints $\times$ 3 inputs $\times$ 2 knots $=$ 18 variables!

Case Study II
Defining the fitness function

Free-floating environment

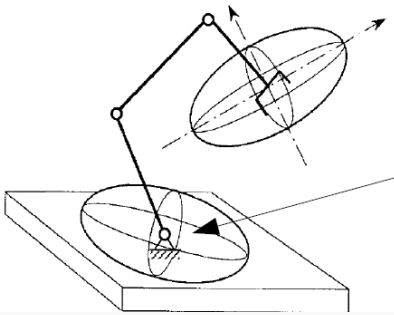
Mutual disturbance between base and end-effector:

$F_B = N^{-1} F_E \leftrightarrow F_E = N F_B$

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = J_i(q)\dot{q}$$

$$\begin{bmatrix} P \\ L \end{bmatrix} = H_B \begin{bmatrix} V_0 \\ \Omega_0 \end{bmatrix} + H_m \dot{q}$$

$$\Rightarrow \begin{bmatrix} V_0 \\ \Omega_0 \end{bmatrix} = -H_B{}^{-1} H_m \dot{q}$$



Fitness function: minimize the integral over time of the norm of the acceleration vector on the base

Thanks for listening

Any questions?



References I

[1] **F. Caraffini, G. Iacca, F. Neri, and E. Mininno.**
The importance of being structured: A comparative study on multi stage memetic approaches.
In *Computational Intelligence (UKCI), 2012 12th UK Workshop on*, pages 1–8, Sept 2012.

[2] **F. Caraffini, G. Iacca, F. Neri, and E. Mininno.**
Three variants of three stage optimal memetic exploration for handling non-separable fitness landscapes.
In *Computational Intelligence (UKCI), 2012 12th UK Workshop on*, pages 1–8, Sept 2012.

[3] **Fabio Caraffini.**
Algorithmic issues in computational intelligence optimization: from design to implementation, from implementation to design.
Number 243 in Jyväskylä studies in computing. University of Jyväskylä, 5 2016.

References II

[4] **Fabio Caraffini, Ferrante Neri, and Michael Epitropakis.**
Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain.
Information Sciences, 477:186 – 202, 2019.

[5] **Fabio Caraffini, Ferrante Neri, Giovanni Iacca, and Aran Mol.**
Parallel memetic structures.
Information Sciences, 227(0):60–82, 2012.

[6] **Fabio Caraffini, Ferrante Neri, and Lorenzo Picinali.**
An analysis on separability for memetic computing automatic design.
Information Sciences, 265(0):1 – 22, 2014.

[7] **Fabio Caraffini, Ferrante Neri, and Ilpo Poikolainen.**
Micro-differential evolution with extra moves along the axes.
In *Differential Evolution (SDE), 2013 IEEE Symposium on*, pages 46–53. IEEE, 2013.

[8] **L. J. Fogel, A. J. Owens, and M. J. Walsh.**
Artificial Intelligence through Simulated Evolution.
John Wiley, 1966.

References III

[9] **Lawrence J Fogel, AJ Owens, and MJ Walsh.**
On the evolution of artificial intelligence(artificial intelligence generated by natural evolution process).
In *National Symposium on Human Factors in Electronics, 5th, San Diego, California*, pages 63–76, 1964.

[10] **John Holland.**
Adaptation in Natural and Artificial Systems.
University of Michigan Press, 1975.

[11] **G. Iacca, F. Caraffini, F. Neri, and E. Mininno.**
Robot base disturbance optimization with compact differential evolution light.
In *EvoApplications*, pages 285–294, April 2012.

[12] **Giovanni Iacca and Fabio Caraffini.**
Compact optimization algorithms with re-sampled inheritance.
arXiv preprint arXiv:1809.04343, 2018.

References IV

[13] **Giovanni Iacca, Fabio Caraffini, and Ferrante Neri.**
Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead.
Journal of Computer Science and Technology, 27(5):1056–1076, 2012.

[14] **Giovanni Iacca, Fabio Caraffini, and Ferrante Neri.**
Memory-saving memetic computing for path-following mobile robots.
Applied Soft Computing, 13(4):2003–2016, 2013.

[15] **Giovanni Iacca, Ferrante Neri, Ernesto Mininno, Yew-Soon Ong, and Meng-Hiot Lim.**
Ockham's razor in memetic computing: Three stage optimal memetic exploration.
Information Sciences, 188:17 – 43, 2012.

[16] **John R Koza.**
Concept formation and decision tree induction using the genetic programming paradigm.
In *Parallel Problem Solving from Nature*, pages 124–128. Springer, 1990.

References V

[17] **John R Koza.**
Genetic programming: on the programming of computers by means of natural selection, volume 1.
MIT press, 1992.

[18] **John R Koza.**
Genetic Programming: vol. 1, On the programming of computers by means of natural selection, volume 1.
MIT press, 1992.

[19] **Pablo Moscato.**
On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.
Technical Report 826, 1989.

[20] **Ingo Rechenberg.**
Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.
PhD thesis, Technical University of Berlin, 1971.