Mutual Information for Feature Selection

Roberto Battiti
DISI - University of Trento, Italy

LION Laboratory (Machine Learning and Intelligent Optimization)

Information Beyond Shannon
Venice - Italy, December 29-30, 2008,

Two topics

•**Feature selection with mutual information**

•**Information in stochastic local-search-based optimization (brainstorming)**

Feature selection with mutual information

R. Battiti.
Using the mutual information for selecting features in supervised neural net learning.
***IEEE Transactions on Neural Networks*, 5(4):537--550, 1994.**

Feature selection

- Classification: input features F → class C
- Supervised learning
- Feature selection / pre-processing
  - reduce **computational cost** during training and operation
  - improve **classification accuracy** (less noise, less overtraining)
  - better **explanation** (more compact models which can be understood and explained)

Feature selection

Example: DNA microarray gene expression profiles.
Of the tens of thousands of genes in experiments, only a small number of them is related to the targeted phenotypes. For example, in a two-class cancer subtype classification problem only a few genes are often sufficient.
When a small number of genes are selected, their biological relationship with the target diseases is more easily identified.
These "marker" genes thus provide additional scientific understanding of the problem.

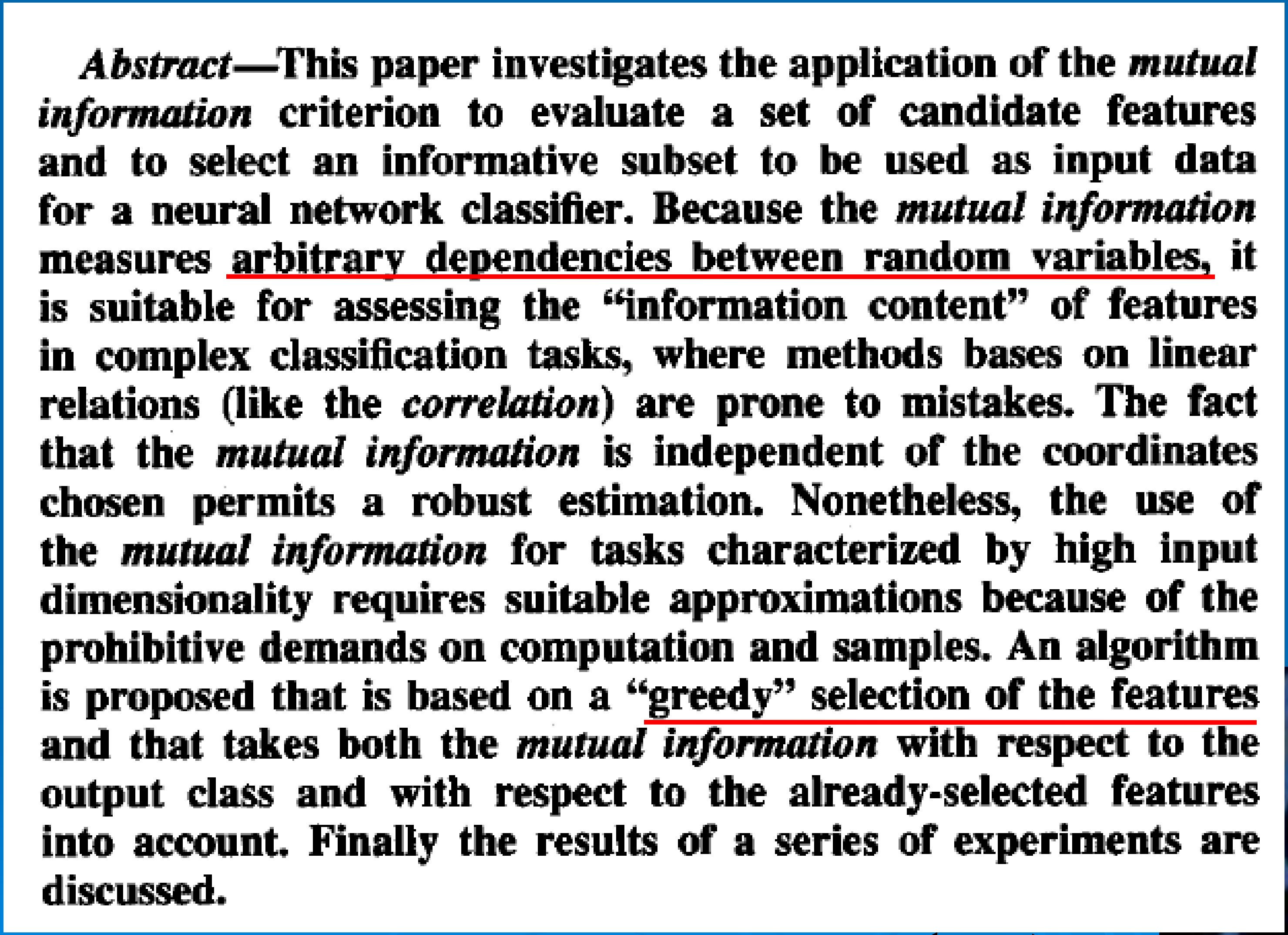
Feature selection

Two general approaches to feature selection: filters and wrappers

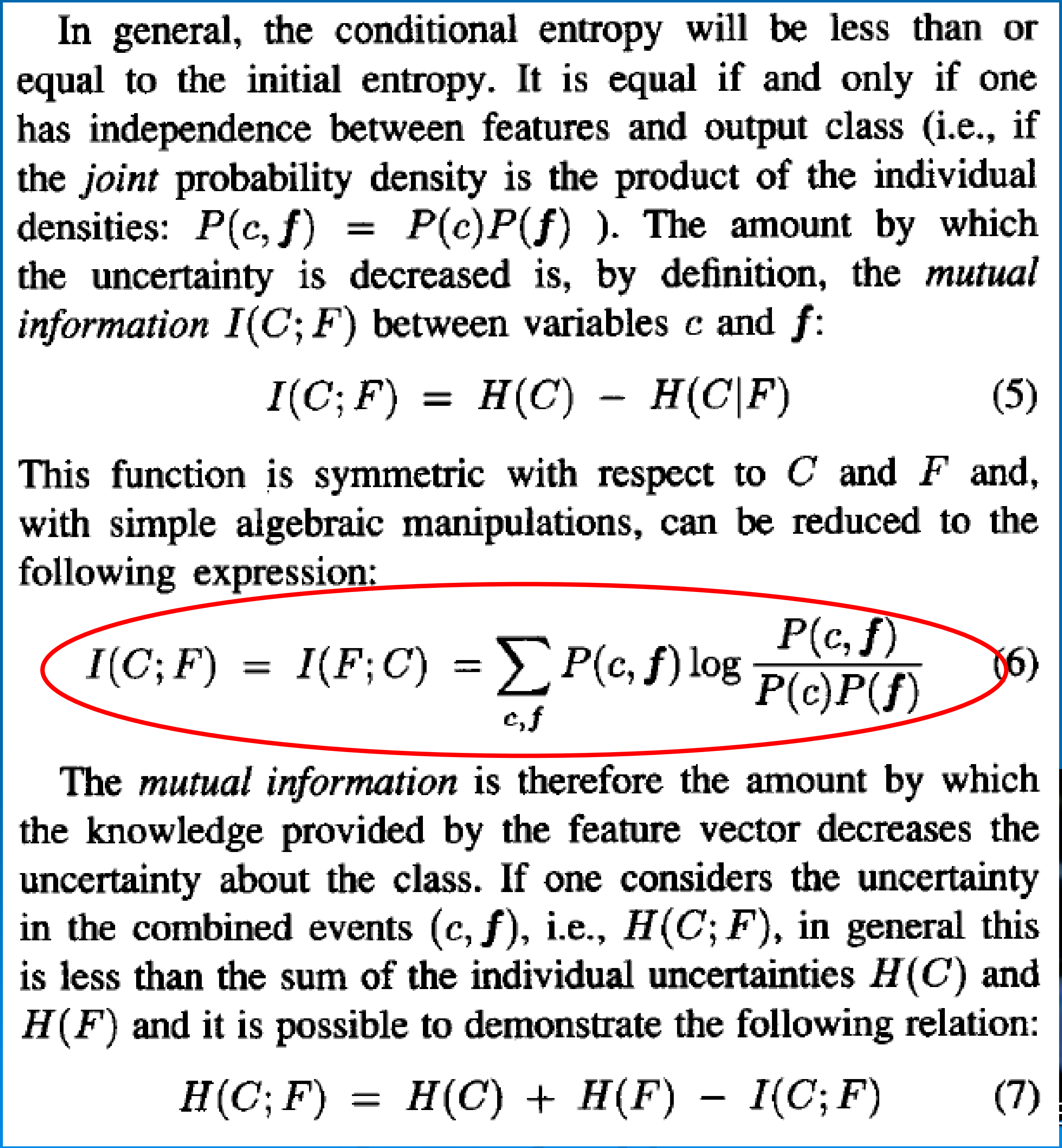
Filter type methods are essentially data pre-processing methods. Features are selected based on the intrinsic characteristics, which determine their relevance with regard to the target classes.

In wrapper type methods, feature selection is “wrapped” around a learning method: the usefulness of a feature is directly judged by the estimated accuracy of the learning method. Computationally demanding!

Feature selection



Abstract—This paper investigates the application of the *mutual information* criterion to evaluate a set of candidate features and to select an informative subset to be used as input data for a neural network classifier. Because the *mutual information* measures arbitrary dependencies between random variables, it is suitable for assessing the "information content" of features in complex classification tasks, where methods bases on linear relations (like the *correlation*) are prone to mistakes. The fact that the *mutual information* is independent of the coordinates chosen permits a robust estimation. Nonetheless, the use of the *mutual information* for tasks characterized by high input dimensionality requires suitable approximations because of the prohibitive demands on computation and samples. An algorithm is proposed that is based on a "greedy" selection of the features and that takes both the *mutual information* with respect to the output class and with respect to the already-selected features into account. Finally the results of a series of experiments are discussed.



In general, the conditional entropy will be less than or equal to the initial entropy. It is equal if and only if one has independence between features and output class (i.e., if the *joint* probability density is the product of the individual densities: $P(c,f) = P(c)P(f)$). The amount by which the uncertainty is decreased is, by definition, the *mutual information* $I(C;F)$ between variables c and f:

$$I(C;F) = H(C) - H(C|F) \tag{5}$$

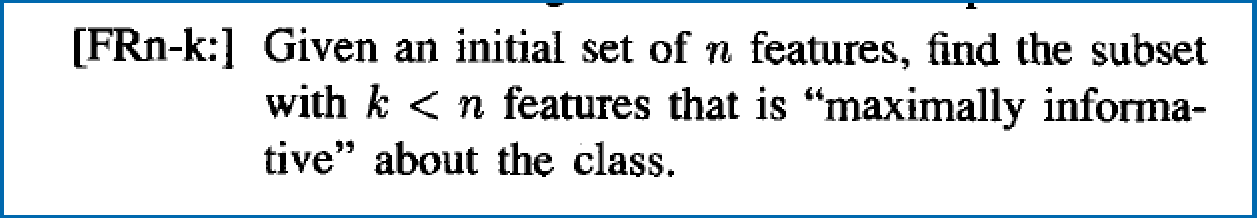
This function is symmetric with respect to C and F and, with simple algebraic manipulations, can be reduced to the following expression:

$$\boxed{I(C;F) = I(F;C) = \sum_{c,f} P(c,f) \log \frac{P(c,f)}{P(c)P(f)}} \tag{6}$$

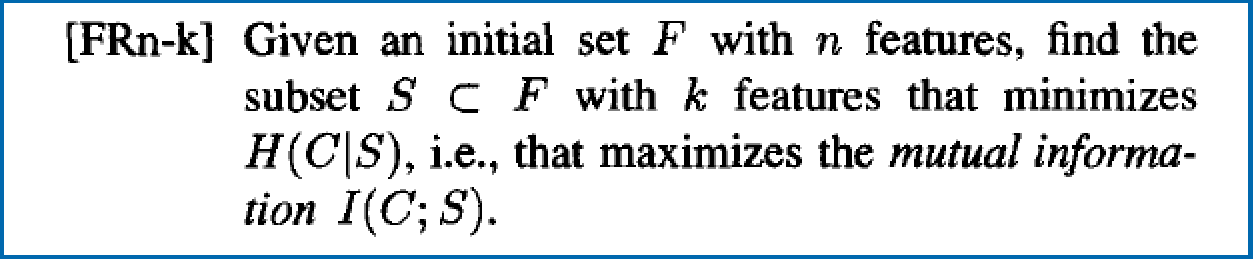
The *mutual information* is therefore the amount by which the knowledge provided by the feature vector decreases the uncertainty about the class. If one considers the uncertainty in the combined events (c,f), i.e., $H(C;F)$, in general this is less than the sum of the individual uncertainties $H(C)$ and $H(F)$ and it is possible to demonstrate the following relation:

$$H(C;F) = H(C) + H(F) - I(C;F) \tag{7}$$

Feature selection



[FRn-k:] Given an initial set of n features, find the subset with $k < n$ features that is "maximally informative" about the class.



[FRn-k] Given an initial set F with n features, find the subset $S \subset F$ with k features that minimizes $H(C|S)$, i.e., that maximizes the *mutual information* $I(C;S)$.

Obstacles:

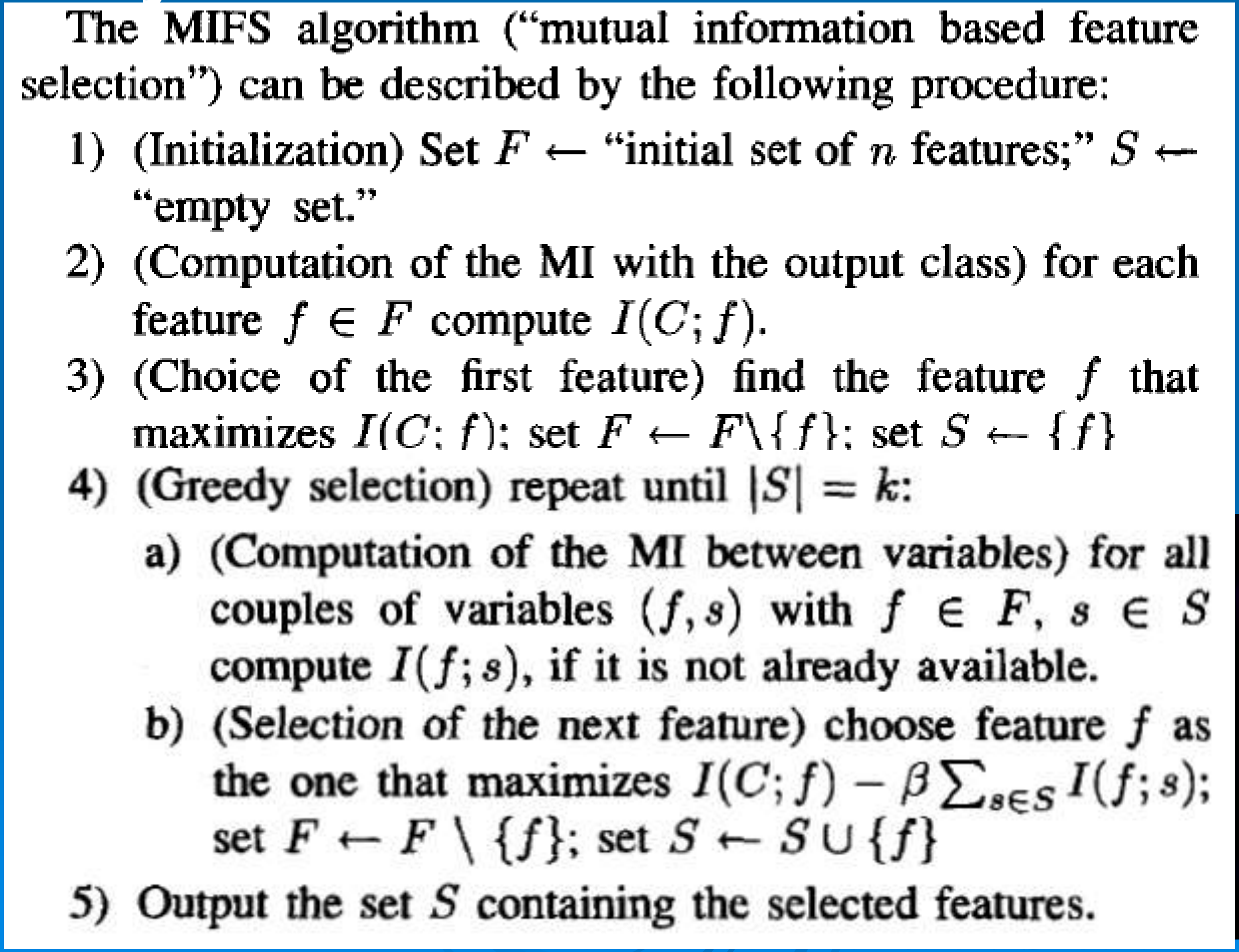
•enormous number of samples when dimension grows

•all possible subsets of size k…

Feature selection

Approximations:

1. **MI only between couples of variables**
2. **Greedy selection**

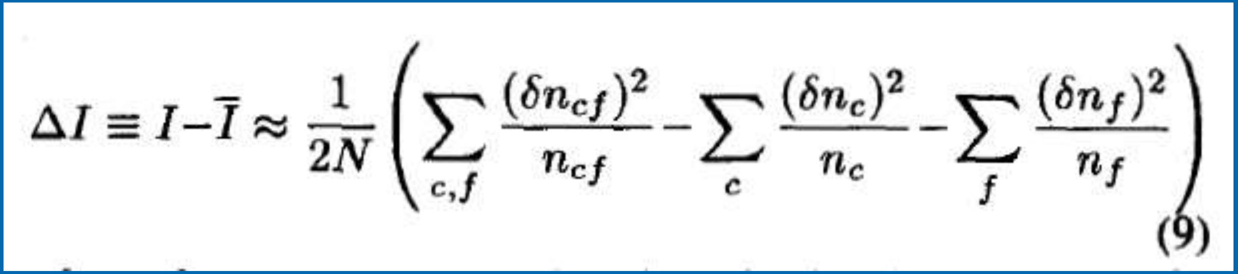


The MIFS algorithm ("mutual information based feature selection") can be described by the following procedure:

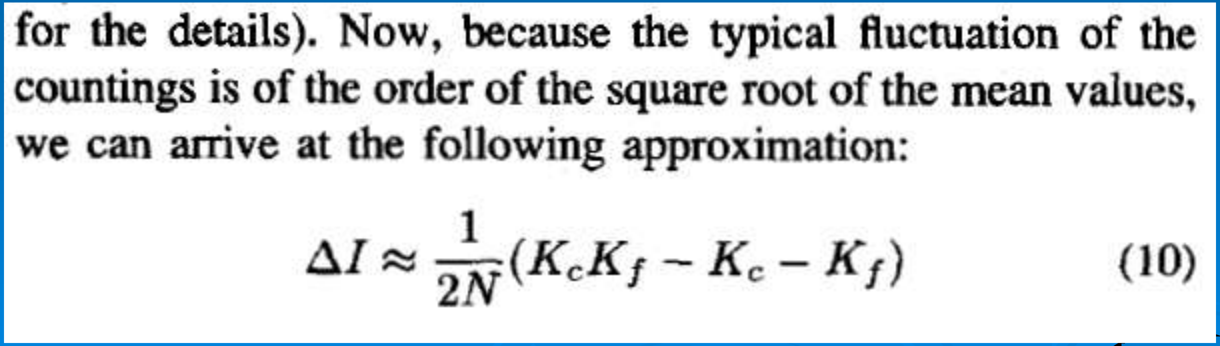
1) (Initialization) Set $F \leftarrow$ "initial set of n features;" $S \leftarrow$ "empty set."
2) (Computation of the MI with the output class) for each feature $f \in F$ compute $I(C; f)$.
3) (Choice of the first feature) find the feature f that maximizes $I(C; f)$; set $F \leftarrow F \setminus \{f\}$; set $S \leftarrow \{f\}$
4) (Greedy selection) repeat until $|S| = k$:
   a) (Computation of the MI between variables) for all couples of variables (f, s) with $f \in F$, $s \in S$ compute $I(f; s)$, if it is not already available.
   b) (Selection of the next feature) choose feature f as the one that maximizes $I(C; f) - \beta \sum_{s \in S} I(f; s)$; set $F \leftarrow F \setminus \{f\}$; set $S \leftarrow S \cup \{f\}$
5) Output the set S containing the selected features.

Feature selection

Estimating MI has difficulties:
-Effect of noise
-Adaptive discretization (Fraser)



$$\Delta I \equiv I - \bar{I} \approx \frac{1}{2N}\left(\sum_{c,f}\frac{(\delta n_{cf})^2}{n_{cf}} - \sum_{c}\frac{(\delta n_c)^2}{n_c} - \sum_{f}\frac{(\delta n_f)^2}{n_f}\right) \qquad (9)$$



for the details). Now, because the typical fluctuation of the countings is of the order of the square root of the mean values, we can arrive at the following approximation:

$$\Delta I \approx \frac{1}{2N}(K_c K_f - K_c - K_f) \qquad (10)$$

Feature selection

Fraser 1986

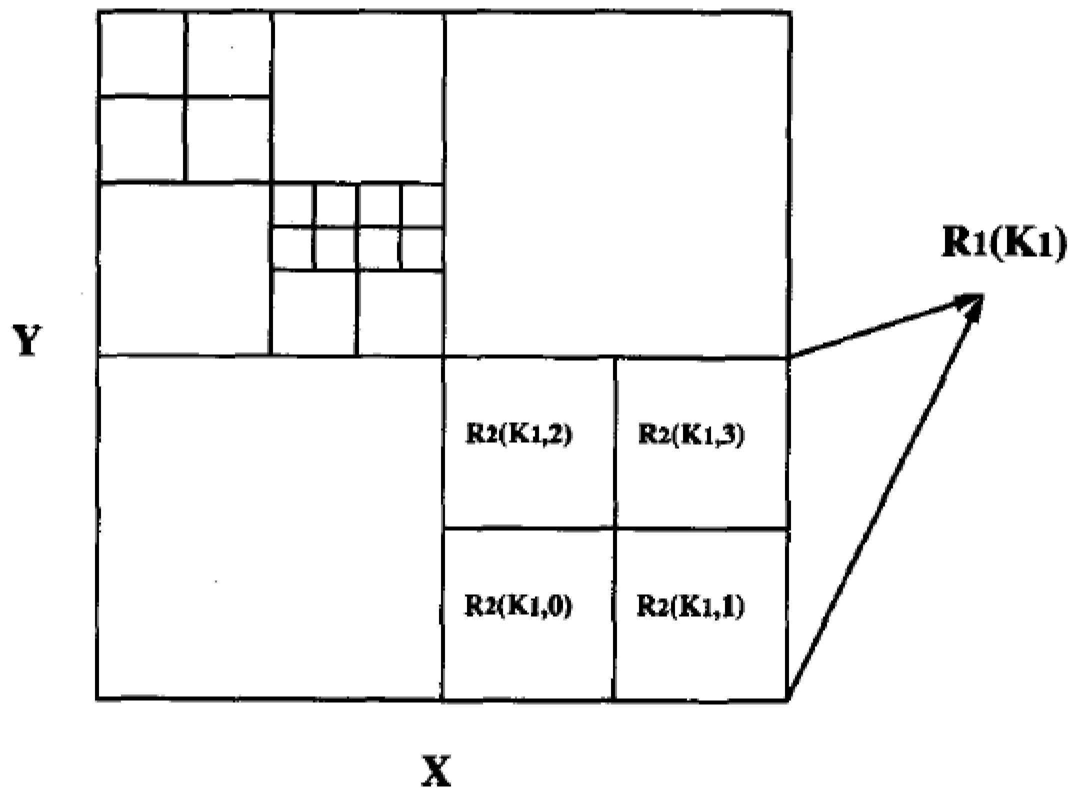
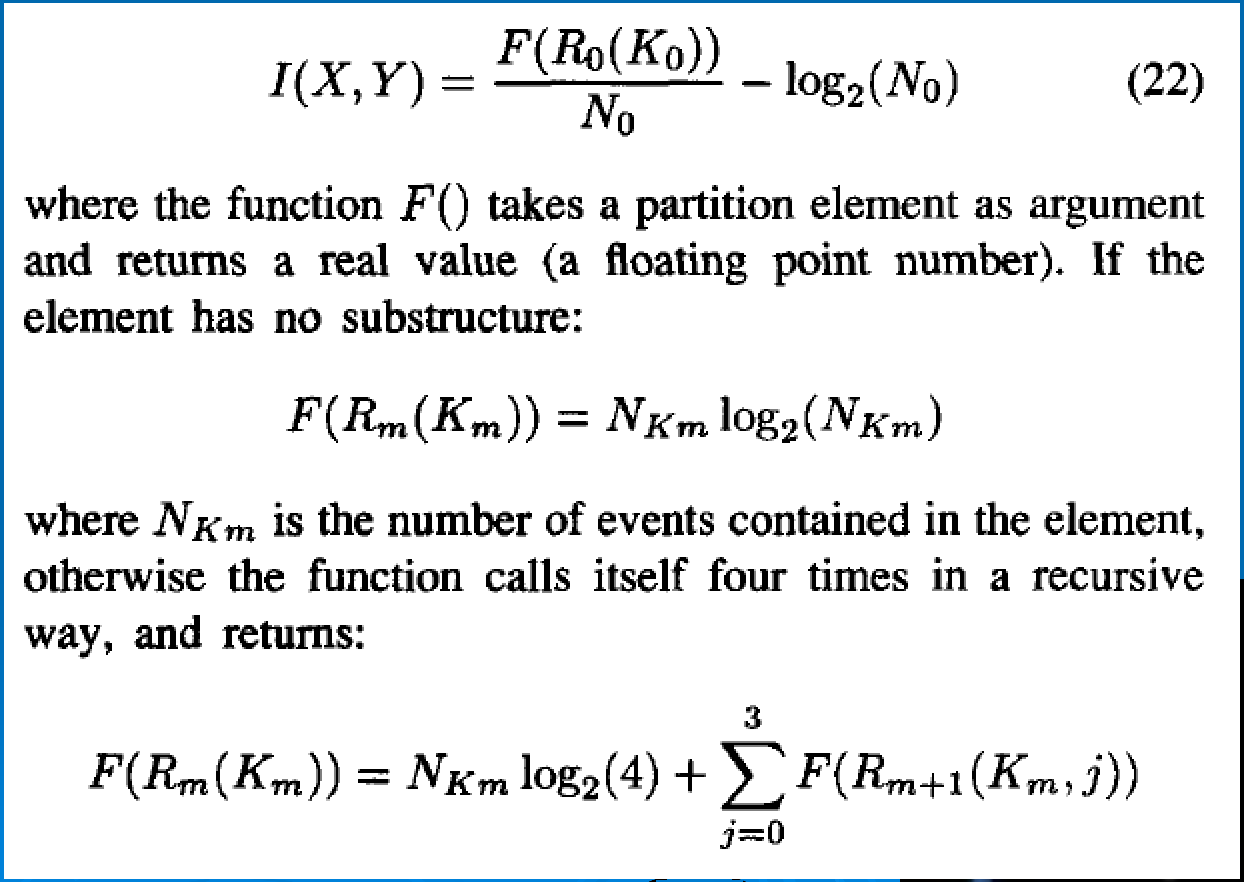


Fig. 11. Recursive partitioning of the X-Y plane executed by Fraser's algorithm. If substructure is found, an element is subdivided into four subelements.

Feature selection

Fraser 1986



$$I(X,Y) = \frac{F(R_0(K_0))}{N_0} - \log_2(N_0) \qquad (22)$$

where the function $F()$ takes a partition element as argument and returns a real value (a floating point number). If the element has no substructure:

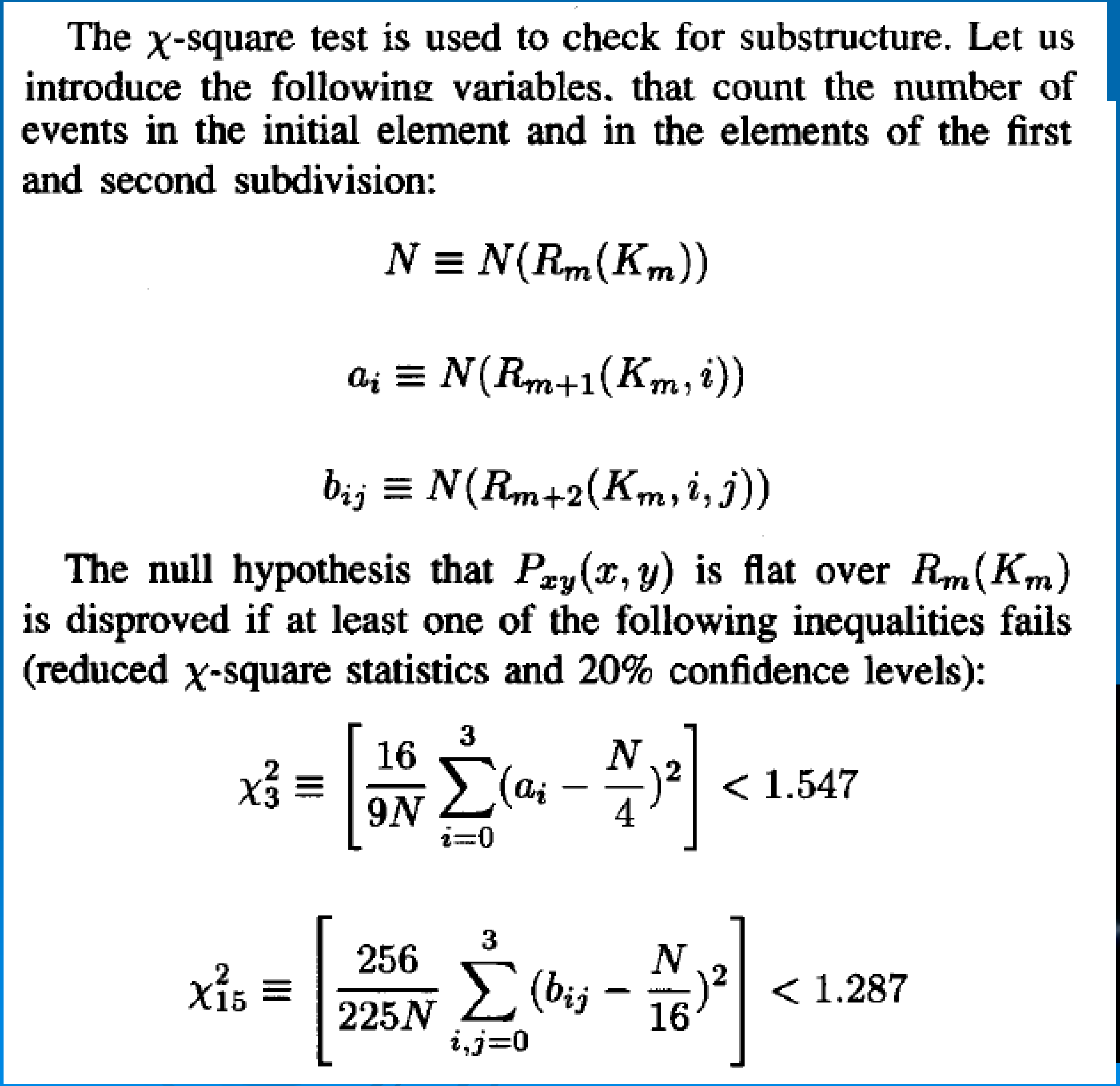
$$F(R_m(K_m)) = N_{Km} \log_2(N_{Km})$$

where N_{Km} is the number of events contained in the element, otherwise the function calls itself four times in a recursive way, and returns:

$$F(R_m(K_m)) = N_{Km} \log_2(4) + \sum_{j=0}^{3} F(R_{m+1}(K_m, j))$$

Feature selection

Fraser 1986



The χ-square test is used to check for substructure. Let us introduce the following variables, that count the number of events in the initial element and in the elements of the first and second subdivision:

$$N \equiv N(R_m(K_m))$$

$$a_i \equiv N(R_{m+1}(K_m, i))$$

$$b_{ij} \equiv N(R_{m+2}(K_m, i, j))$$

The null hypothesis that $P_{xy}(x, y)$ is flat over $R_m(K_m)$ is disproved if at least one of the following inequalities fails (reduced χ-square statistics and 20% confidence levels):

$$\chi_3^2 \equiv \left[\frac{16}{9N}\sum_{i=0}^{3}(a_i - \frac{N}{4})^2\right] < 1.547$$

$$\chi_{15}^2 \equiv \left[\frac{256}{225N}\sum_{i,j=0}^{3}(b_{ij} - \frac{N}{16})^2\right] < 1.287$$

Feature selection

Lesson learned

- **MI is valid in particular when nonlinear dependencies**

- **MI can be approximated in a crude manner while still obtaining optimal classification results**

Feature selection:
A couple of citations of biological interest

MINIMUM REDUNDANCY FEATURE SELECTION FROM MICROARRAY GENE EXPRESSION DATA
CHRIS DING *and HANCHUAN PENG*
Journal of Bioinformatics and Computational Biology
Vol. 3, No. 2 (2005) 185–205 Published: 22 September 2004

Profiled support vector machines for antisense oligonucleotide efficacy prediction
Gustavo Camps-Valls, Alistair M Chalk, Antonio J Serrano-López
José D Martín-Guerrero and Erik LL Sonnhammer
BMC Bioinformatics
Published: 22 September 2004

Feature selection: A couple of citations of biological interest

An Entropy-based gene selection method for cancer classification using microarray data
Xiaoxing Liu, Arun Krishnan and Adrian Mondry
Published: 24 March 2005
BMC Bioinformatics 2005

Reactive search: optimization with online learning

WHY…condemned to live with heuristics!

Papadimitriou and Steiglitz '82

- *6. Heuristics* Any of the < five > approaches above without a formal guarantee of performance can be considered a "heuristic." However *unsatisfactory mathematically*, such approaches are certainly valid in practical situations.

"hardness of approximation" results (in addition to NP-hardness results) of last decades: for many problems abandon the hope of finding approximation algorithms (with formal performance guarantees)!

WHY RS and Intelligent Optimization?

- **Basic optimization is a solved problem**

  More and more difficult to discover radically new techniques

- **Optimization ``in the large''**
  - Simplify life for the final user! ..automating the design process.
  - Algorithm selection, adaptation, integration, comprehensive solution.
  - Diversity, stochasticity, dynamicity
  - Interaction with final user (HCI)
  - Relationships between problem definition and problem solution (learning the definition!)

Increase in automation → final user wins…
… but more challenging work for the researcher… no off-the-shelf hyper-heuristics

Reactive search

=

ON-LINE MACHINE LEARNING FOR OPTIMIZATION

=

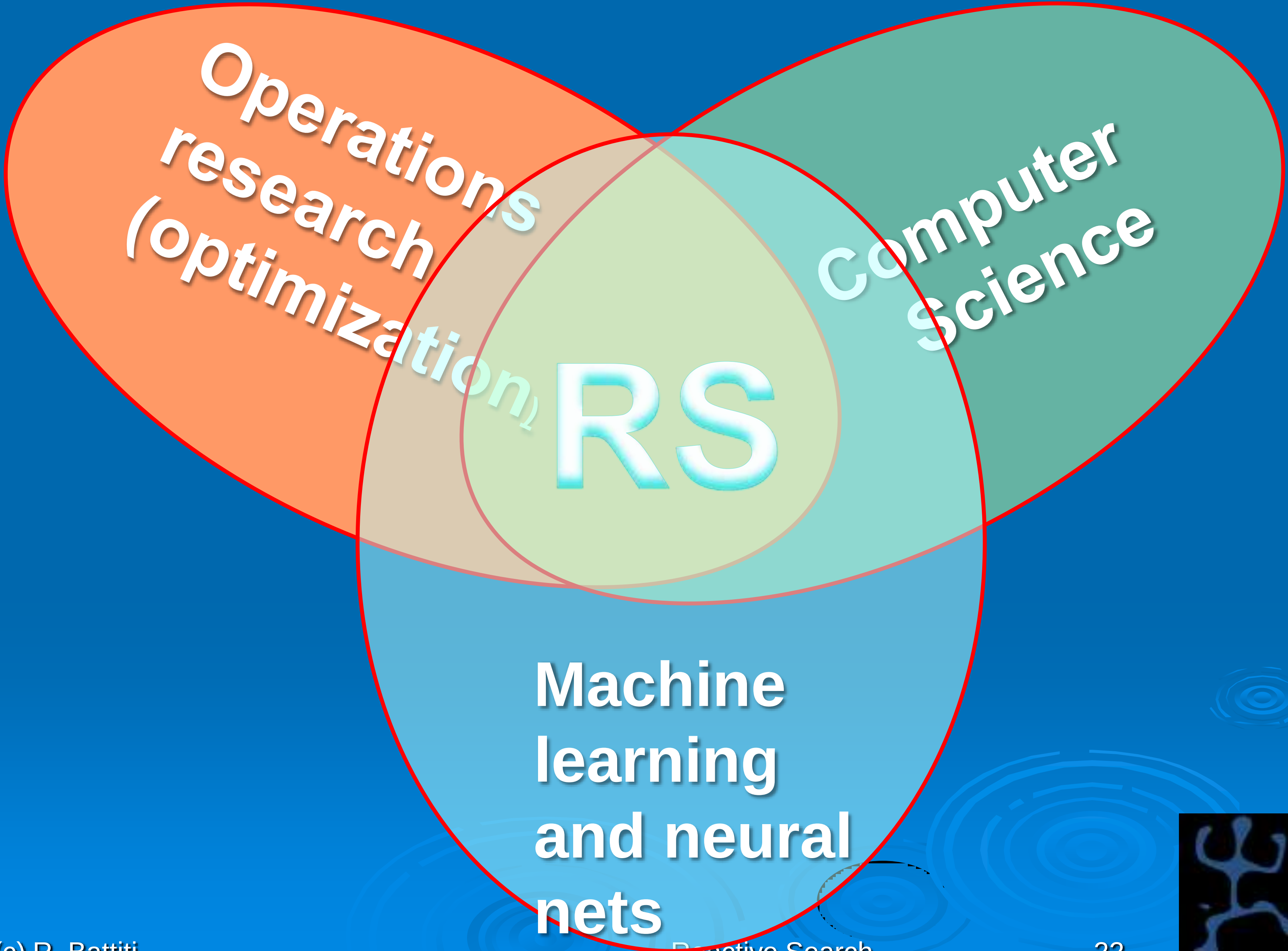
on-line dynamic and adaptive search

=

on-line reinforcement learning for optimization

Reactive Search: Learning on the Job

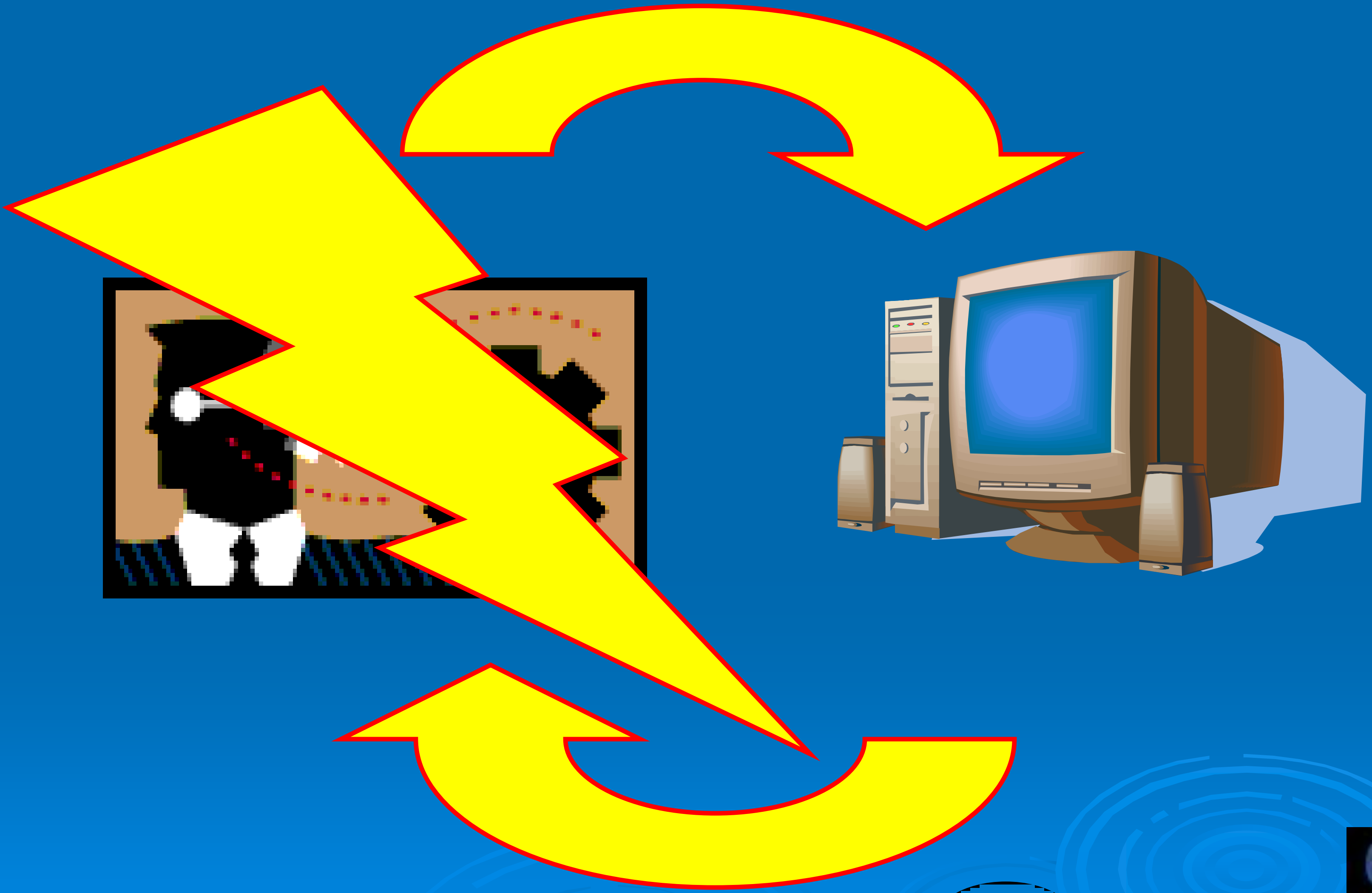
Intelligent optimization includes also learning in different contexts: off-line tuning, swarm intelligence, genetic algorithms and natural adaptation, tabu search, …


Operations research (optimization)
Computer Science
RS
Machine learning and neural nets

Who invented reactive search ?

Iceman - Ötzi
"L'uomo venuto dal ghiaccio"
15 Gennaio - 18 Giugno 2006
Dalle Alpi di 5.300 anni fa,
la più antica mummia del mondo

The role of the user



The role of the user

- **choices and free parameters**

  Algorithm(T)

- **the user as a crucial learning component** ("trial and error")

- **Parameter tuning is a typical "learning" process** where experiments are designed, with the support of statistical estimation (parameter identification) tools.

Automated tuning through machine learning

- **Automation**. The time-consuming tuning phase is now substituted by an automated process.
- **Complete and unambiguous documentation**. The algorithm becomes self-contained: its quality can be judged independently from the designer.

Complexity is shifted
Final user → algorithm developer

On-line tuning

- Take into account:
  - Problem-dependent
  - Task-dependent
  - Local properties in configuration space (see local search), parameters are dynamically tuned based on optimization state and previous history

Reactive search is about

- integration of sub-symbolic machine learning techniques into 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.

Methodologies of interest for Reactive Search include machine learning and statistics, in particular reinforcement learning, active or query learning, transfer learning, neural networks

Different from Markov process

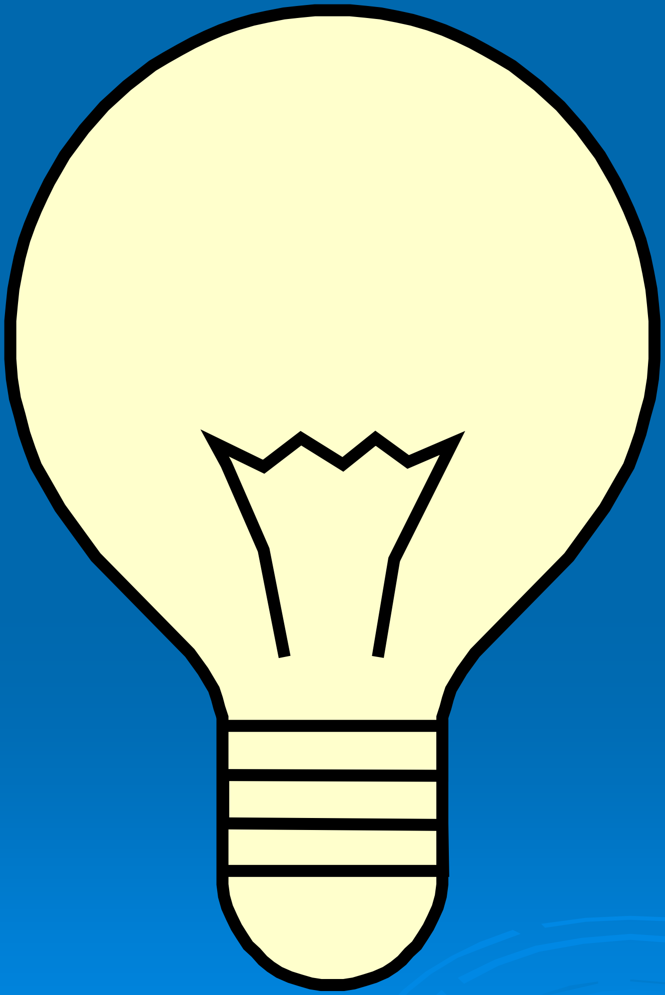
$$Y \leftarrow \textsc{Neighbor}(\, N(X^{(t)})\,)$$

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

Simulated Annealing

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

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



Reactive prohibitions

- beyond local optimality: use prohibitions for diversification

It is a good morning exercise for a research scientist to **discard a pet hypothesis** every day before breakfast. It keeps him young.
Konrad Lorenz

- Prohibition-based: history
  - 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
  - Tabu Search
  - Steepest Ascent Mildest Descent

An example: prohibition-based local search

- ***X*** is the search space **0010110001000**
- Neighborhood

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

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

$$Y \leftarrow \textsc{Best-Neighbor}(\, N(X^{(t)}) \,)$$

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

Prohibition-based local search

- Local search leads to local minima
- What next?
  - (random) restart
  - try to use knowledge accumulated during the previous searches *(learn !)*
  - … escape from previously visited basins of attraction around a local minimizer (diversification)
  - simple diversification through prohibitions

Prohibition-based local search (3)

- diversification through prohibitions

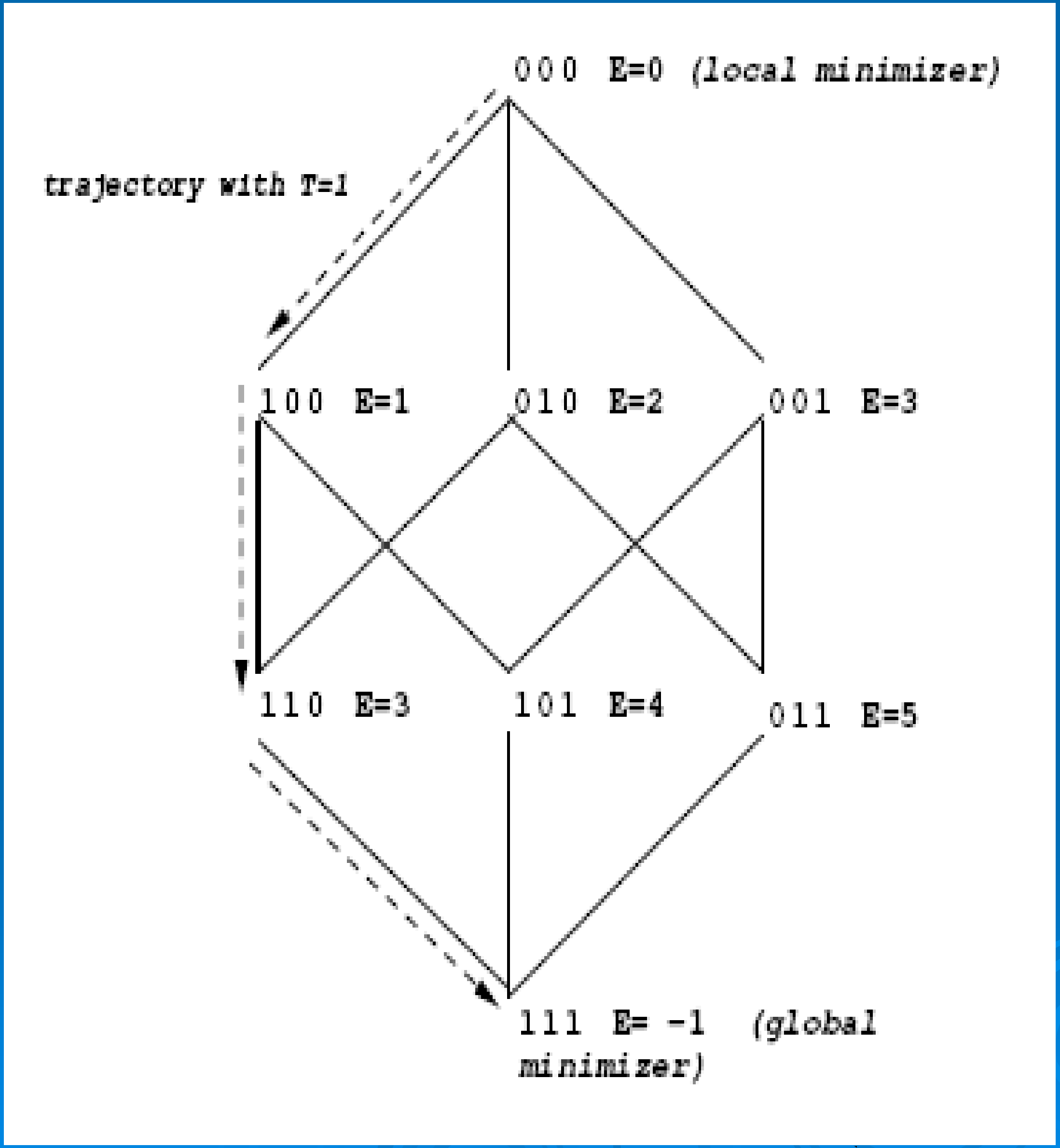
0010110001000

0010010001000 H=1

0010010011000 H=2

- Binary strings: if one prohibits changing again a bit for T iterations, Hamming distance has to increase up to T+1

T=1 example


000 E=0 (local minimizer)
trajectory with T=1
100 E=1
010 E=2
001 E=3
110 E=3
101 E=4
011 E=5
111 E= -1 (global minimizer)

Prohibition and diversification

Basic relationships

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

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

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

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

Some forms of Tabu Search

- Allowed neighbors

$$N_A(X^{(t)}) \subseteq N(X^{(t)})$$

- Discrete dynamical system

$$\begin{aligned} X^{(t+1)} &= \textsc{Best-Neighbor}\left(N_A(X^{(t)})\right) \\ N_A(X^{(t+1)}) &= \textsc{Allow}\left(N(X^{(t+1)}), X^{(0)}, \ldots, X^{(t+1)}\right) \end{aligned}$$

Tabu Search: Prohibition Mechanisms

- Strict-TS

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

- Fixed-TS

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

- Reactive-TS

? Are the dynamical systems comparable ?

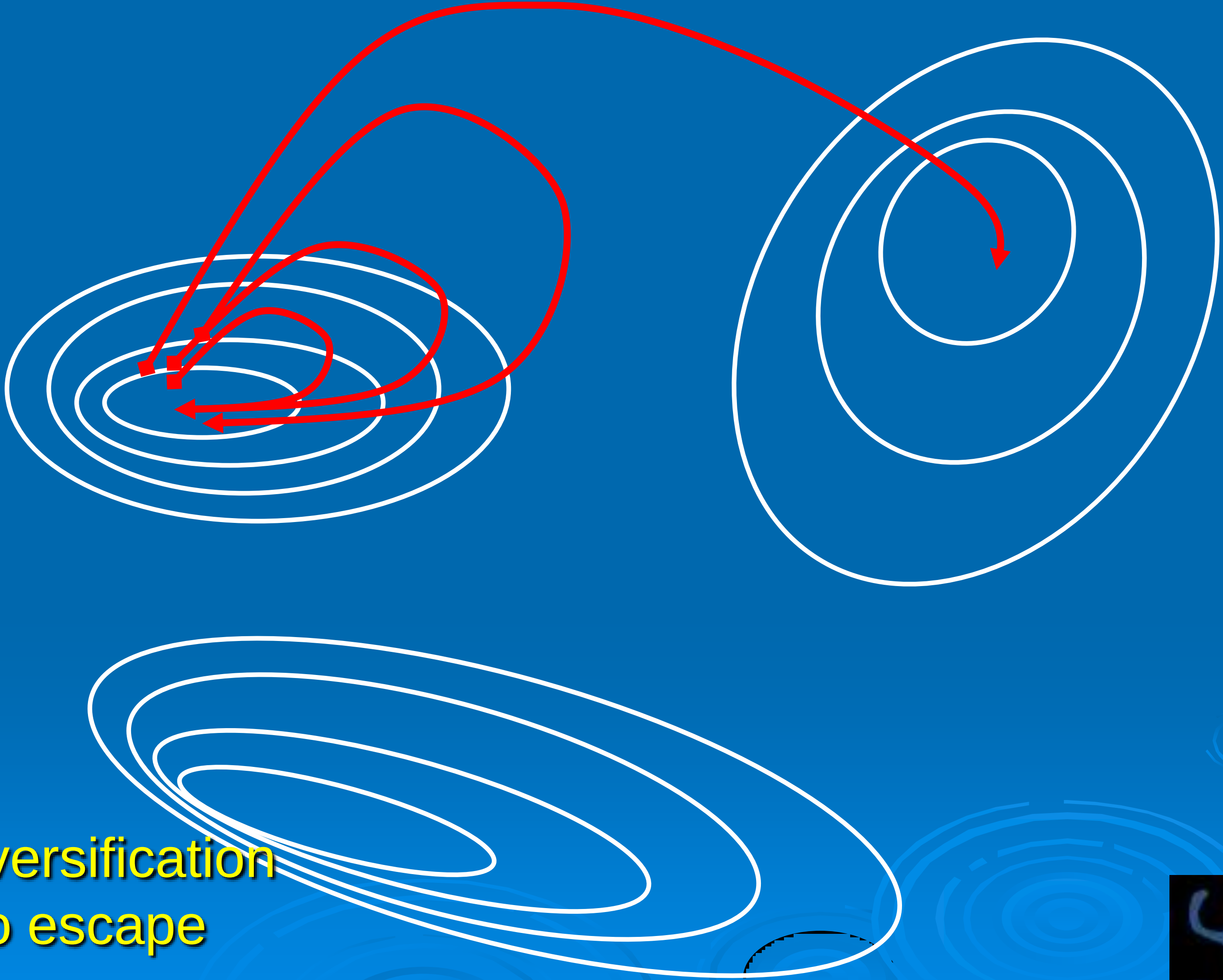
? Or qualitative differences ?

Distinguish policies from mechanisms

Issues in prohibition-based search

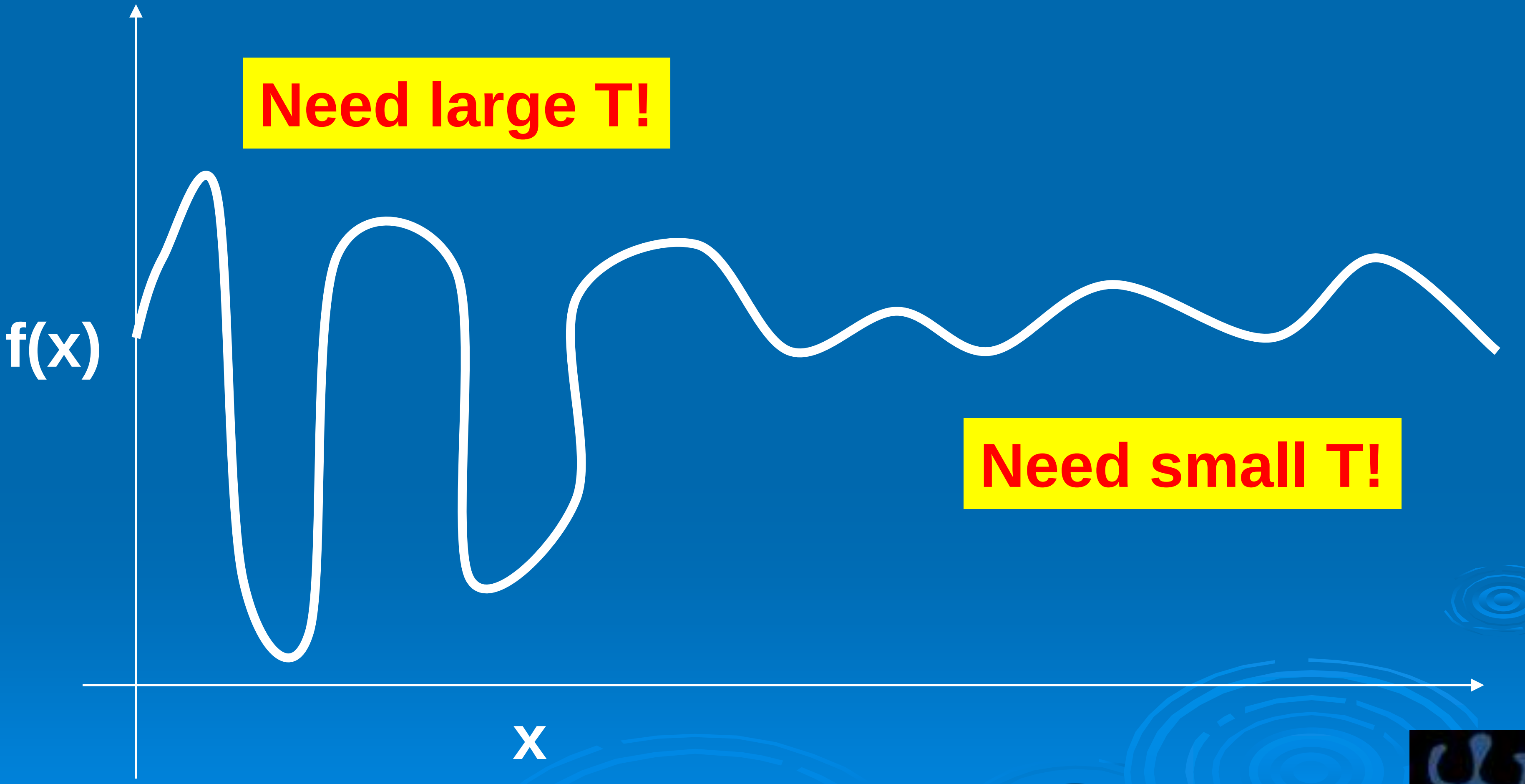
- Tuning of T (offline vs. reactive/online)
- Appropriate data structures for storing and accessing search history
- Robustness for a variety of applications

Reactive Prohibition-based search



Minimal diversification sufficient to escape

Motivations for a dynamic T

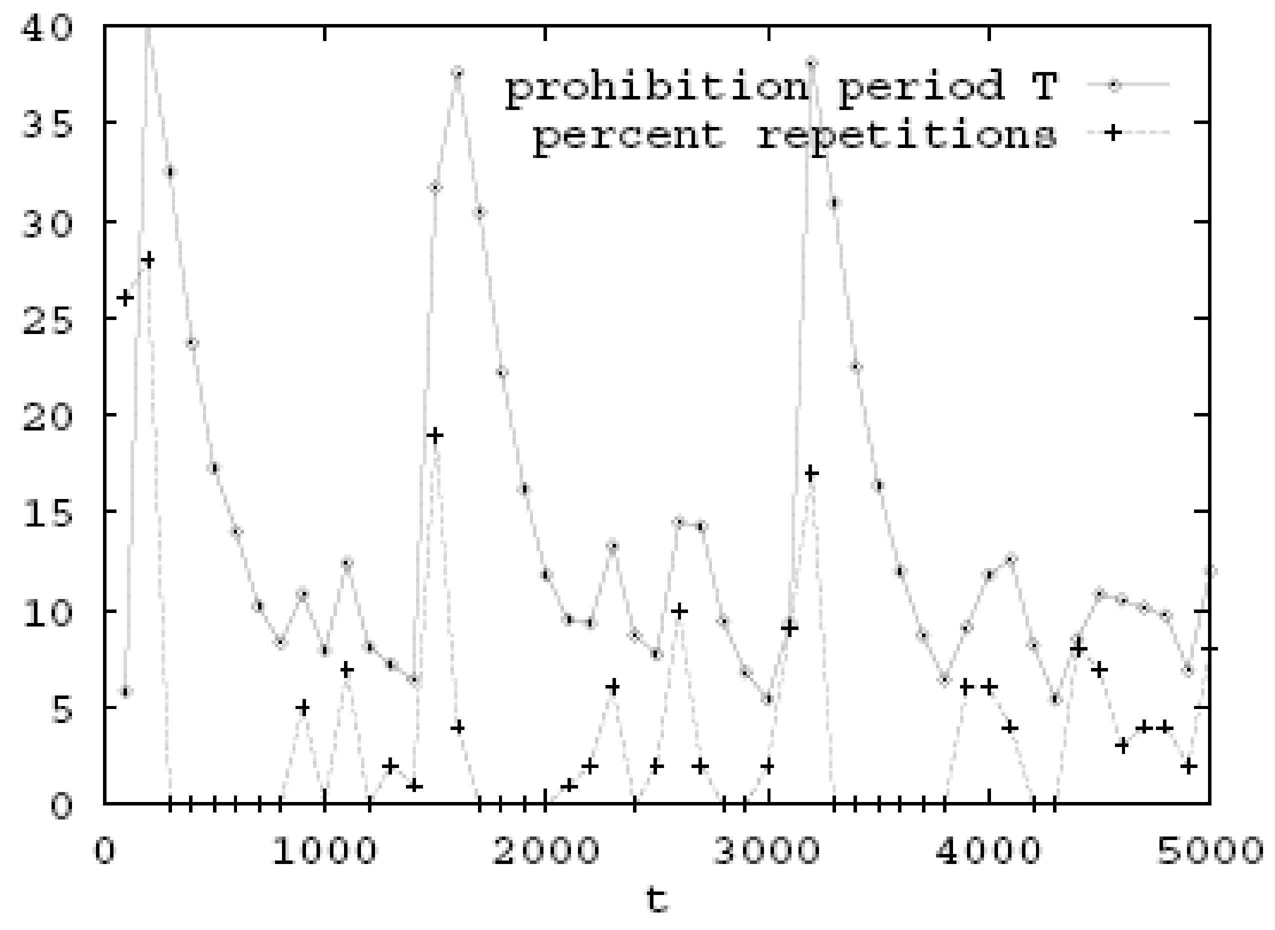


Self-adjusted T

- T=1 at the beginning
- Increase T if evidence of *entrapment*
  - **$T \leftarrow T\ \ 1.1$**
- Decrease T if evidence disappears
  - **$T \leftarrow T\ 0.9$**

- Details do not matter provided average value is appropriate

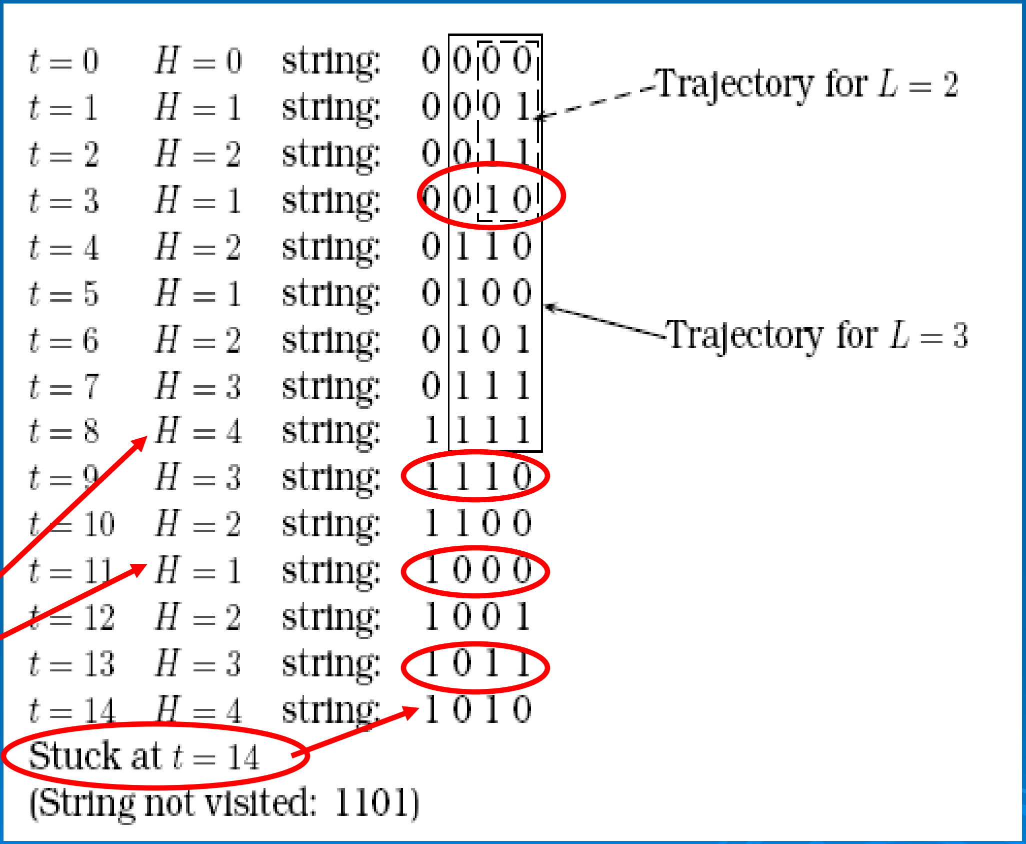
Self-adjusted T (2)



How to escape from an attractor

- Cost= Hamming distance from 00000
- Strict-TS

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



$t = 0$	$H = 0$	string:	0 0 0 0
$t = 1$	$H = 1$	string:	0 0 0 1
$t = 2$	$H = 2$	string:	0 0 1 1
$t = 3$	$H = 1$	string:	0 0 1 0
$t = 4$	$H = 2$	string:	0 1 1 0
$t = 5$	$H = 1$	string:	0 1 0 0
$t = 6$	$H = 2$	string:	0 1 0 1
$t = 7$	$H = 3$	string:	0 1 1 1
$t = 8$	$H = 4$	string:	1 1 1 1
$t = 9$	$H = 3$	string:	1 1 1 0
$t = 10$	$H = 2$	string:	1 1 0 0
$t = 11$	$H = 1$	string:	1 0 0 0
$t = 12$	$H = 2$	string:	1 0 0 1
$t = 13$	$H = 3$	string:	1 0 1 1
$t = 14$	$H = 4$	string:	1 0 1 0

Trajectory for $L = 2$

Trajectory for $L = 3$

Stuck at $t = 14$

(String not visited: 1101)

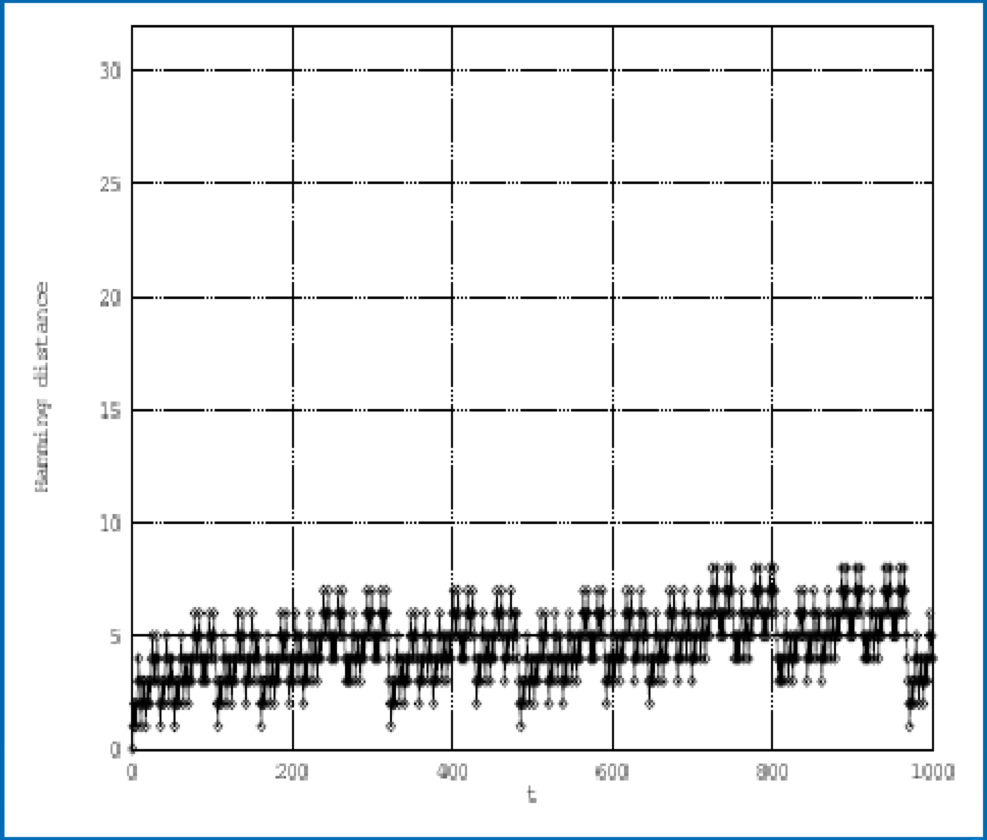
Non so intensifying…

How to escape from an attractor

- Strict-TS

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

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

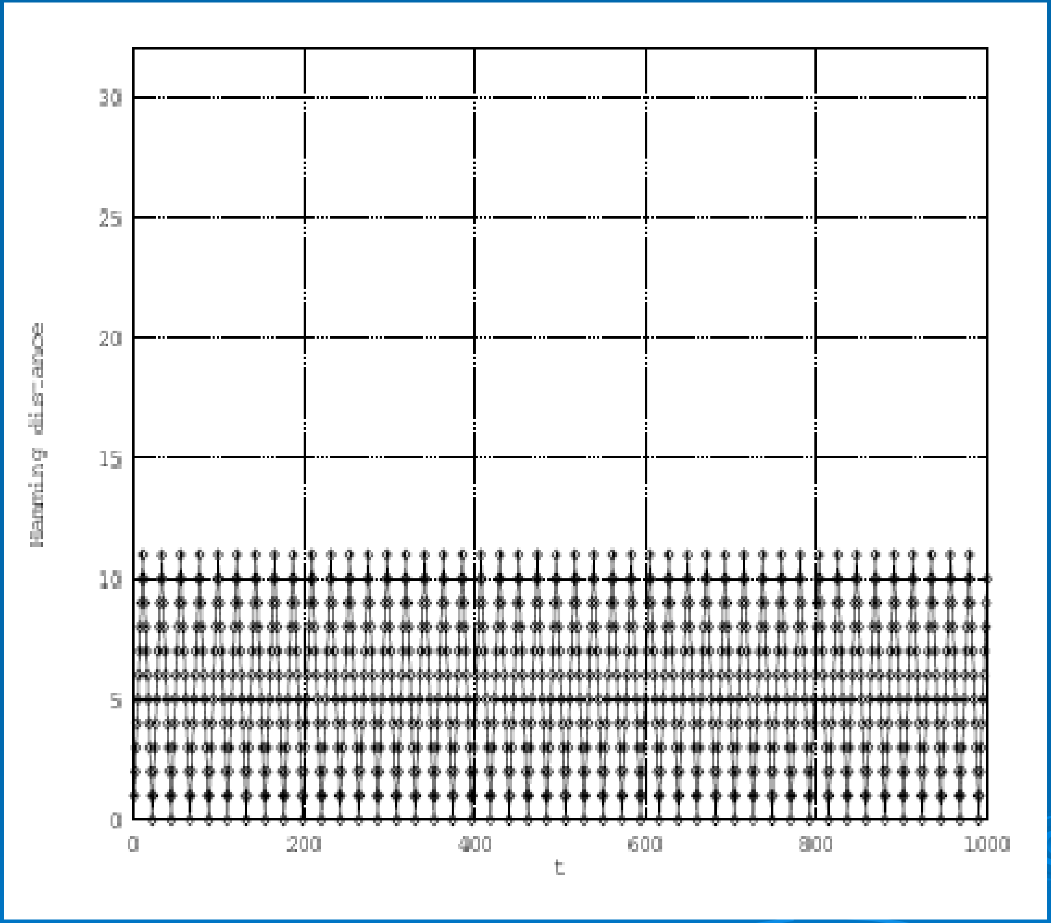


- Curse of dimensionality, “basin filling”

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

How to escape from an attractor

- Fixed-TS

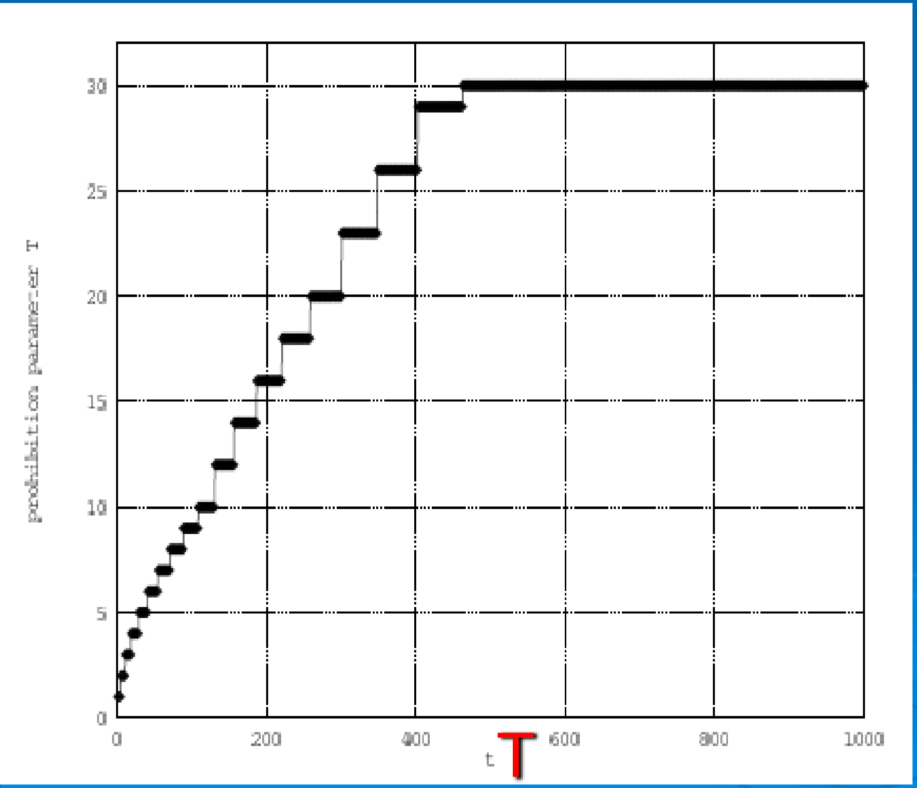


?Sufficient to escape or not ?

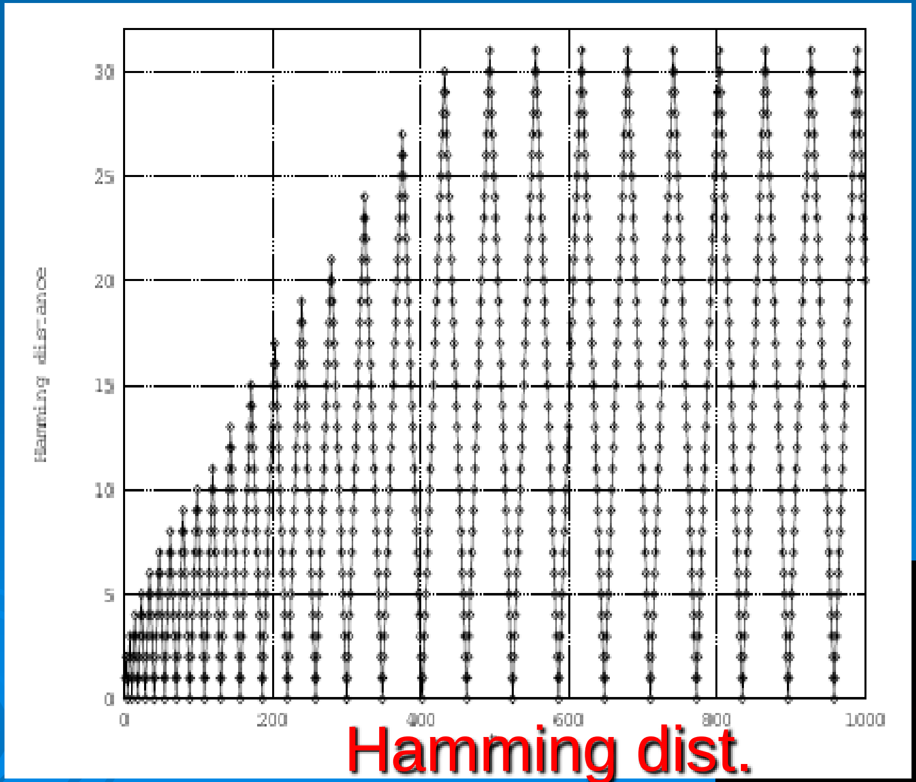
How to escape from an attractor

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

$$\text{REACT}(T) = \min\{\max\{T \times 1.1, T+1\}, L-2\}$$



T



Hamming dist.

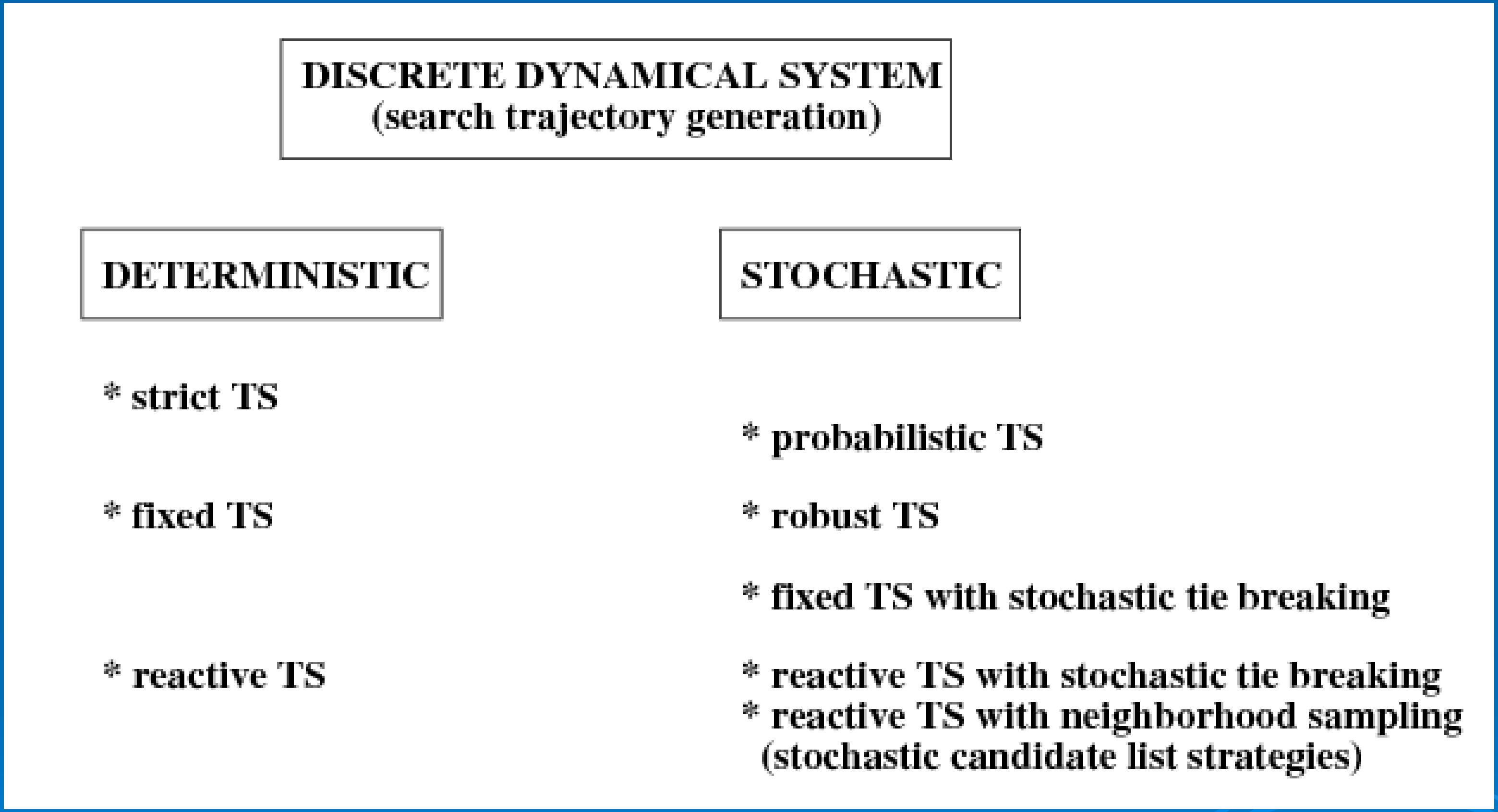
How to escape from an attractor

- 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.

Dynamical systems versus implementation (policies vs mechanisms)



DISCRETE DYNAMICAL SYSTEM
(search trajectory generation)

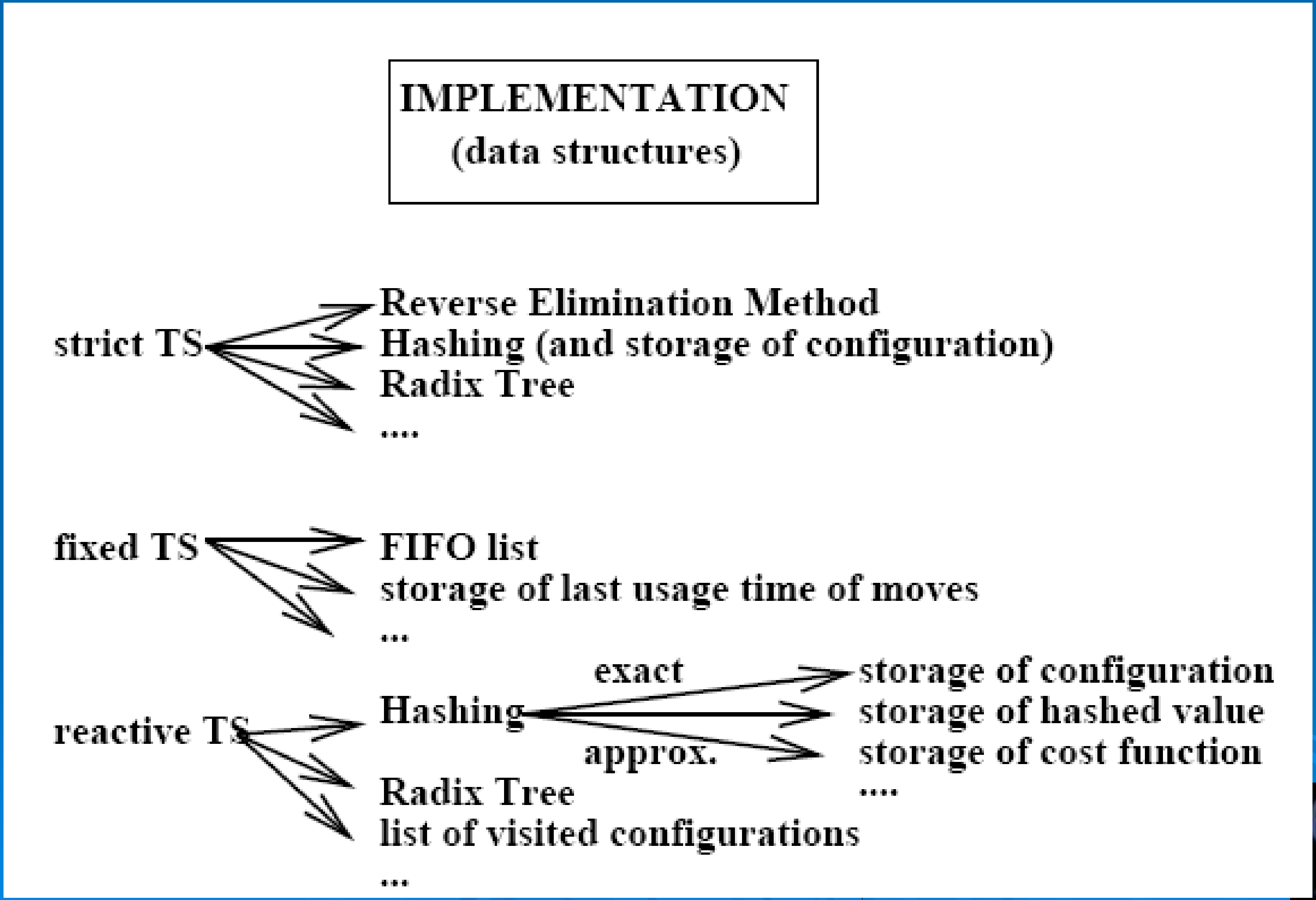
DETERMINISTIC

* strict TS
* fixed TS
* reactive TS

STOCHASTIC

* probabilistic TS
* robust TS
* fixed TS with stochastic tie breaking
* reactive TS with stochastic tie breaking
* reactive TS with neighborhood sampling (stochastic candidate list strategies)

Dynamical systems versus **implementation**



Other reaction opportunities

- **Objective function modifications**, tunneling, dynamic local search
- **Iterated Local Search**, kicks, …
- **Variable Neighborhood Search**
- **Different randomness levels** (SAT)

Open problems

- **Unification of methods to "escape from local minima" and qualitative differences**
- Distinguish very clearly dynamical systems from implementation
- Fitness surface complexity and information
- Algorithm engineering of efficient and flexible frameworks
- Parallel and distributed schemes (with "gossip-like" exchanges)
- Relationships with RL

Relationships between RS and RL

- **Reinforcement Learning / NDP**
  - sequential decision process, optimize “long term reward”
  - learning by interacting with an unknown environment
  - learn actions by experimenting and getting feedback
- **Reactive Search**
  - optimize f
  - self-tune local search by analyzing local search history
  - learns appropriate balance of intensification and diversification

Recent work: modelling RTS via RL

Reaction on $\mathcal{T}$ implemented via RL

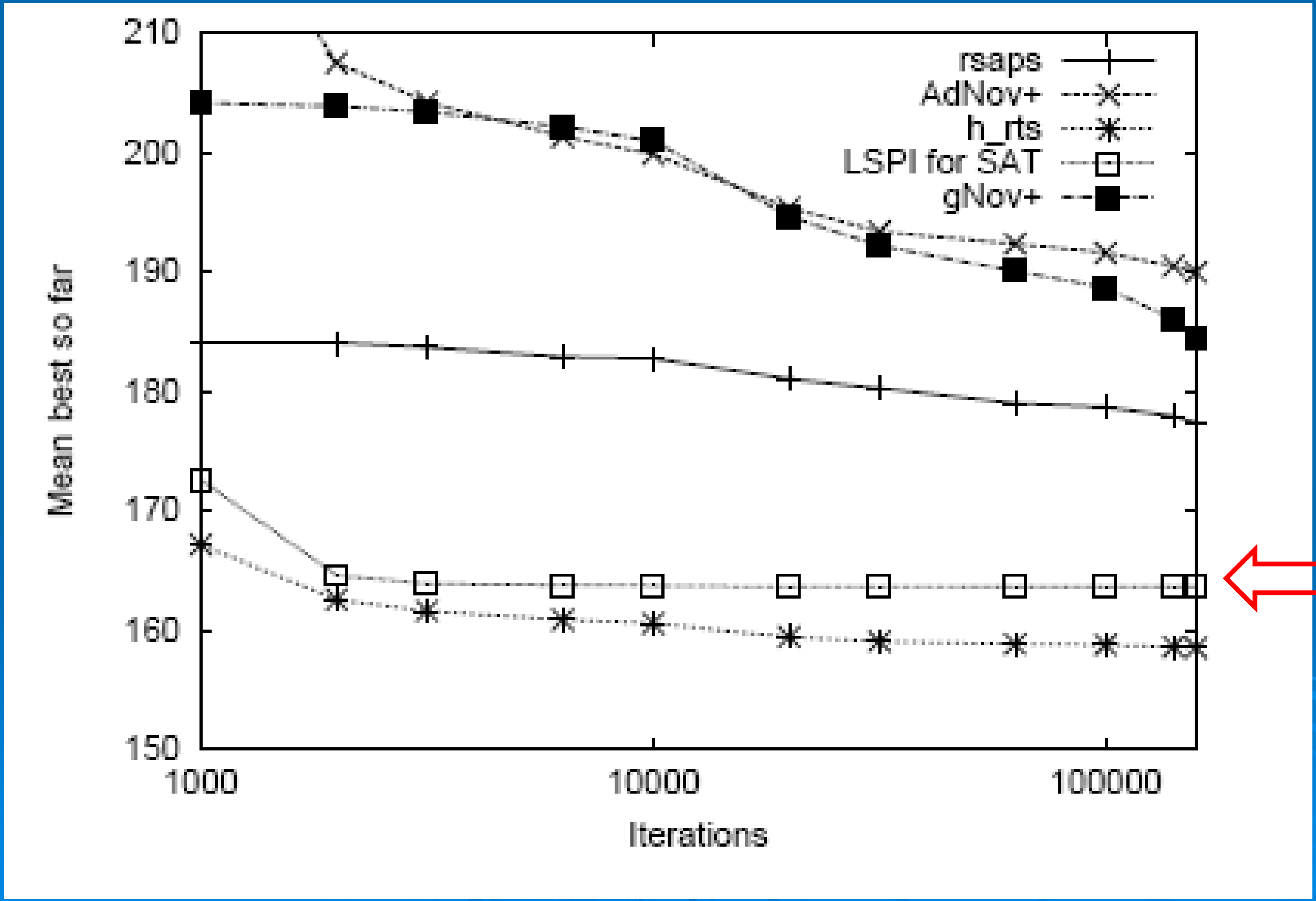
- MDP definition
- "Least-squares policy iteration" (LSPI) algorithm
  - Basis function definition
- Examples generation: training phase executed online while solving a single instance

Modelling RTS via RL

- Work in progress:
  - different features / reward, actions, basis functions tested
  - different SAT benchmarks (structured vs. random)
- Open issue:
  - improve run time performance!
- Results: LSPI-based approach is competitive!
  - In particular, on-line learning is also competitive

Experimental results

- **Comparison with state of the art MAX-SAT solvers**
  - **MAX-3-SAT random benchmark instances**



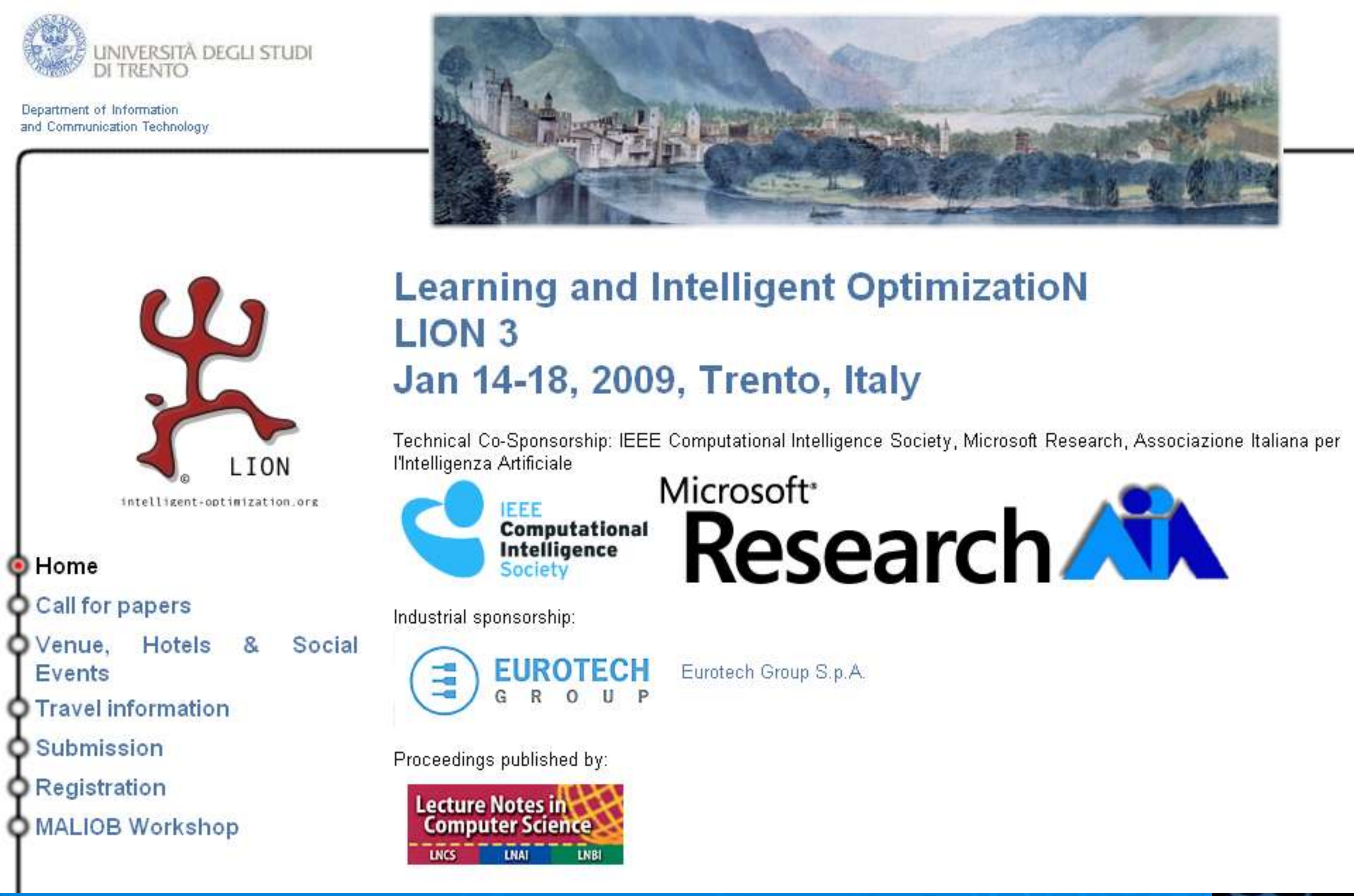
Additional info:

www.reactive-search.org
www.reactive-search.com

BOOK:

Reactive search and Intelligent Optimization
Battiti, Brunato and Mascia,
Springer Verlag, in press, 2008

(draft version in the web)



UNIVERSITÀ DEGLI STUDI DI TRENTO

Department of Information and Communication Technology

LION

intelligent-optimization.org

- Home
- Call for papers
- Venue, Hotels & Social Events
- Travel information
- Submission
- Registration
- MALIOB Workshop

Learning and Intelligent OptimizatioN
LION 3
Jan 14-18, 2009, Trento, Italy

Technical Co-Sponsorship: IEEE Computational Intelligence Society, Microsoft Research, Associazione Italiana per l'Intelligenza Artificiale

IEEE Computational Intelligence Society

Microsoft Research

Industrial sponsorship:

EUROTECH GROUP

Eurotech Group S.p.A.

Proceedings published by:

Lecture Notes in Computer Science

LNCS LNAI LNBI

http://www.intelligent-optimization.org/LION3/

THE END