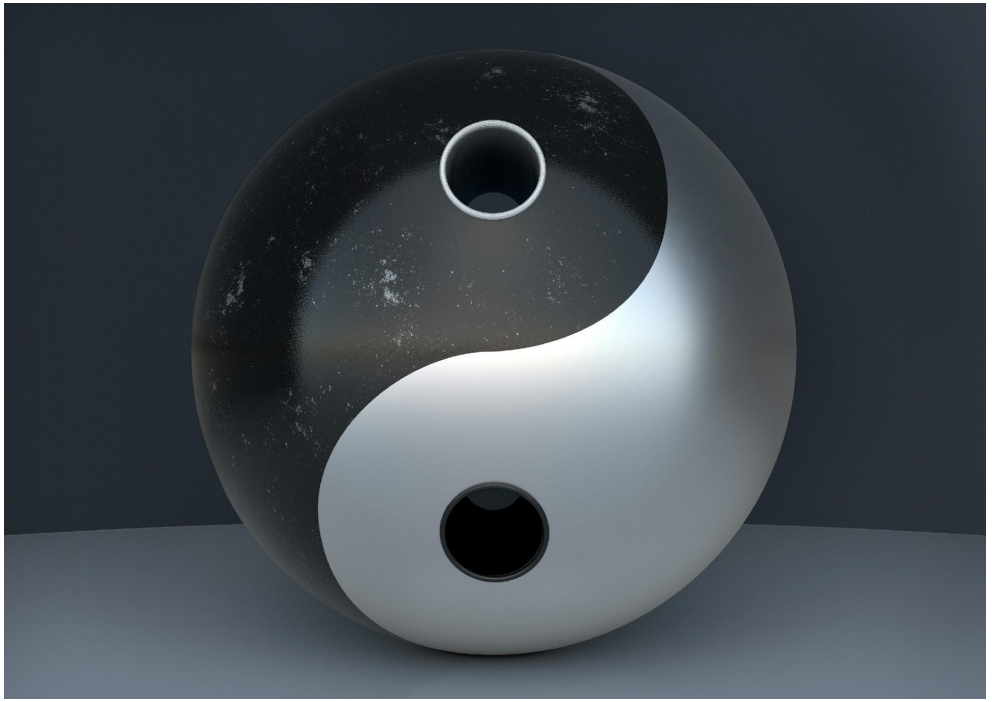
INTELLIGENT OPTIMIZATION FOR SELF-IMPROVING RELIABLE ARTIFICIAL INTELLIGENCE

Roberto Battiti

Intelligent Optimization Association

intelligent-optimization.org



INVITED TUTORIAL

MIC 2026

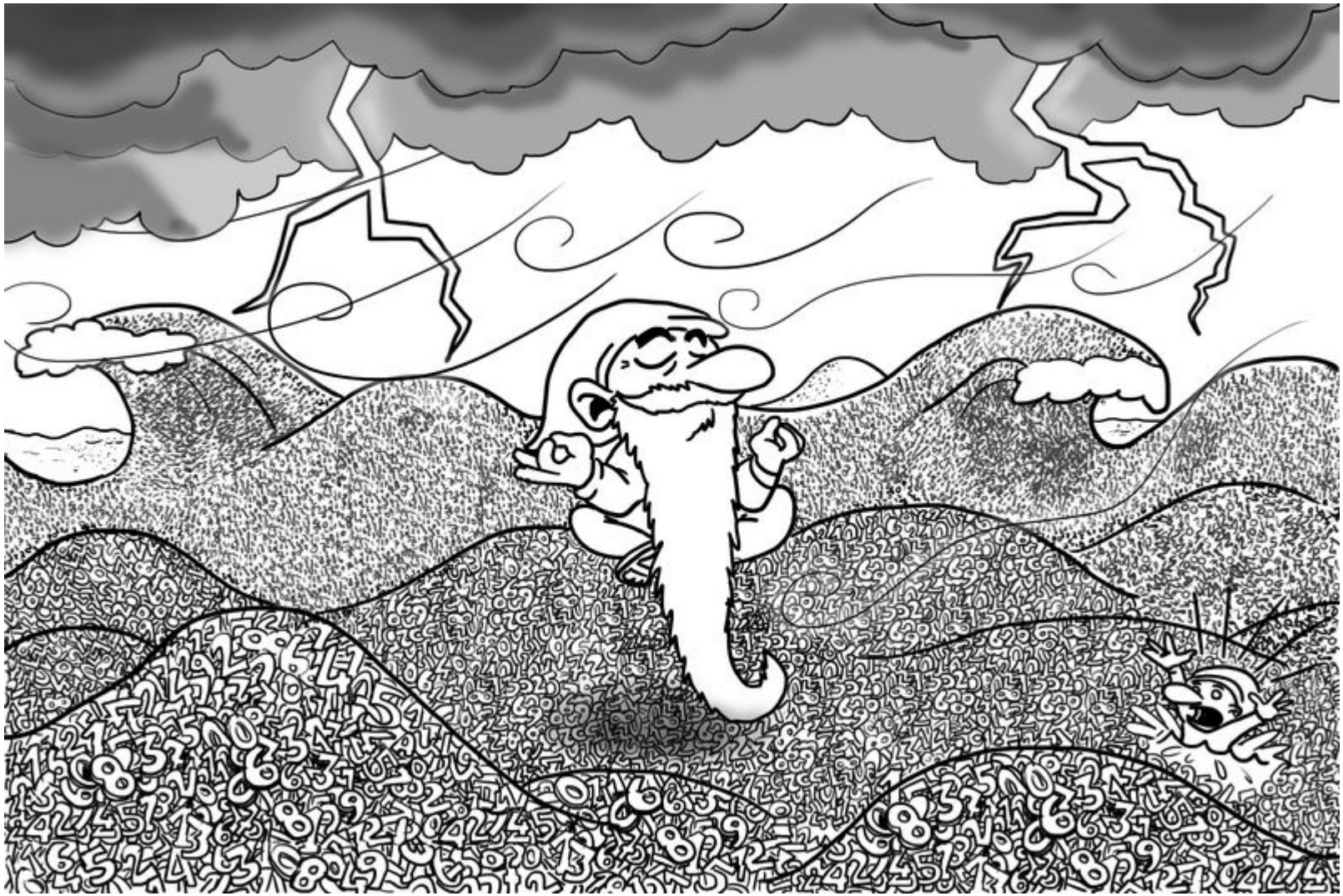
16th Metaheuristics International Conference

Ischia, June 8-11, 2026

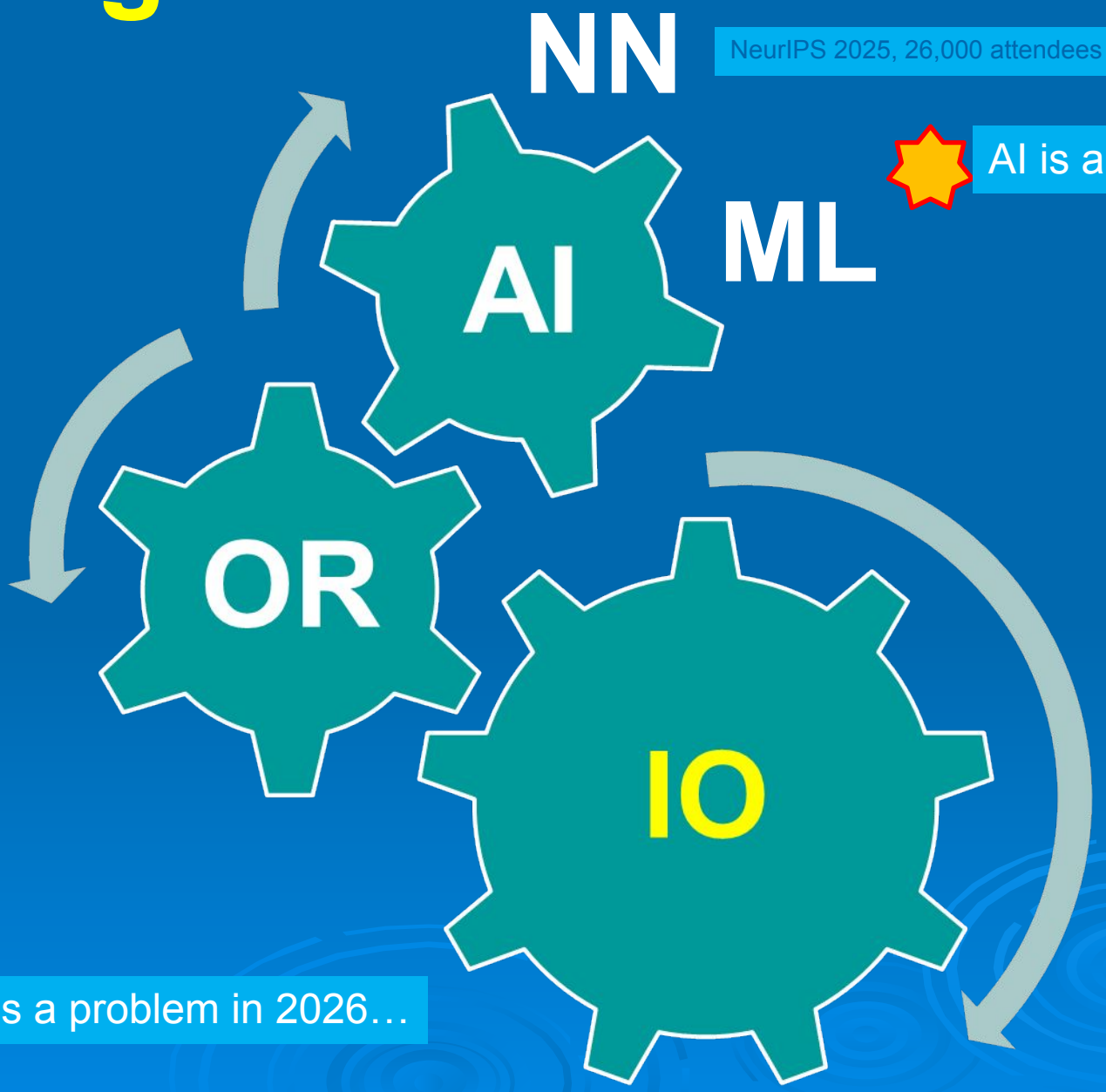
Intelligent Optimization book

Roberto Battiti, Kevin Tierney and Mauro Brunato

(in press) https://intelligent-optimization.org/*iobook/*



Intelligent Optimization: focus on the goal



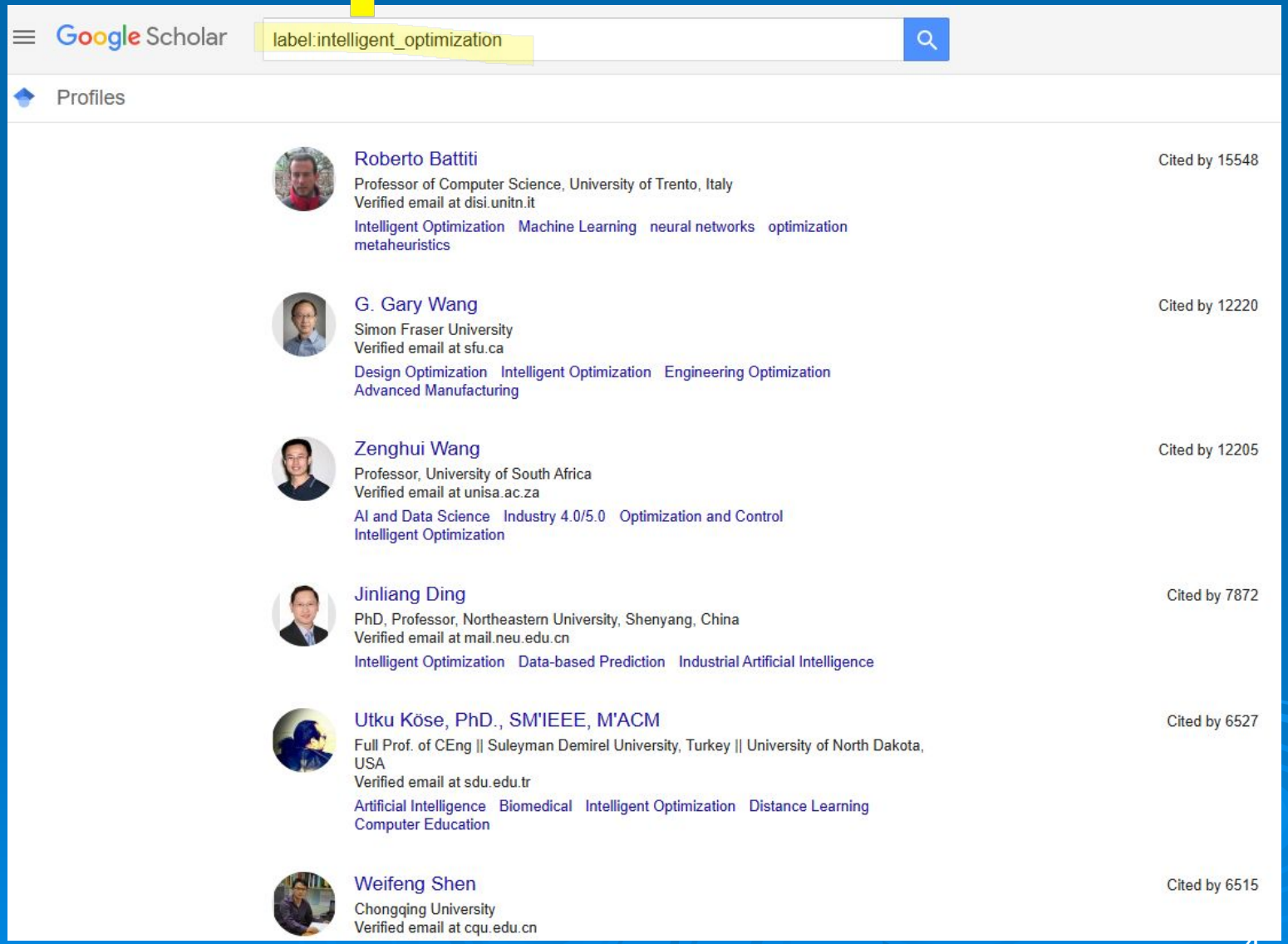
NeurIPS 2025, 26,000 attendees

AI is a misleading term

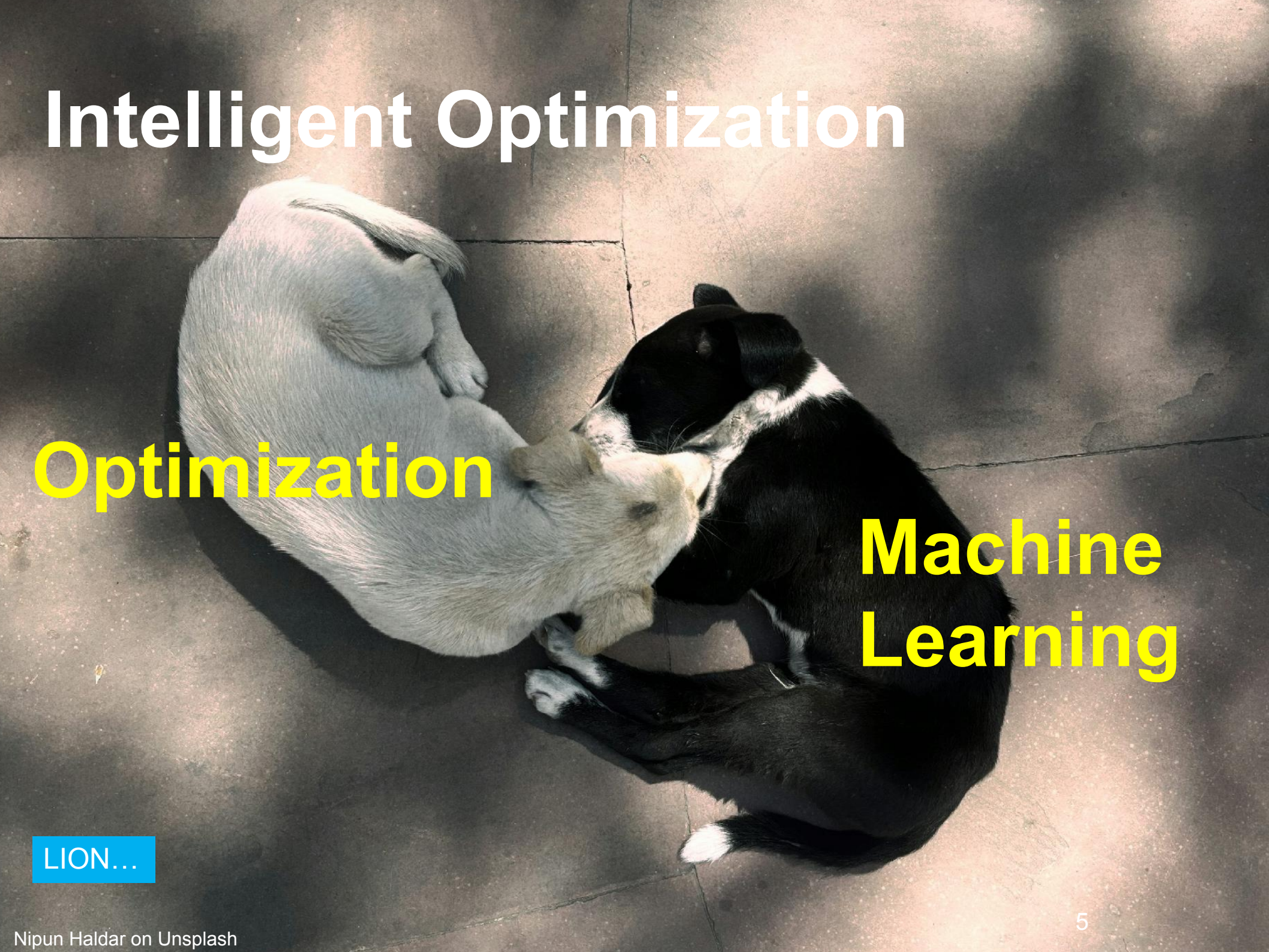
If somebody has a problem in 2026…

Intelligent Optimization

A new keyword in Google scholar



Intelligent Optimization

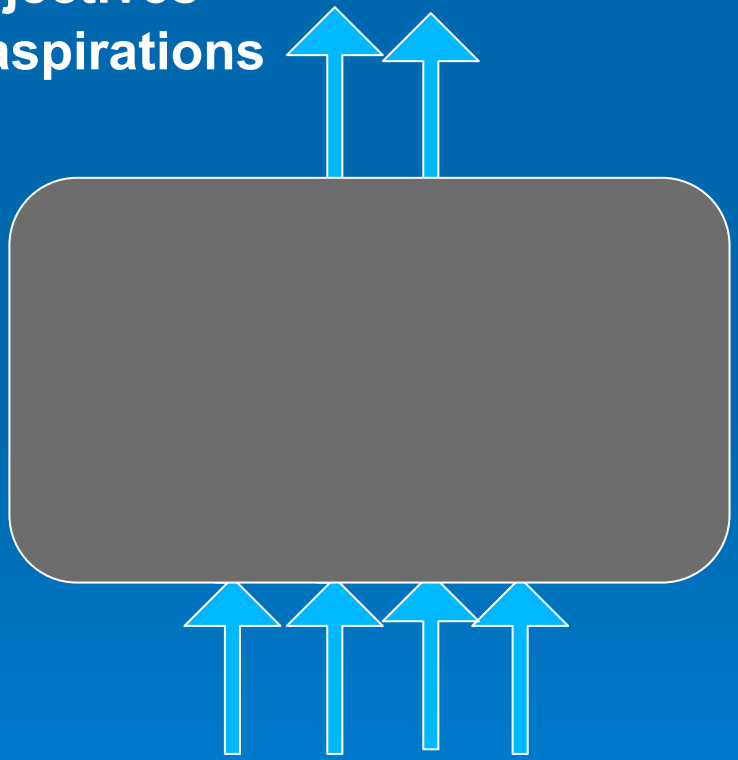


LION…

Nipun Haldar on Unsplash

Real world problems: illuminated or dark?

Some posivite objectives (MOOP)
Combination not clear
Hidden objectives
Dynamic aspirations

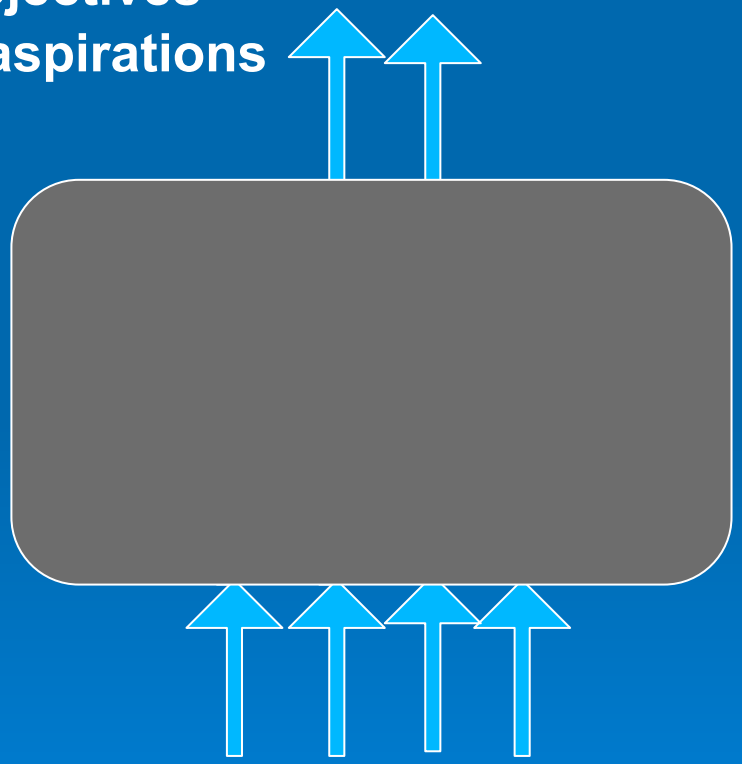


No math formula
Maybe some high-level knowledge and intuition

Many inputs,
noisy,
some irrelevant

Real word: illuminated or dark?

Some posivite objectives (MOOP)
Combination not clear
Hidden objectives
Dynamic aspirations



No math formula
Maybe some
high-level
knowledge
and intuition

Many inputs,
noisy,
some irrelevant

Learn !

Learn !

Learn !

Machine Learning !

Machine Learning for Optimization

Automatically improve/tune/self-tune a flexible optimization algorithm
by learning from
- previous instances (offline)
- previous problems
- instance characteristics

- previous search history (online)
- local properties of fitness landscape

Learn/fine-tune the goal (objective)
by learning from
- previous samples (BO)
- decision maker (MOOP)

Local Search: *the universal building block*

Trial and error

Small modifications (*perturbations*)…

Embarassingly simple, but can *improve* most problems

On a desert island…

Nature-inspired

Are nature-inspired algorithms
an incredibly successful marketing strategy
for local search plus stronger diversification?
See Kenneth Sörensen … metaphor exposed

Occam razor: entia non sunt multiplicanda praeter necessitatem

1000 years from now

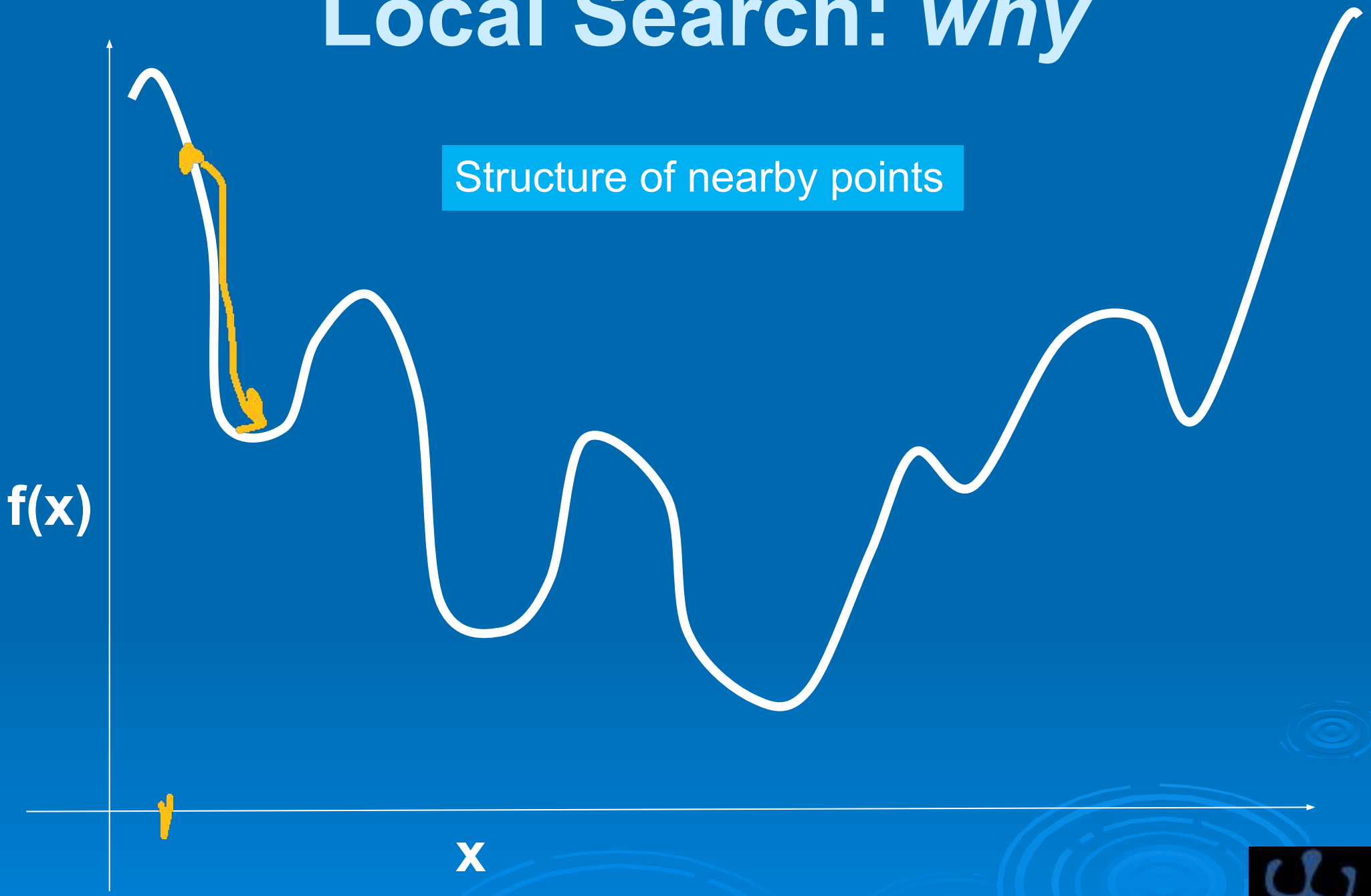
~ten basic parametric **building blocks + learning** x automated tuning and self-tuning

OR

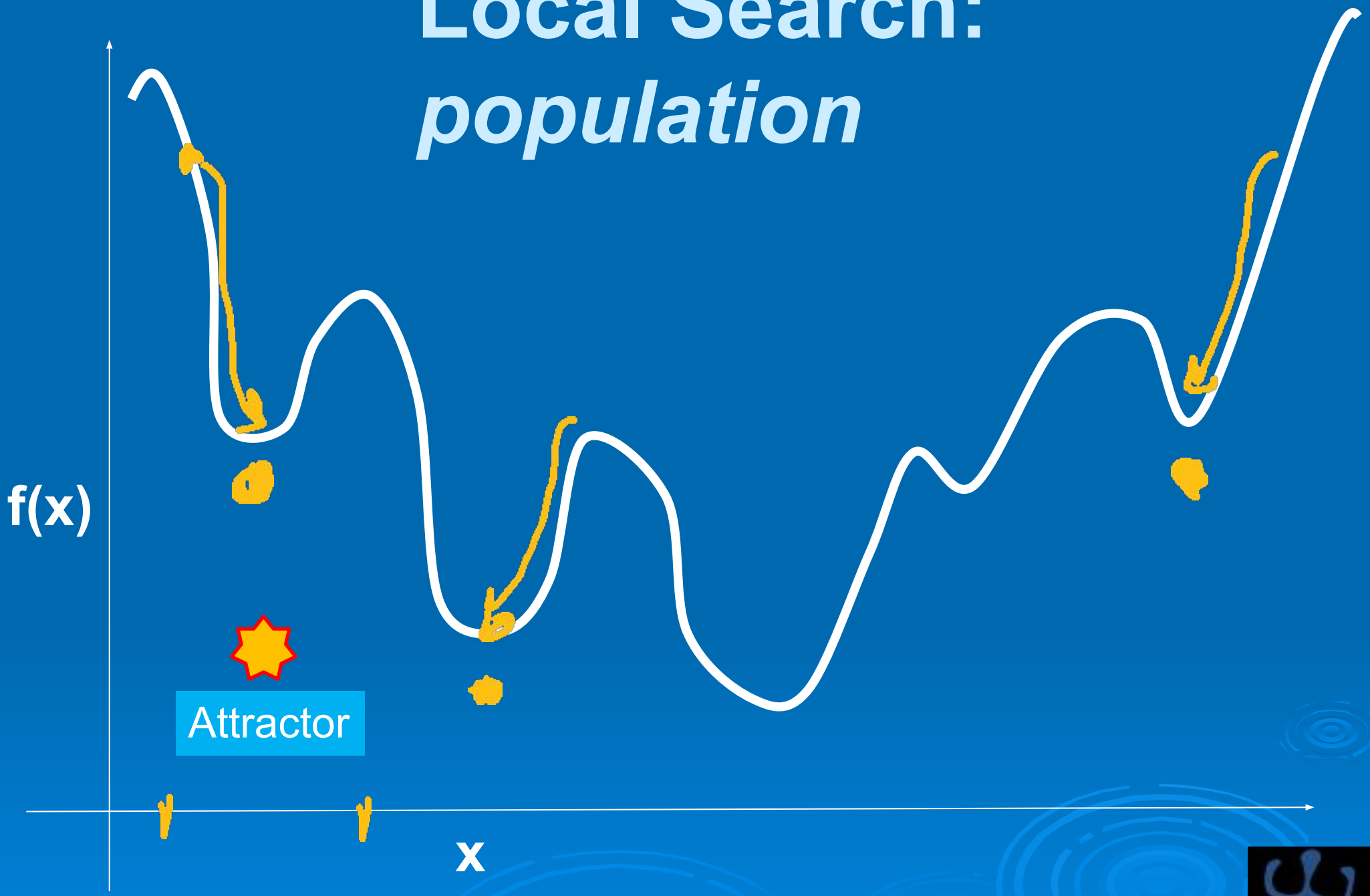
500++ **metaphors**

Local Search: *why*

Structure of nearby points



Local Search: *population*



Local Search: *meta…iterative…*

Structure of nearby *local optima*
Path relinking, x-over, kicks, VNS, TS

Beyond local minima

Accept also worsening moves but…
Determinism => cycles
Stochasticity… random walk?
Simulated Annealing!

No learning: Markov process

Simulated Annealing

$$Y \leftarrow \text{NEIGHBOR}(N(X^{(t)}))$$

$$X^{(t+1)} = \begin{cases} Y & \text{if } f(Y) < f(X^{(t)}) \\ Y & \text{with probability } e^{-\Delta f/T}, X^{(t)} \text{ otherwise} \end{cases}$$

$$T_k \geq \frac{\Gamma}{\log(k + k_0)} \longrightarrow \lim_{k \to \infty} \mathbb{P}\{X^{(k)} \in \mathcal{S}^*\} = 1$$

- **Asymptotic convergence is irrelevant**
- Slow “speed of convergence” to the stationary distribution... Complete enumeration can be faster!

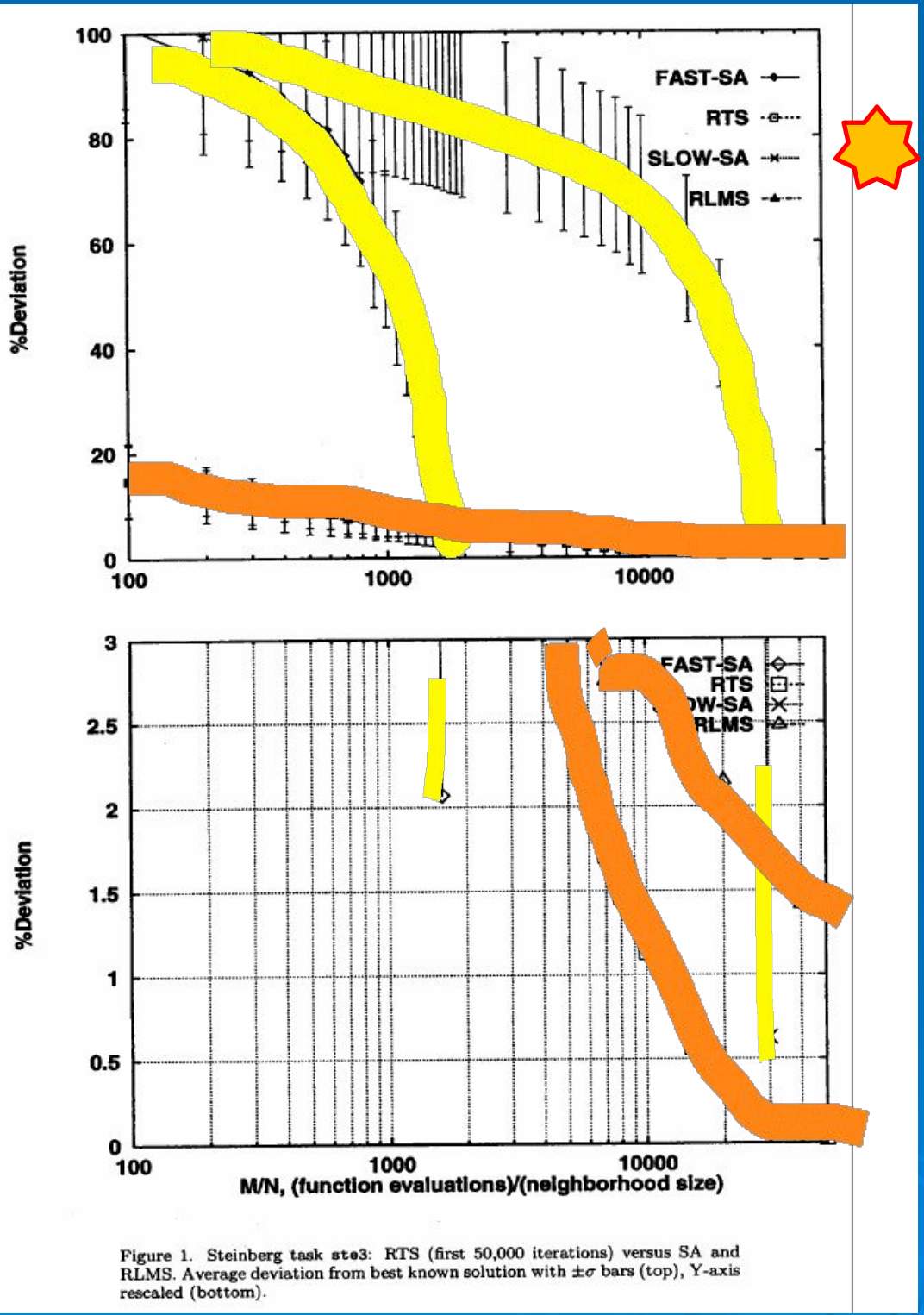
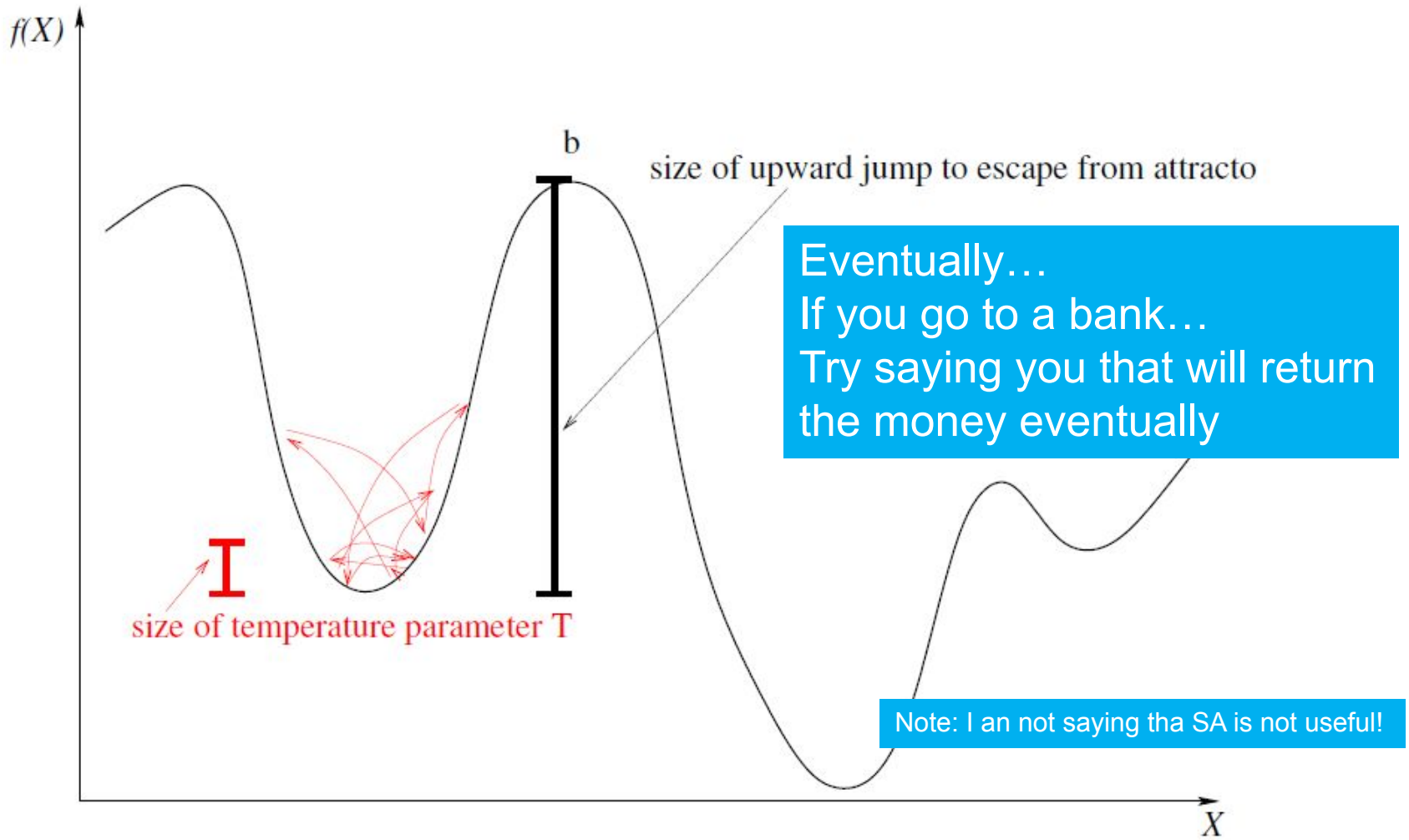


Figure 1. Steinberg task ste3: RTS (first 50,000 iterations) versus SA and RLMS. Average deviation from best known solution with $\pm\sigma$ bars (top), Y-axis rescaled (bottom).

R. Battiti and G. Tecchiolli. ***Simulated annealing and tabu search in the long run: a comparison on qap tasks.*** *Computer and Mathematics with Applications, 28(6):1--8, 1994.*

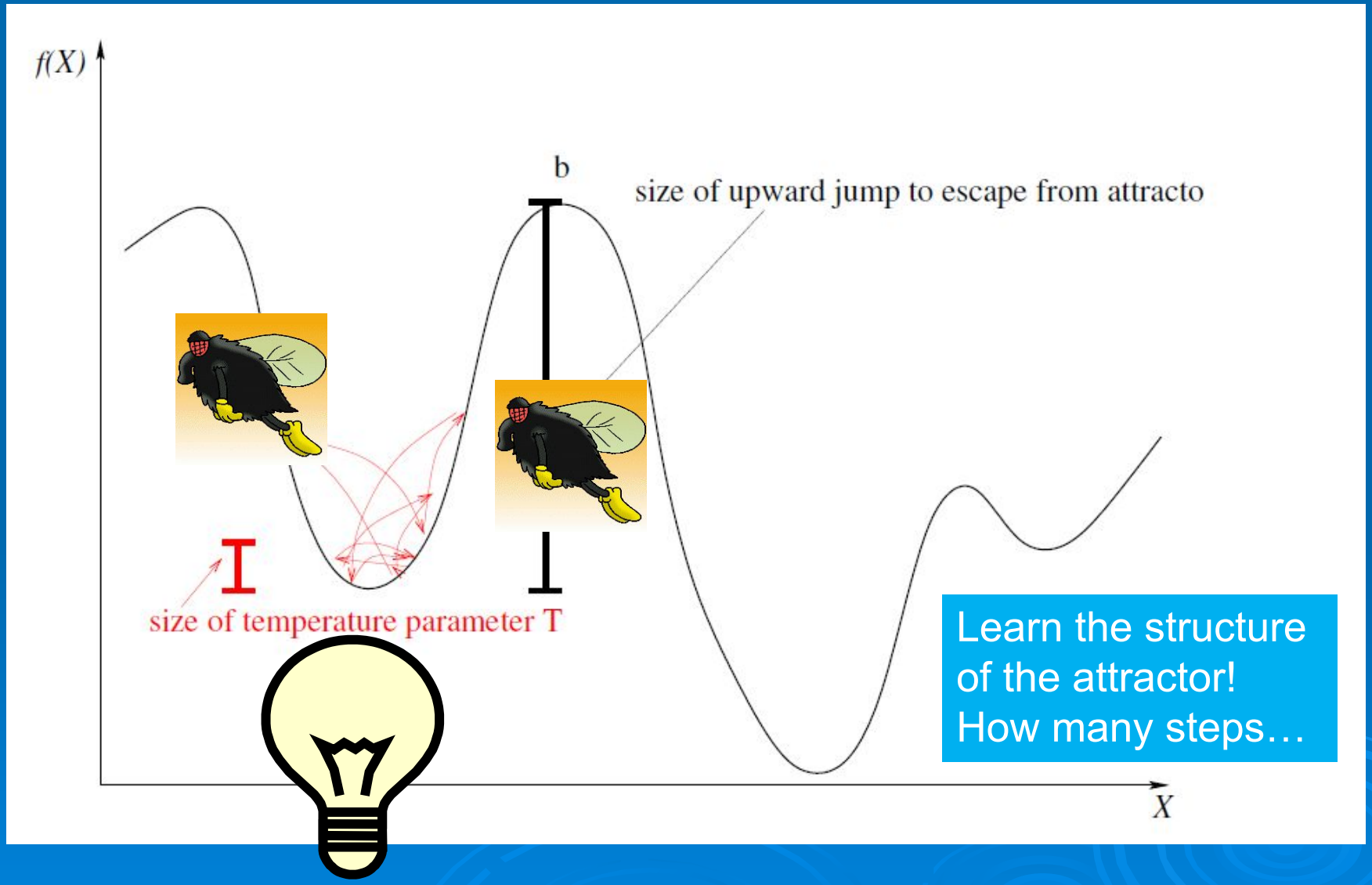
Memory-less: no self-observation/learning



Eventually…
If you go to a bank…
Try saying you that will return the money eventually

Note: I an not saying tha SA is not useful!

Memory-less: no self-observation/learning



Reactive Search Optimization: *learning* while optimizing

- free parameters

  Algorithm(T)

- **Parameter tuning – and self-tuning – is a typical "learning" process**

Complexity is shifted
Final user algorithm developer
Reactive search optimization needs proactive researchers

Reactive Search Optimization

integration of online machine learning techniques for local search heuristics.

The word ***reactive*** hints at a ready response to events *during* the search through an internal online feedback loop for the *self-tuning* of critical parameters.

Biological systems are highly adaptive; they use signals coming in from receptors and sensors to fine-tune their functioning. Adaptivity is a facet of the **reactivity** of such systems.

An example: reactive prohibition-based local search

- X is the search space

0010110001000

- Neighborhood

$$N(X^{(t)}) = \{X \in \mathcal{X} \text{ such that } X = \mu_i \circ X^{(t)}, i = 0, \ldots M\}$$

- Search **trajectory** $X^{(0)}, \ldots, X^{(t+1)}$

$$Y \leftarrow \text{BEST-NEIGHBOR}(\, N(X^{(t)}) \,)$$

$$X^{(t+1)} = \begin{cases} Y & \text{if } f(Y) < f(X^{(t)}) \\ X^{(t)} & \text{otherwise (search stops)} \end{cases}$$

Prohibition-based local search

Beyond local optima … escape from previously visited basins of attraction around a local minimizer (diversification)

- Without learning
  - (random) restart
  - Stochasticity (SA)
- With **learning:** use data accumulated during the previous phase of the search
  - direct diversification through **prohibitions**

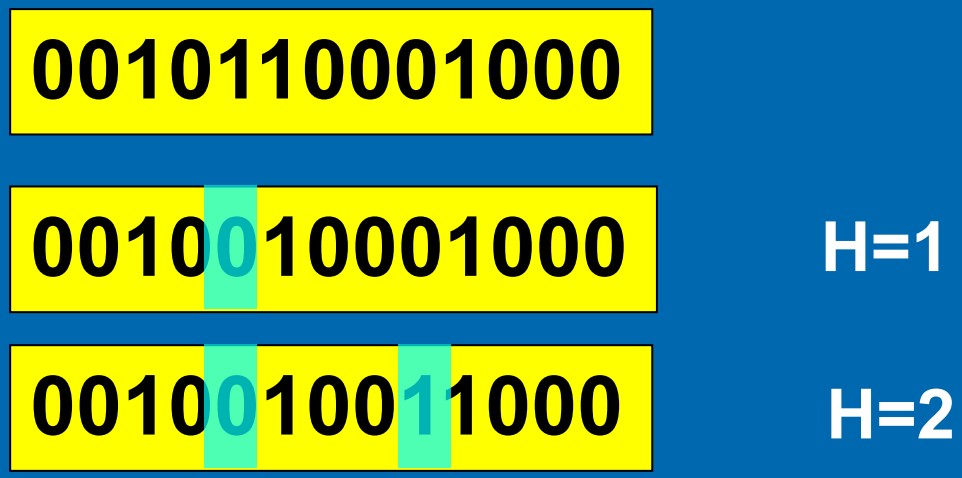
Prohibition-based local search

- Prohibition-based diversification:
  - Steiglitz Weiner- *denial* strategy for TSP (once common features are detected in many suboptimal solutions, they are *forbidden*) (opposite to *reduction* strategy: all edges that are common to a set of local optima are fixed)
  - Lin-Kernighan for graph partitioning (fixing nodes)
  - Tabu Search (Fred Glover)
  - Steepest Ascent Mildest Descent (Hansen – Jaumard)

One can have TS without tabu list, with «memory» of one int per bit

Prohibition-based local search (3)

- diversification through prohibitions



Which **theory**? (not only *horse-racing*)

Distinguish **policies** from **implementation** (lists…)
Use the abstraction of **dynamical systems** (trajectory)

Fundamental relationship between **prohibition and diversification** (Battiti, 1999)

Binary strings

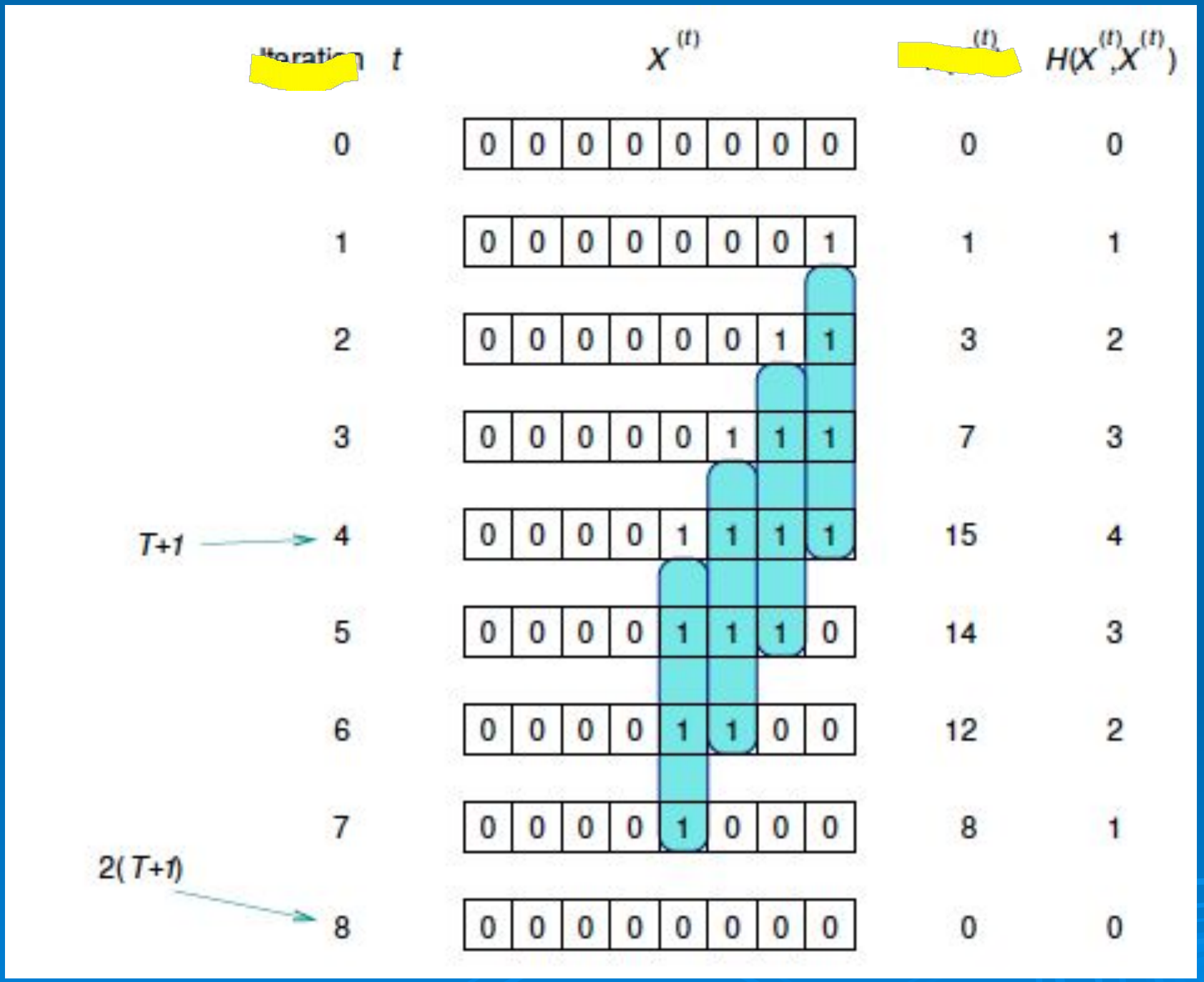
- The Hamming distance H between a starting point and successive point along the trajectory is strictly increasing for $T+1$ steps.

$$H(X^{(t+\Delta t)}, X^{(t)}) = \Delta t \quad \text{for} \quad \Delta t \leq T+1$$

- The minimum repetition interval R along the trajectory is $2(T+1)$.

$$X^{(t+R)} = X^{(t)} \Rightarrow R \geq 2(T+1)$$

Open problem: extend to other local moves (like exchanges for permutations)



An example of the relationship between prohibition T , and diversification measured by the Hamming distance H(X(t);X(0)) . T = 3 in the example

«magic springs»
Universal disentanglement

Prohibition Mechanisms

- Strict-TS

$$N_A(X^{(t+1)}) = \{X \in N(X^{(t+1)}) \text{ s. t. } X \notin \{X^{(0)}, ..., X^{(t+1)}\}\}$$

- Fixed-TS

$$N_A(X^{(t)}) = \{X = \mu \circ X^{(t)} \text{ s. t. } \textsc{LastUsed}(\mu^{-1}) < (t - T)\}$$

- Reactive-TS

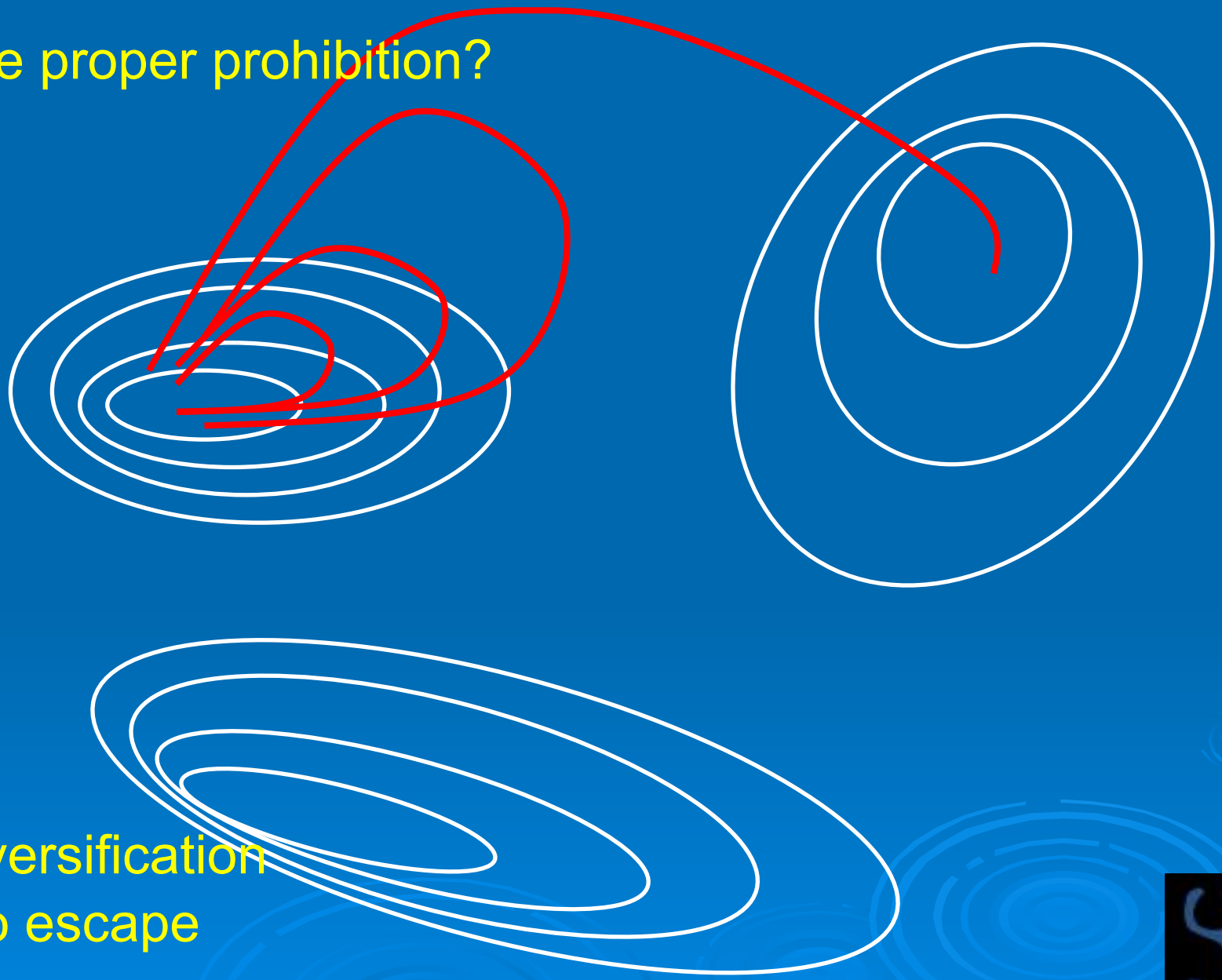
Are the dynamical systems comparable ?

Or qualitative differences ?

Distinguish policies from mechanisms

Reactive Prohibition-based search

What is the proper prohibition?



Minimal diversification
sufficient to escape

Reactive prohibitions

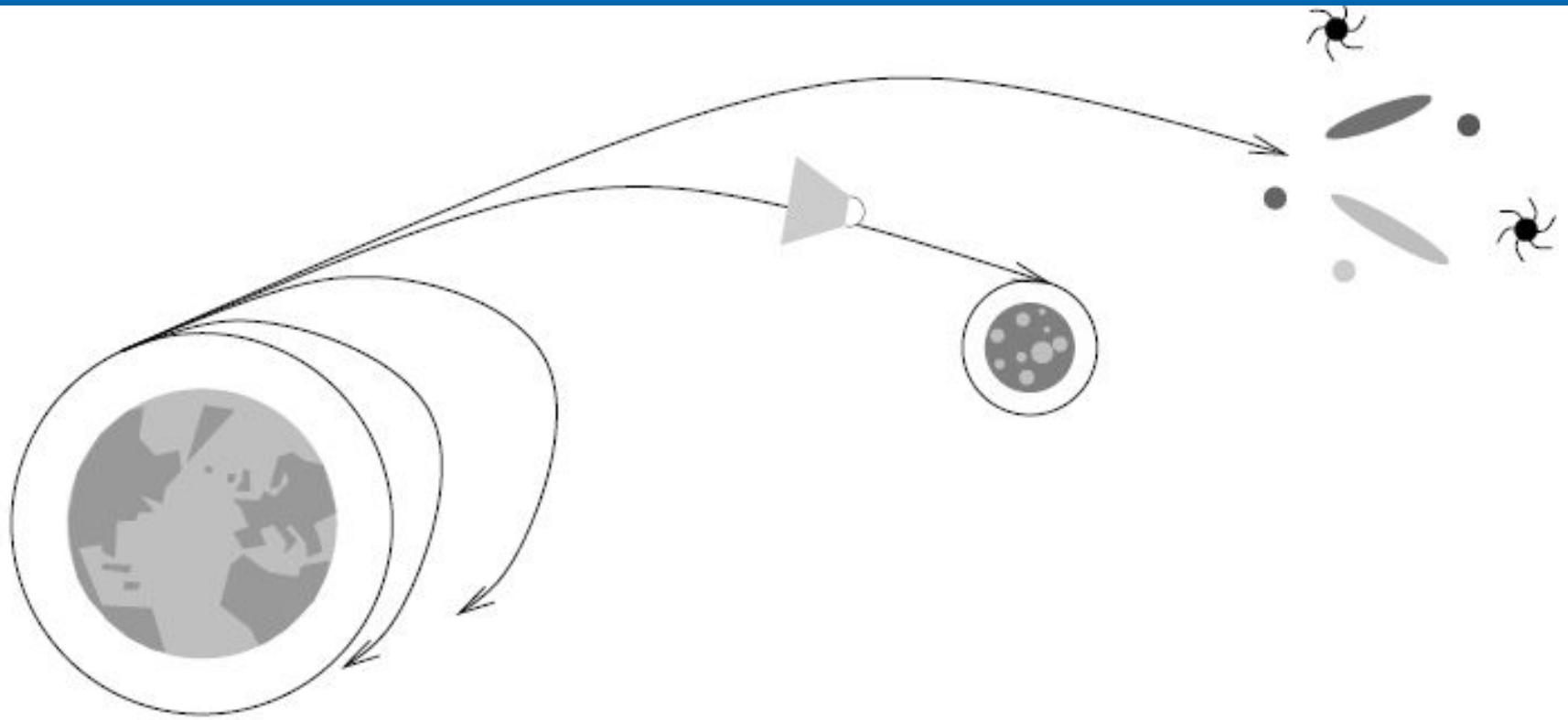
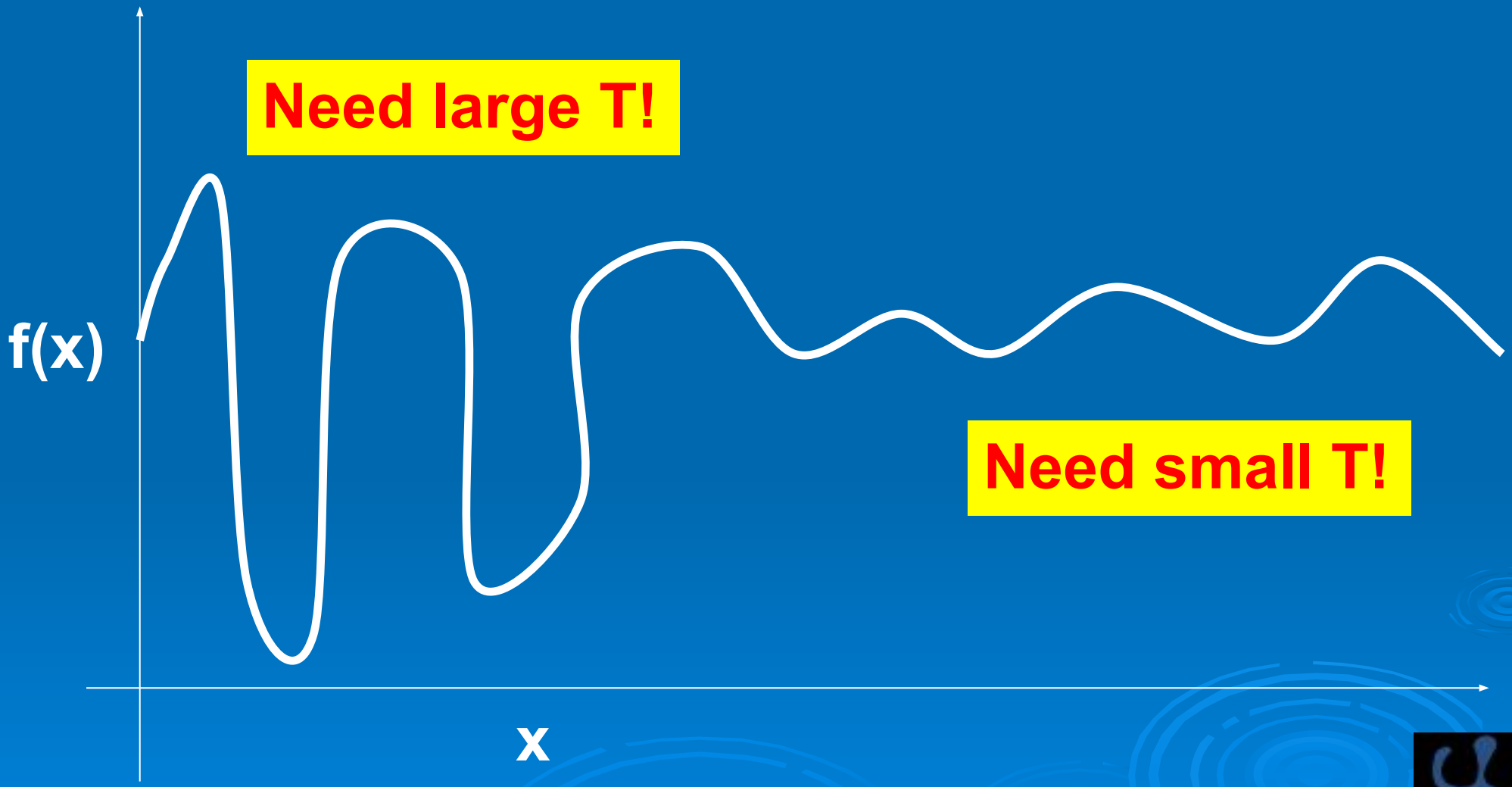


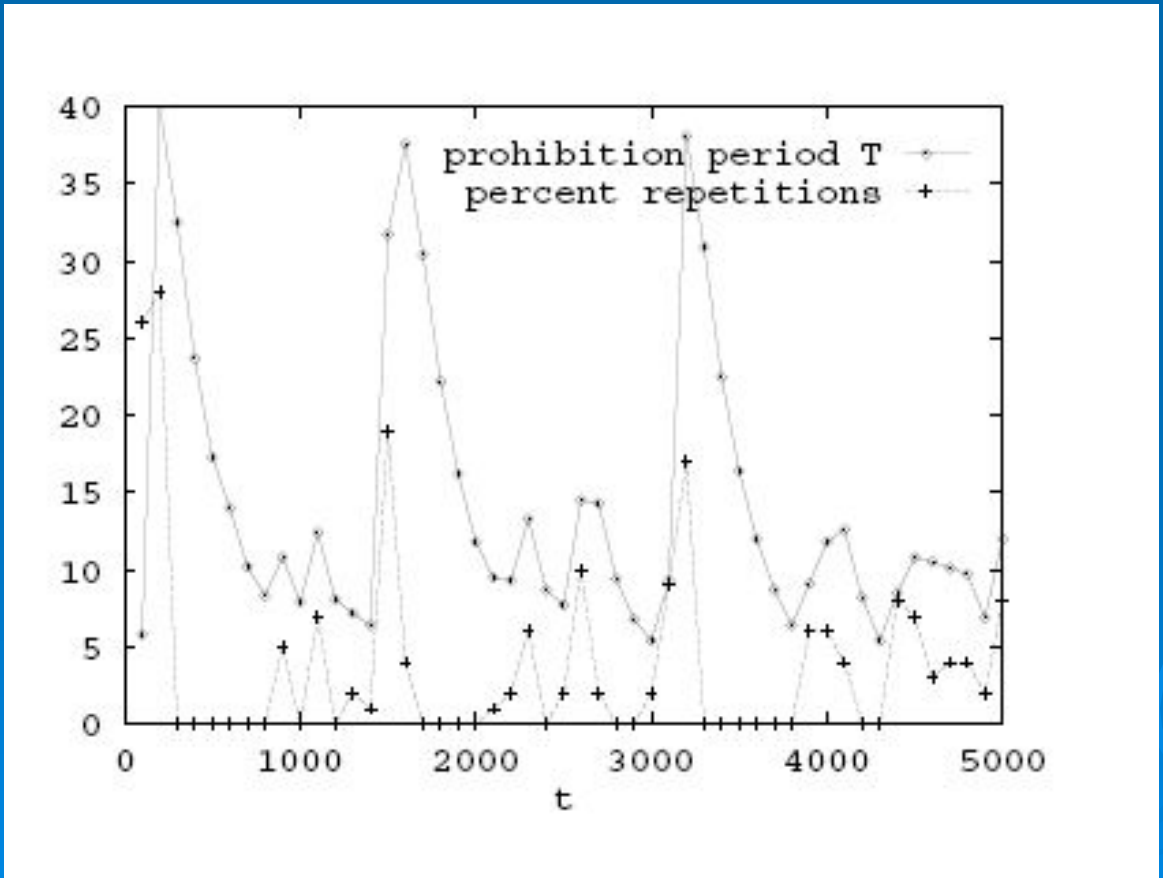
Figure 4.3: Applying the minimum diversification needed to escape the local attractor. Top: if too little diversification is applied, the trajectory falls back to the same minimum; if diversification is excessive, nearby minima may not be visited. Bottom: the same concept applied to space travel: the right amount of diversification (plus ballistic calculations) allows the capsule to reach a suitable moon orbit; too little and the capsule falls back, too much and the target is missed.

Motivations for a dynamic T



Self-adjusted T

- T=1 at the beginning
- Increase T if evidence of *entrapment*
- Decrease T if evidence disappears

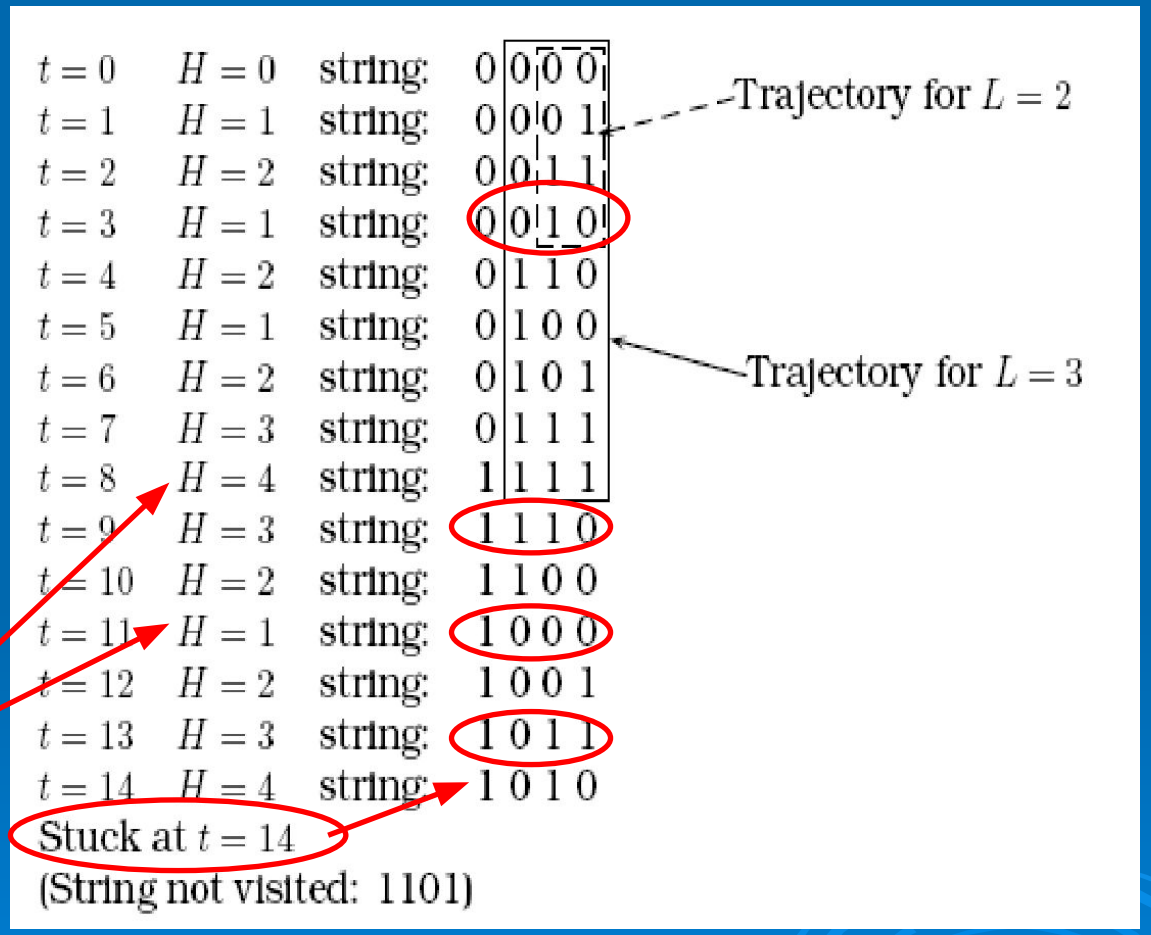


How to escape from an attractor

- Cost= Hamming distance from 00000
- Strict-TS

$$H(t) \leq \lfloor \log_2(t) \rfloor + 1$$

Non so intensifying…

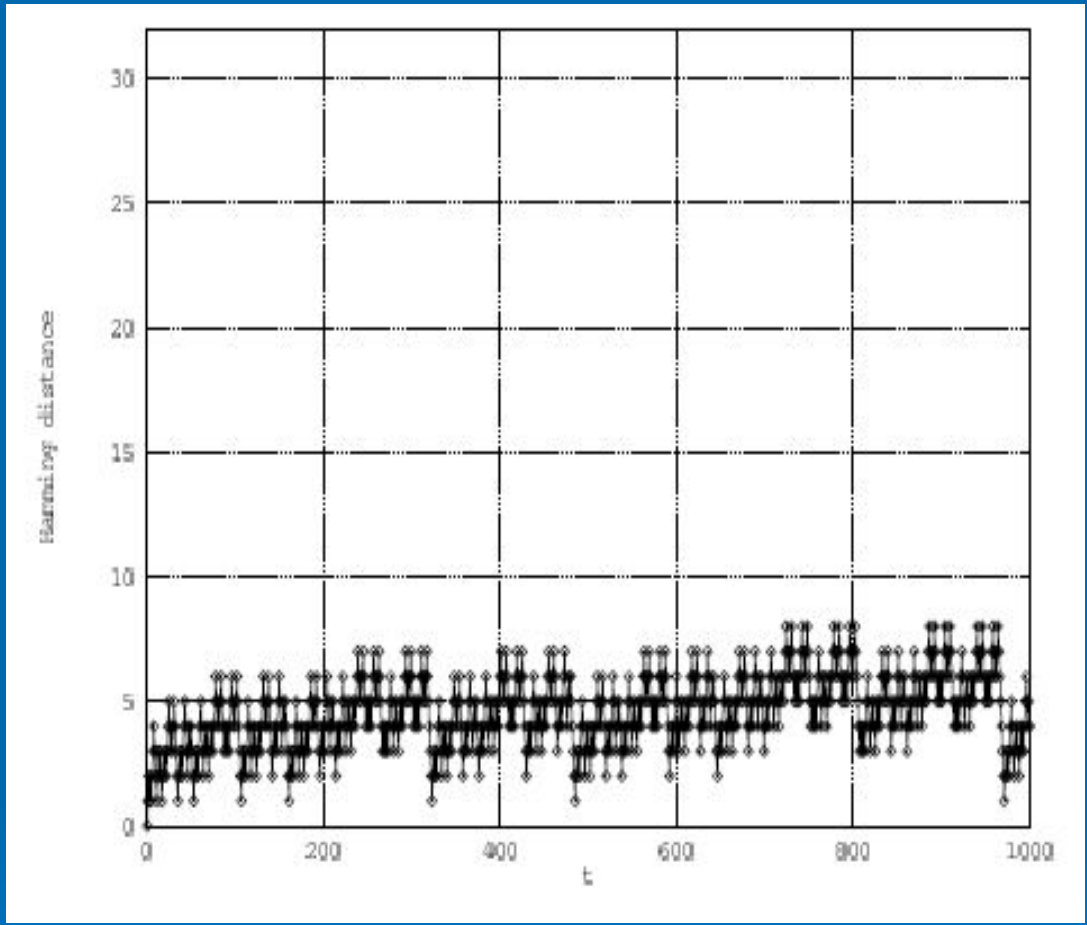


How to escape from an attractor

- Strict-TS

$$C_H = \sum_{i=0}^{H} \binom{L}{i}$$

$$C_H >> 2^H, \text{ if } H << L$$



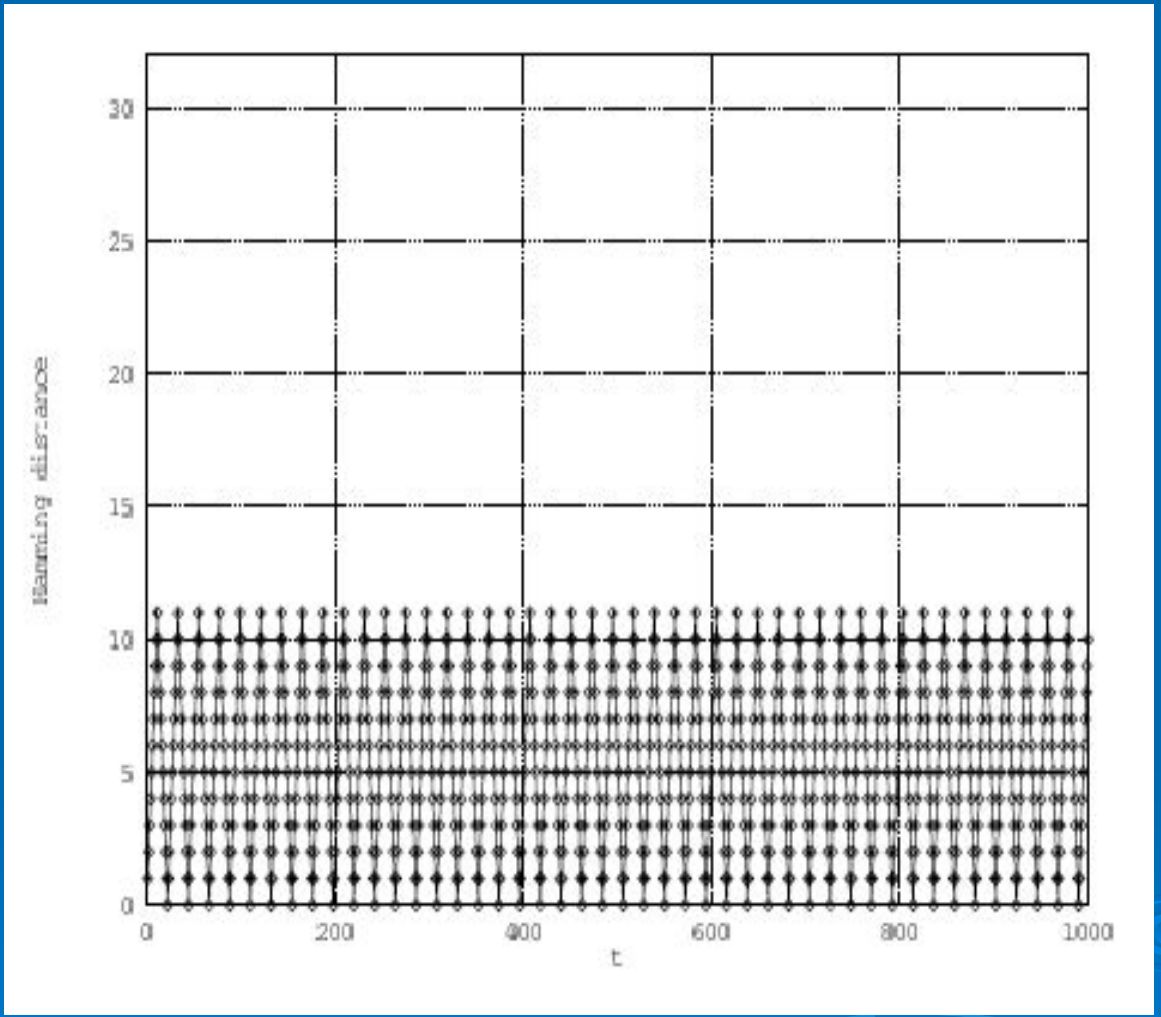
- Curse of dimensionality, “basin filling”

$$L = 64,\ C_5 = 8\ 303\ 633,\ C_4 = 679\ 121.$$

How to escape from an attractor

- Fixed-TS

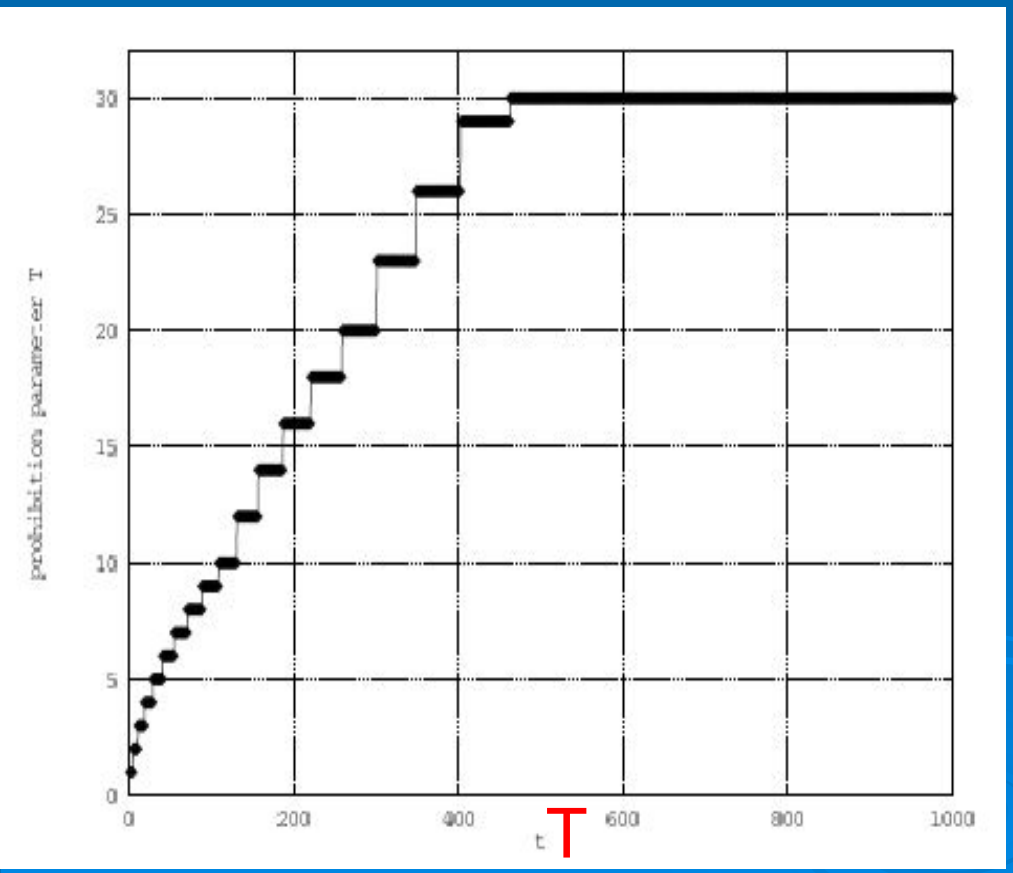
?Sufficient to escape or not ?

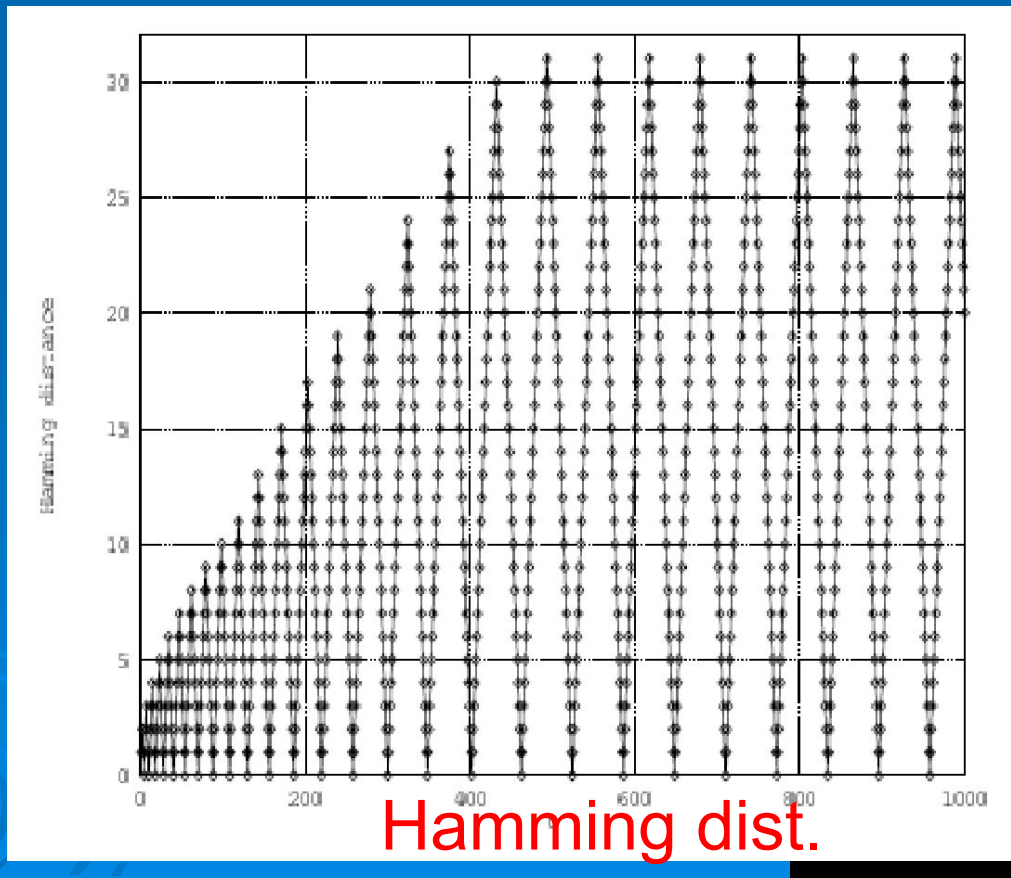


How to escape from an attractor

- Reactive-TS (react when loc. minimum is repeated)

$$\textsc{React}(T) \quad = \quad \min\{\max\{T \times 1.1, T+1\}, L-2\}$$





How to escape from an attractor

- Reactive-TS

$$
\begin{aligned}
t(T) &= \sum_{i=1}^{T} 2(i+1) = 3T + T^2 \\
t(H_{max}) &= \left(H_{max}^2 + H_{max} - 2\right) \\
H_{max}(t) &= \frac{1}{2}\left(\sqrt{9+4t} - 1\right)
\end{aligned}
$$

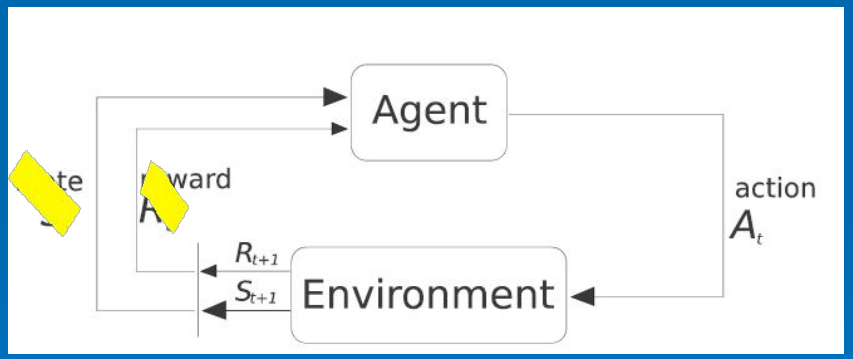
- reachable Hamming distance is approximately O(√t) during the initial steps.
- Qualitative difference: an (optimistic) **logarithmic** increase in the strict algorithm, and a (pessimistic) increase that behaves like the **square root** of the number of iterations in the reactive case.

Other reaction opportunities

- **Variable Neighborhood Search**
- **Iterated Local Search**, kicks, …
- **Annealing schedule**
- **Objective function modifications**, tunneling, dynamic local search
- **Model-based search**
- **Different randomness levels** (SAT)
- **Algorithm portfolios and restart**
- **Racing**

The dream of more automation: Reinforcement Learning

- From a *designed* policy to one *discovered*
- model the parameter-tuning policy as a **Markov Decision Process** (*state* summarizes relevant information about the recent history of the search),
- near-optimal policy by using the **Least Squares Policy Iteration** (LSPI) method



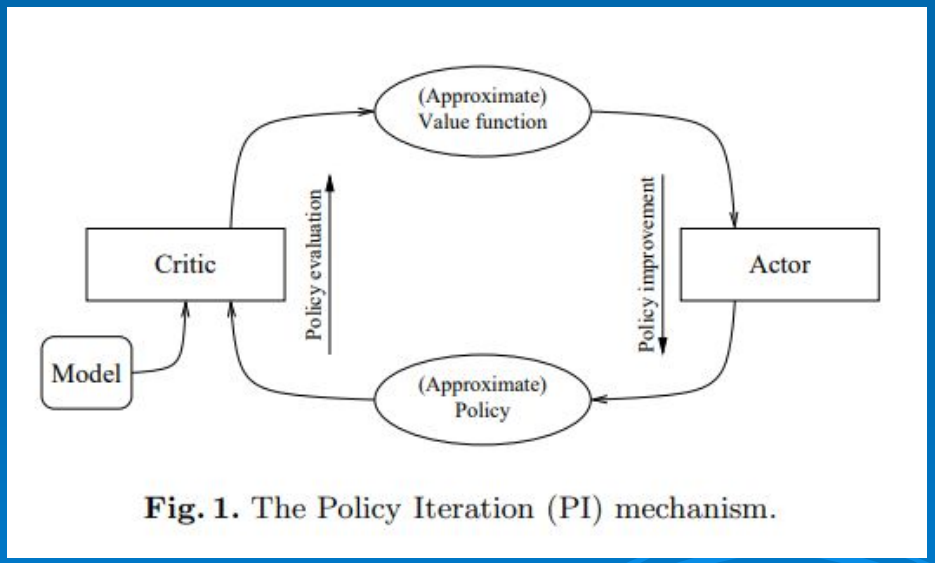


Fig. 1. The Policy Iteration (PI) mechanism.

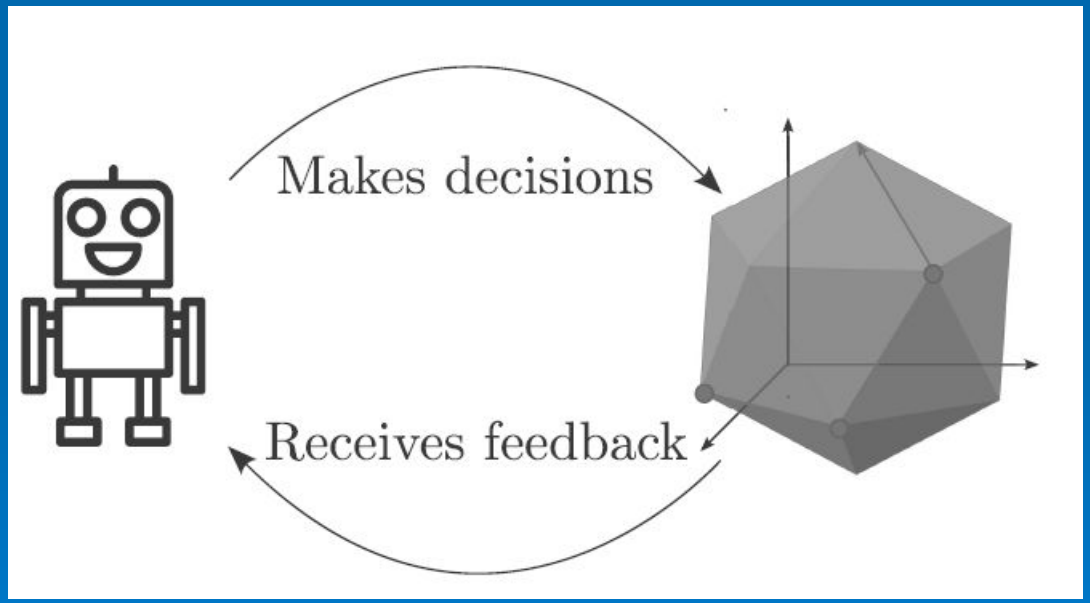
Roberto Battiti, Mauro Brunato, Paolo Campigotto.
Learning while Optimizing an Unknown Fitness Surface.
*Proc. 2nd Learning and Intelligent Optimization Workshop (LION2007 II, Trento, Italy, December 2007), pag. 25-40. LNCS 5313 - Springer, December **2008**.*

S Adriaensen, A Biedenkapp, G Shala, N Awad, T Eimer, M Lindauer, F Hutter
Automated Dynamic Algorithm Configuration
Journal of Artificial Intelligence Research 75 (**2022**) 1633-1699

Neural Combinatorial Optimization
RL plus Deep Neural Networks

Directly build new heuristics from scratch

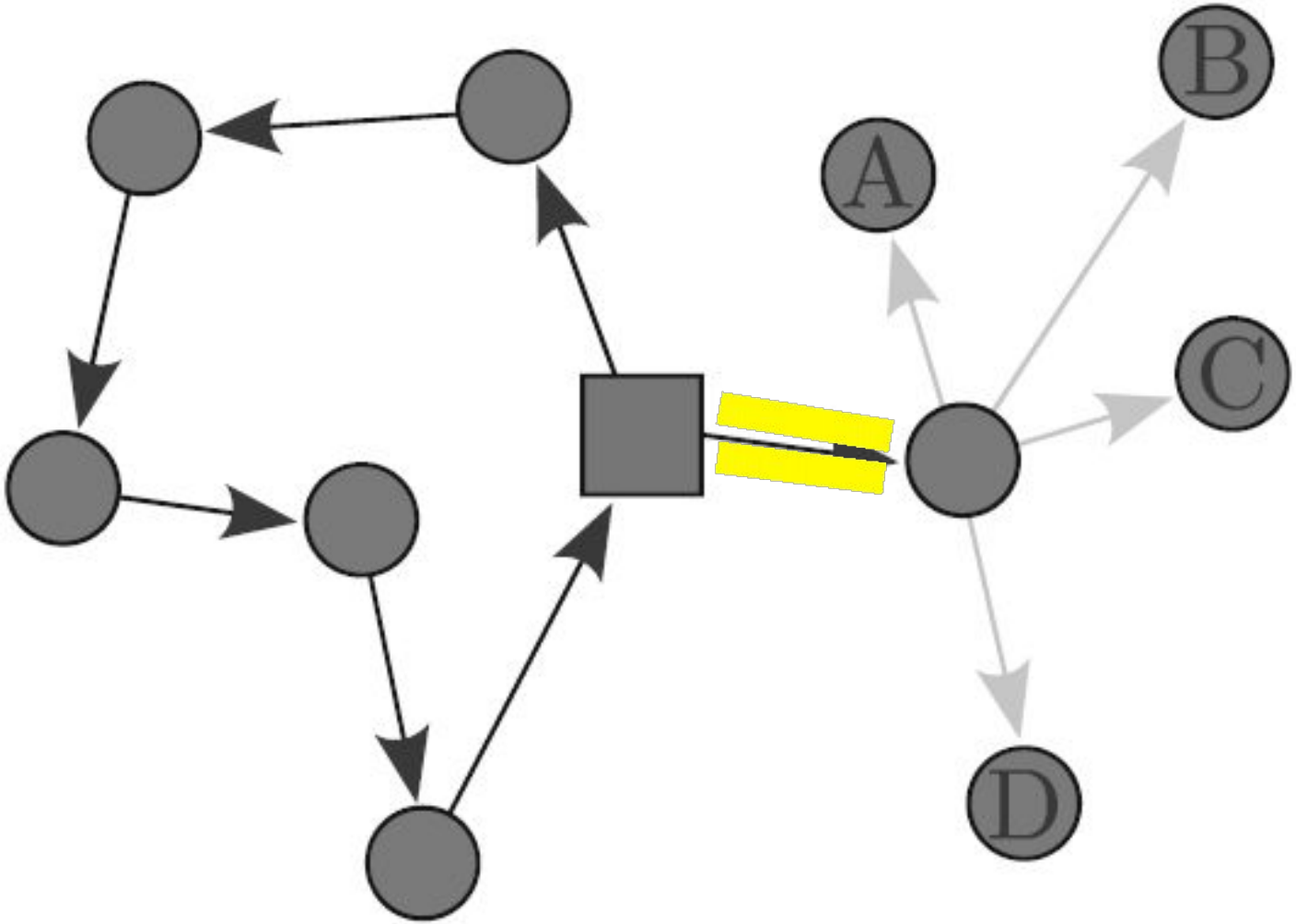
Learn to solve optimization problems with RL



Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural Combinatorial Optimization with Reinforcement Learning. No. arXiv:1611.09940, arXiv (2017)

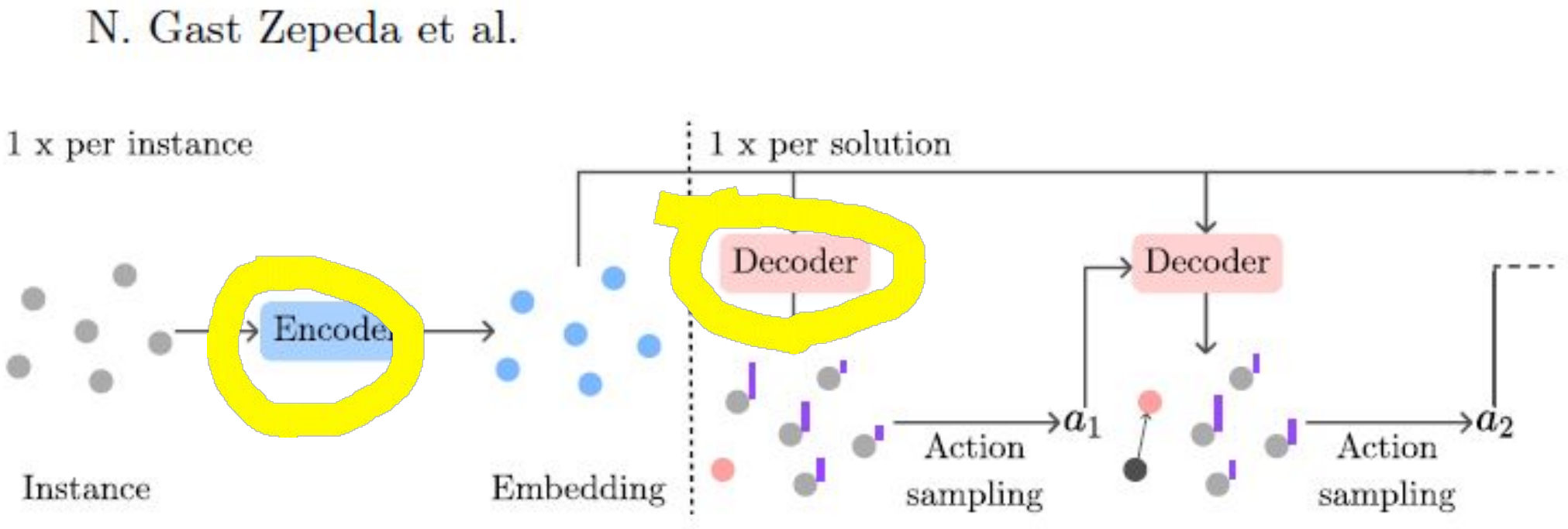
Gast Zepeda, N., Hottung, A., Tierney, K. (2026). Learning to Solve the Skill Vehicle Routing Problem with Deep Reinforcement Learning. In: Zhang, Y., Hladik, M., Moosaei, H. (eds) Learning and Intelligent Optimization. LION 2025.

Ex.: vehicle routing



Create a solution with **Sequential Decisions** (construction step by step)

DNN: Automated feature extraction



Gast Zepeda, N., Hottung, A., Tierney, K. (2026). Learning to Solve the Skill Vehicle Routing Problem with Deep Reinforcement Learning. In: Zhang, Y., Hladik, M., Moosaei, H. (eds) Learning and Intelligent Optimization. LION 2025.

LLMs produce the next word
NCO for vehicle routing:
similar architectures to produce the next customer to visit

DNN: internal representations

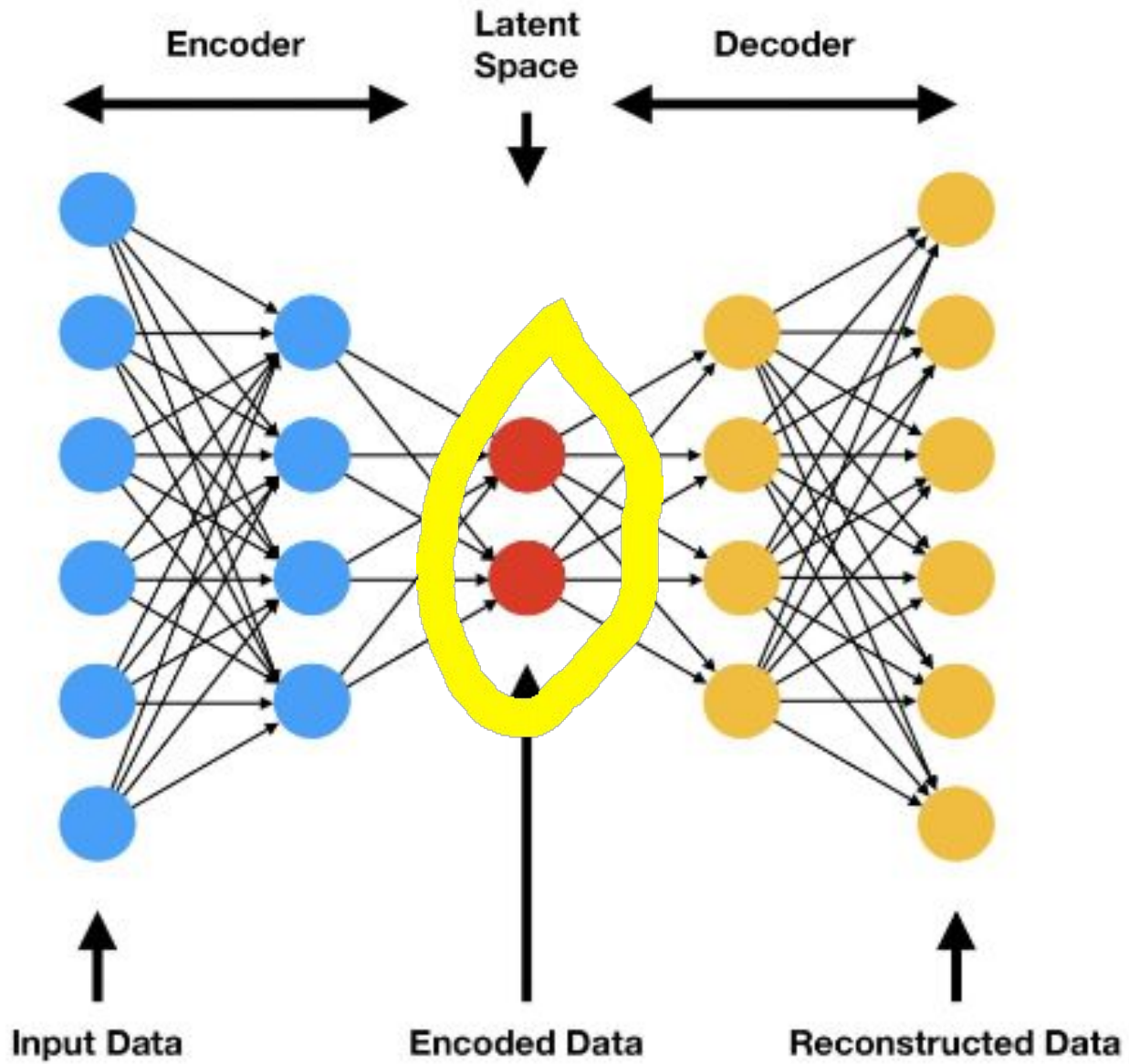


Figure 3.4: Auto-encoder.

Tricks to get a gradient

Policy is parametric (possibly a DNN)

Make a stochastic version of the problem

$$\min_{z \in Z} F(z) \quad \Longrightarrow \quad \min_{p \in \mathcal{P}_Z} \mathbb{E}_p\{F(z)\}$$

$\nabla \ln(f(x))$ is $\nabla f(x)/f(x))$

$$\begin{aligned}
\nabla\left(\mathbb{E}_{p(z;\theta)}[F(z)]\right) &= \nabla\left(\sum_{z \in Z} p(z;\theta)\, F(z)\right) \\
&= \sum_{z \in Z} \nabla p(z;\theta)\, F(z) \\
&= \sum_{z \in Z} p(z;\theta)\, \frac{\nabla p(z;\theta)}{p(z;\theta)}\, F(z) \\
&= \sum_{z \in Z} p(z;\theta)\, \nabla\left(\log(p(z;\theta))\right) F(z),
\end{aligned}$$

$$\theta^{k+1} = \theta^k - \epsilon^k \nabla_\theta\left(\log(p(z^k;\theta^k))\right) F(z^k)$$

REINFORCE for training a parametric policy

Neural Combinatorial Optimization

No stochastic parrot of existing software!

Advantages:

- **Lower barrier** to entry (Optimization becomes **learning from data**)
- **Tuned (specialized) heuristics** for specific instance distributions

Disadvantages:

- **Expensive computations** (GPU's)
- **Interpretability (and trust) lower than for handcrafted solutions**
- **Fragility** in certain cases

A very exciting area for the (near) future!

Up to this point

Goal f(x) is given (analytically or as a black box)

…

- learning local models of the fitness landscape and using them while optimizing (RSO)
- Selecting / tuning / self-tuning algorithms
- Automated heuristics discovery AHD (NCO)

… but in some cases **f(x) to optimize is only partially given**, additional info is required

Try asking a decision maker:
“give me the f(x) that you are optimizing”

Learning *what* to optimize

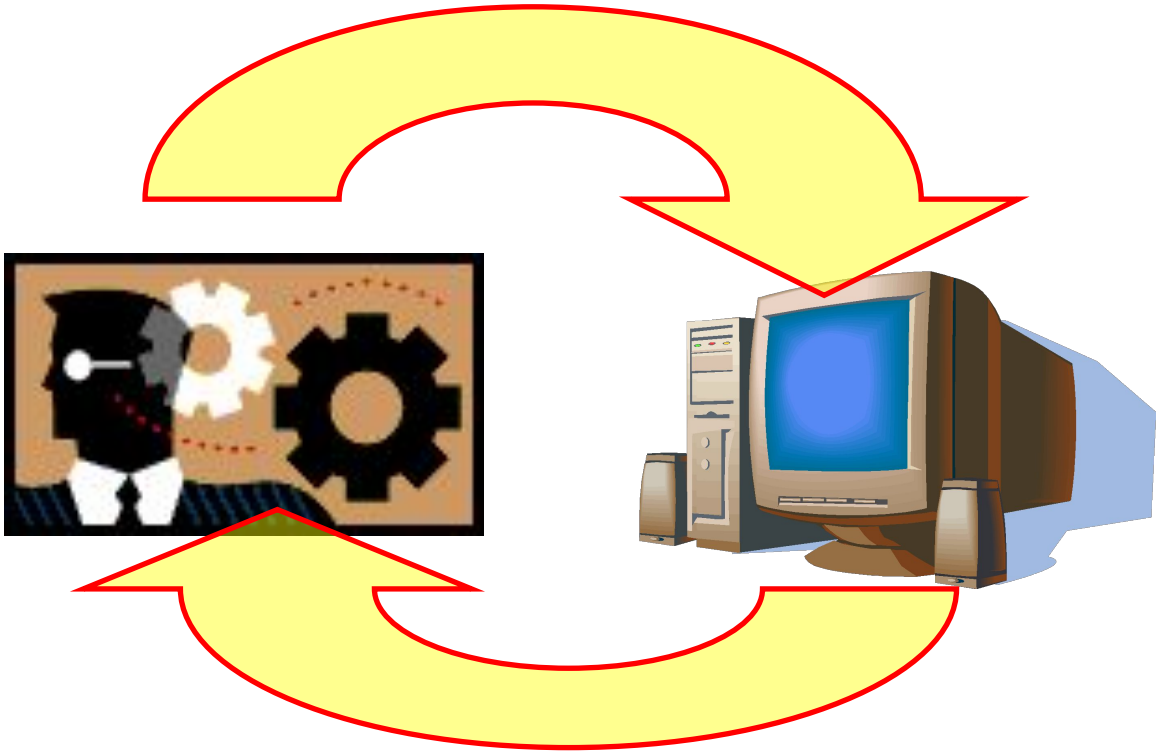
Example: MOP: Finding a partner: *intelligence* versus *beauty*

How many IQ points for one less beauty point?

Is beauty more important than intelligence for you? By how much?

(Analytic Hierarchy Process - AHP)

Effective optimization as iterative process with learning



Multiobjective optimization

intermediate (classical) case of missing knowledge:

some criteria are given **f1(x) f2(x) … fk(x)**

but not easily **combined** into a single f(x)

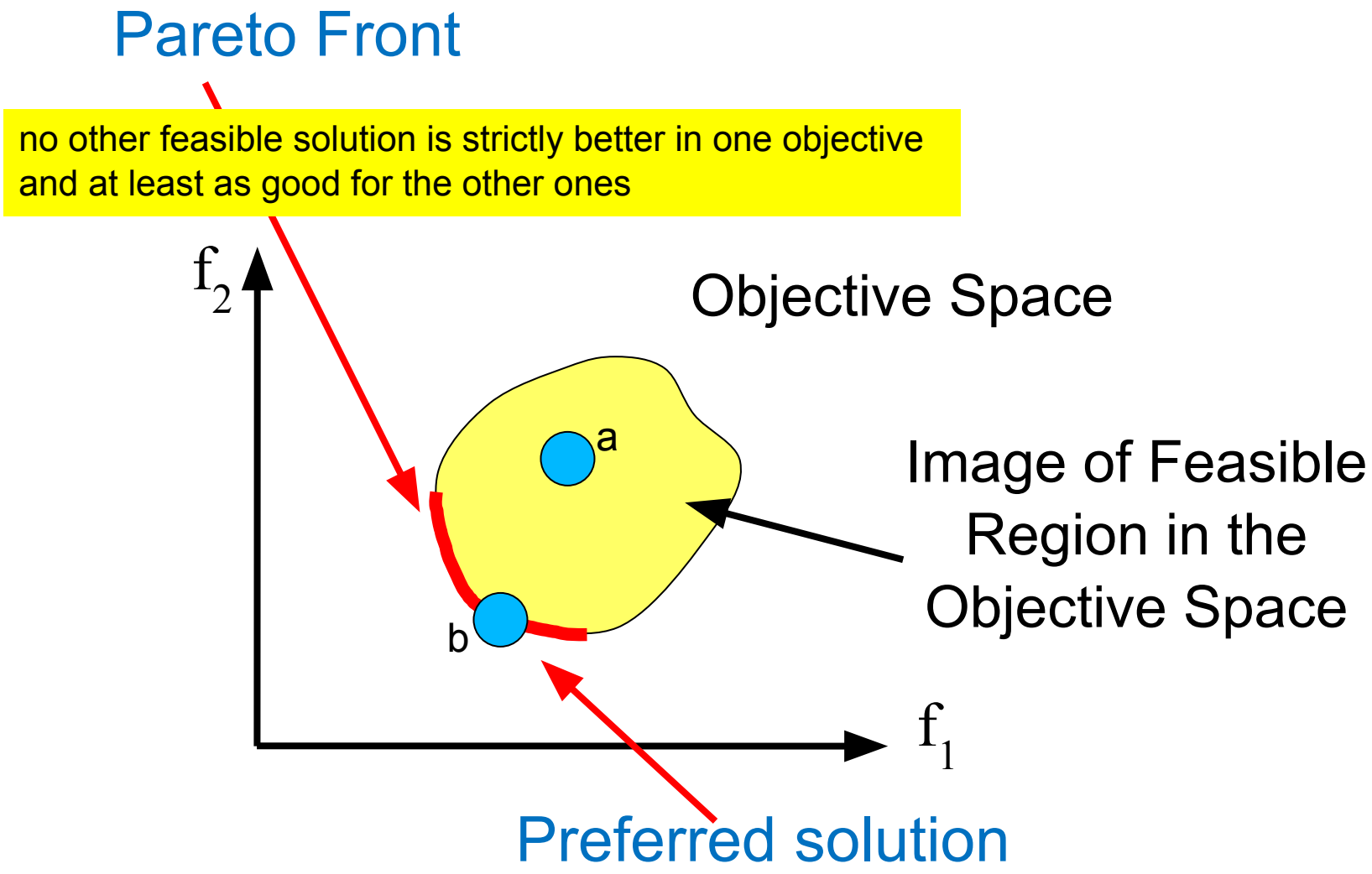
…provide efficient vector **solutions (f1,…,fk)**

leave to the user the possibility to ***decide***

(and to ***learn*** about possibilities and “real”

objectives, even if not formalized)

Efficient frontier (PF)



Preference information

- Critical task: identify the preferred solution for the DM from the efficient frontier

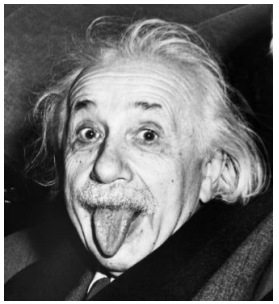
- Based on the DM preference information usage:
  - *A priori* MOO methods
  - *A posteriori* MOO methods
  - Interactive MOO methods (IM)

A priori methods

- Assumptions about preference information before optimization process

Not really MOOP

- DM specifies preference on the objectives a priori
- Drawbacks:
  - Very difficult task for DM
  - DM often does not know before how realistic his expectations are (no learning possiblities)



A posteriori methods

- The Pareto optimal set (or part of it) is generated and presented to the DM who selects the most preferred among the alternatives.
- Drawbacks:
  - Generation is computationally expensive: find all the non dominated solutions!
  - Hard for the DM to select among a large set of alternatives
  - Presenting / displaying the alternatives to the DM

Decision paralysis

Interactive methods

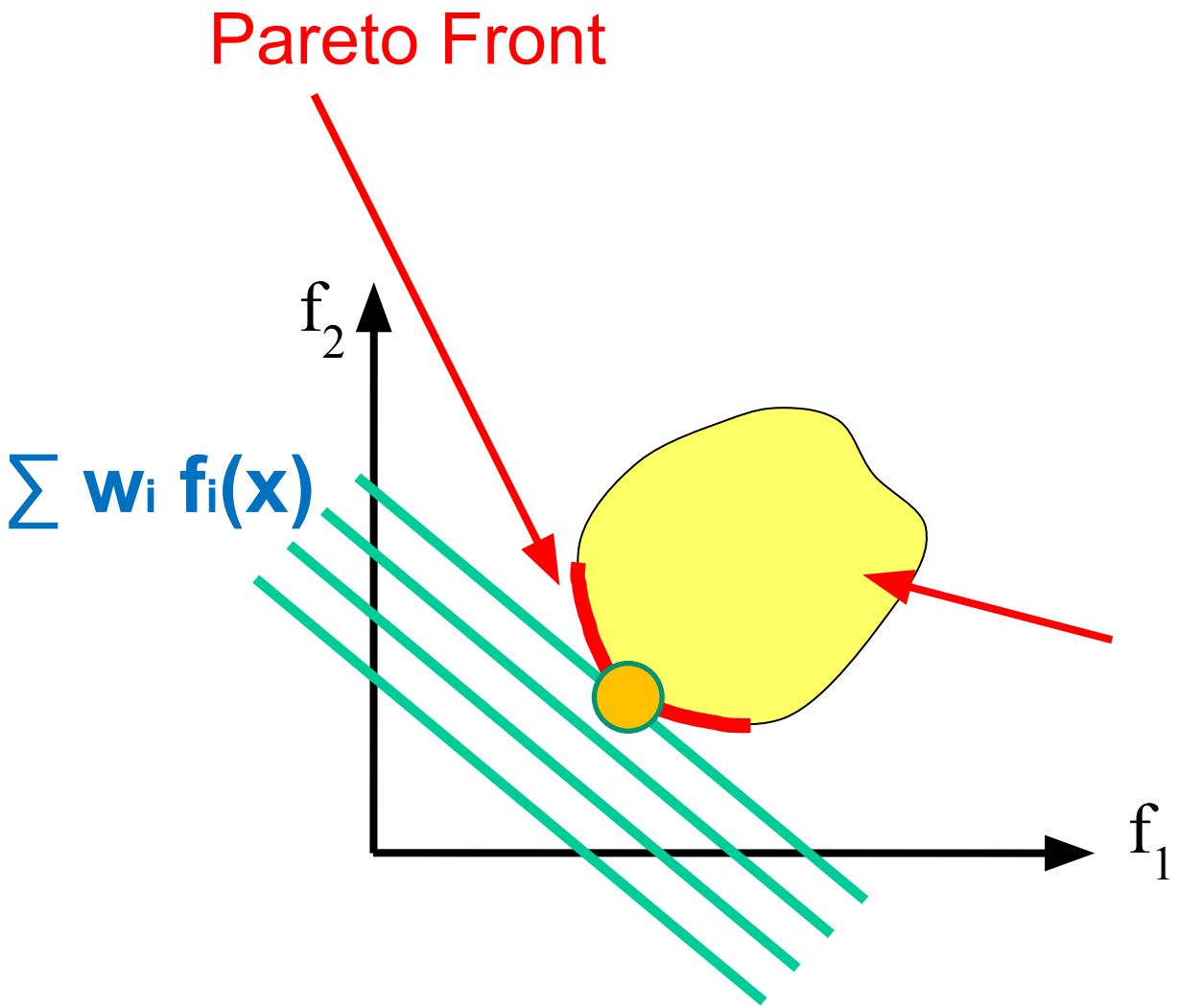
- Solutions generation phases alternated to solution evaluation phases requiring user interaction
- Effective approach
  - Only a subset of the Pareto optimal set has to be generated and evaluated by the DM
  - The DM drives the search process
  - The DM gets to know the problem better (learning on the job)

Interactive methods

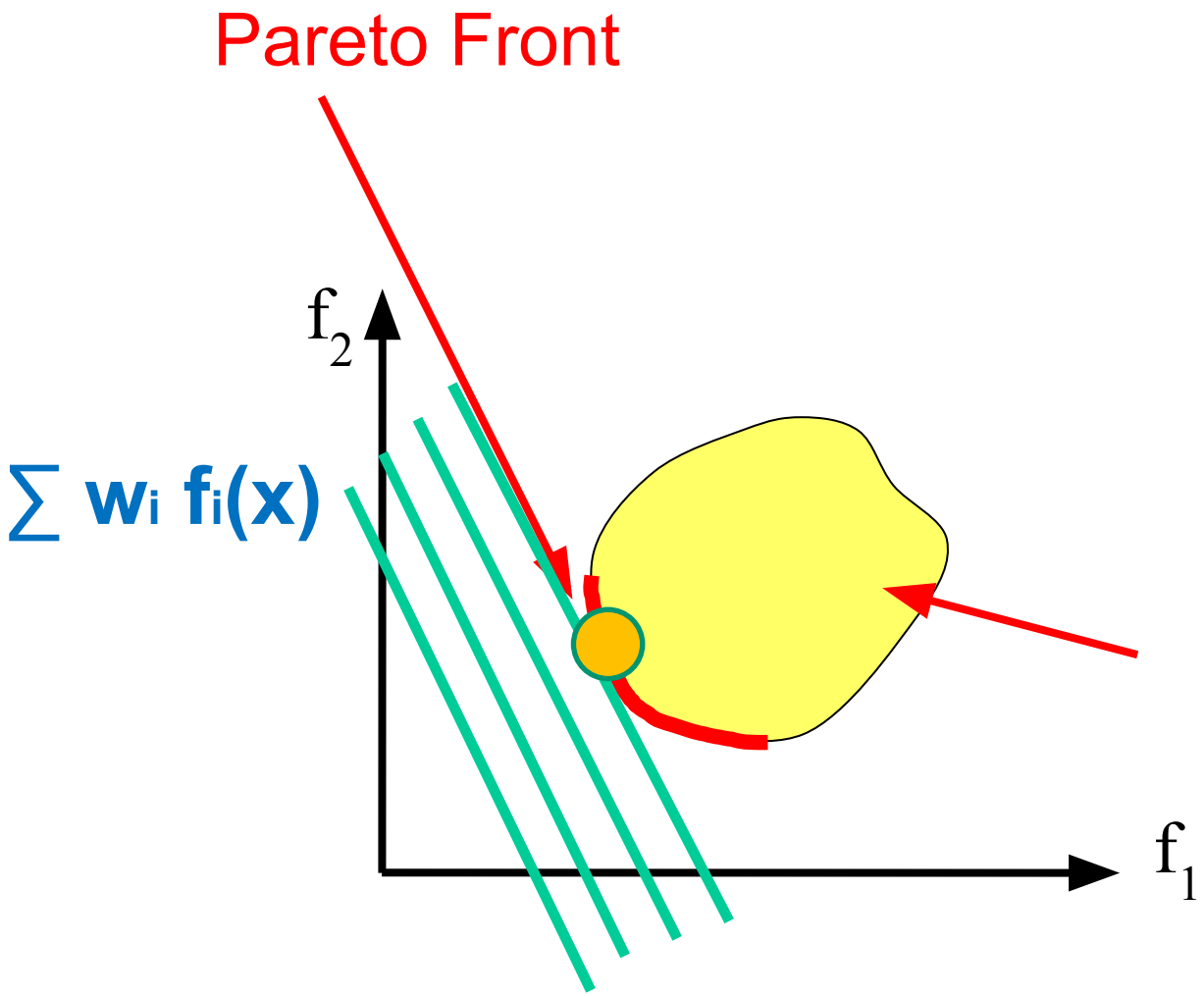
- Choices:
  - How information is provided to DM
  - How preference information is obtained from DM
  - How the search process is updated based on the preference information
  - How the original MOO problem is transformed into a single-objective optimization problem (scalarization process)

    e.g. optimize: $\sum w_i f_i(x)$

Scalarization



Scalarization



Brain-Computer EMO: learning for multiobjective optimization

Battiti, Roberto, and Andrea Passerini. "Brain–computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker." *IEEE Transactions on Evolutionary Computation* 14.5 (2010): 671-687.

Learn user preferences

Train a predictor able to reproduce **user preferences**

Use it to **guide the search** in place of the user

DM time is a scarce and costly resource It is crucial to minimize the number of queries made to the user and their complexity

Robustness for noise (inconsistencies), model flexibility

Population-based approach (EMO)

Formalizing user preferences

$$U(\mathbf{z}) = \sum_{k=1}^{m} w_k z_k.$$

Ideal objective vector

$$U(\mathbf{z}) = -\left(\sum_{k=1}^{m} (w_k |z_k^* - z_k|)^p \right)^{1/p}$$

L∞ metric

augmented weighted Tchebycheff program (AWTP):

utopian objective vector

$$\min_{\mathbf{x},\alpha} \quad \alpha + \rho \sum_{k=1}^{m} (z_k^{**} - z_k) \qquad (4)$$

Properly Pareto-optimal
Improvement possible only with **reasonable** worsening

subject to:

$$w_k(z_k^{**} - z_k) \le \alpha$$

$$\mathbf{w} \in \mathbb{R}^m,\ w_k \ge 0,\ \sum_{k=1}^{m} w_k = 1$$

$$f_k(\mathbf{x}) = z_k,\ \mathbf{x} \in \Omega$$

$$k = 1, \ldots, m$$

BC-EMO summary:

- 1) Learning an arbitrary U from feedback interactively provided by the DM
- 2) Using only DM holistic judgements
- 3) Accounting for incomplete, imprecise and contradictory feedback
- 4) Using directly the learned U to guide the search for refined solutions

Experiments (and a list of future TODO's) in:
Battiti, Roberto, and Andrea Passerini. "Brain–computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker." *IEEE Transactions on Evolutionary Computation* 14.5 (2010): 671-687.

Conclusions: Intelligent Optimization

IO is about Machine Learning for Optimization

- **Tune and self-tune** parametric (flexible) heuristics on problems (*offline*) but also on individual instances and local characteristics (*online – reactive*)
- **Create new heuristics** from scratch
- **Learn the goal f(x)** from experiments, decision makers
- More **automation** → democratization of OR, more autonomy of the final user

challenging problems still ahead!

Intelligent Optimization

Do not aim at AI – intelligence is for us,machines do not care about deep understanding/meaning

Optimize clear goal(s):

measure, understand, control, revise

"Measure what is measurable,
and make measurable what is not so“
Galileo Galilei

Artificial *General* Intelligence (AGI) is totalitarian, IO will remain a *tool* for humans

A democracy of clearly-specified specific IO *tools* better than “almighty” AI impossible to control

Thank you