1. Computação Evolutiva

Renato Tinós

Departamento de Computação e Matemática
Fac. de Filosofia, Ciência e Letras de Ribeirão Preto

Universidade de São Paulo
B R A S I L

Programa de Pós-Graduação Em Computação Aplicada

1.6. Aspectos Teóricos*

1.6.1. Is interesting to apply Evolutionary Algorithm (EAs) to my problem?

1.6.2. Introduction to the theory of EAs

1.6.3. Theory of Genetic Algorithms (GAs): Some Approaches

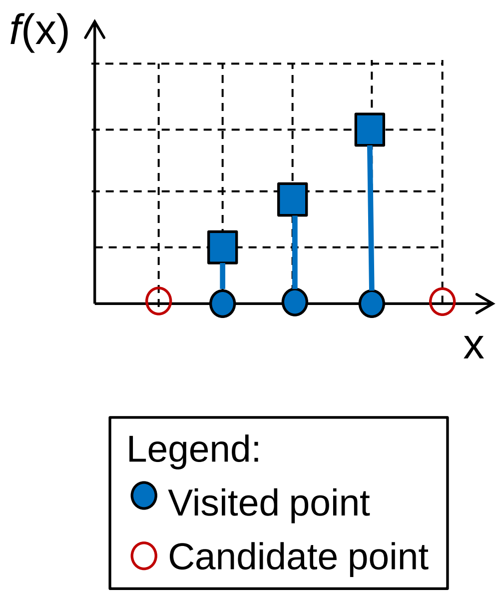
1.6.4. Examples

1.6.5. Conclusions

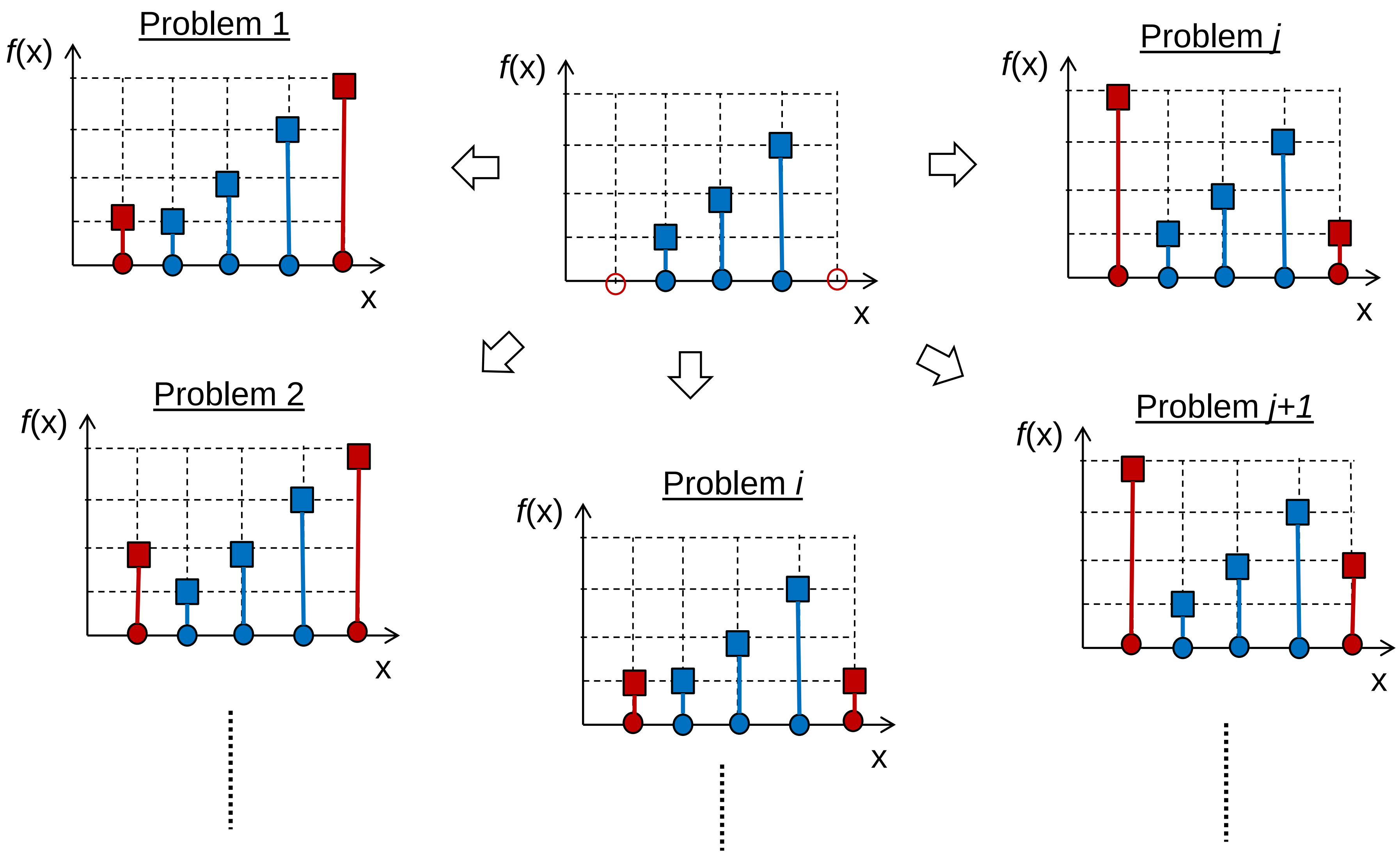
* Apresentado anteriormente como um tutorial na *Latin American and Brazilian Schools on Computational Intelligence* (LASCI & SBIC**)**, Curitiba, October 13th, 2015

1.6.1. Is interesting to apply EAs to my problem?

- **Black box optimization**
  - What should be the next point?



1.6.1. Is interesting to apply EAs to my problem?



1.6.1. Is interesting to apply EAs to my problem?

- ***No Free Lunch Theorem***
  - Considering all possible problems:

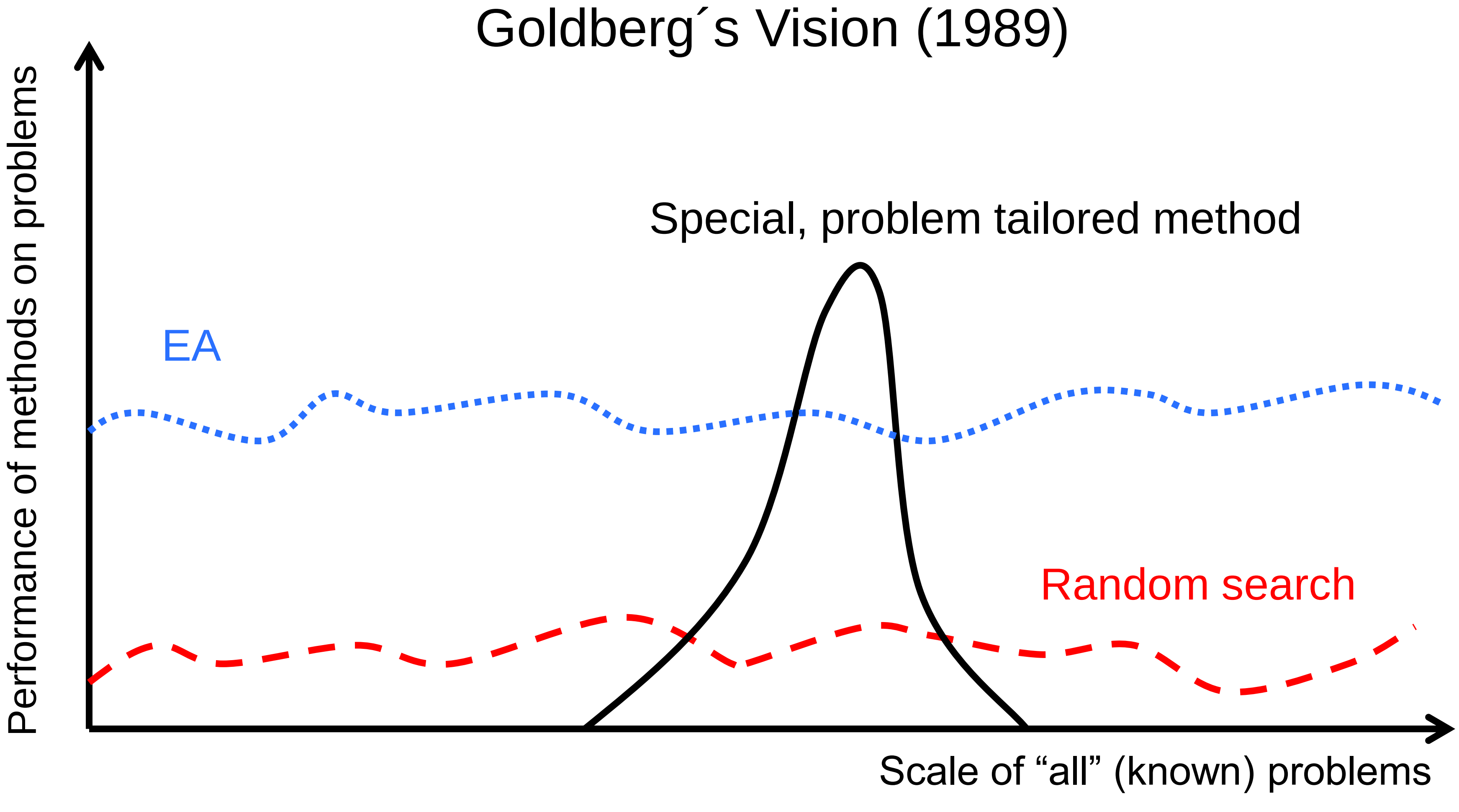
All black box algorithms that do not revisit points will show the same average performance!

  - Implications:
    - New black-box algorithm *A* is better than old black-box algorithm *B* (e.g., random walk without revisiting points) in half of the problems (in average)
      - But it is worse in the other half (in average)
    - So, why should we bother in creating new algorithms?

1.6.1. Is interesting to apply EAs to my problem?

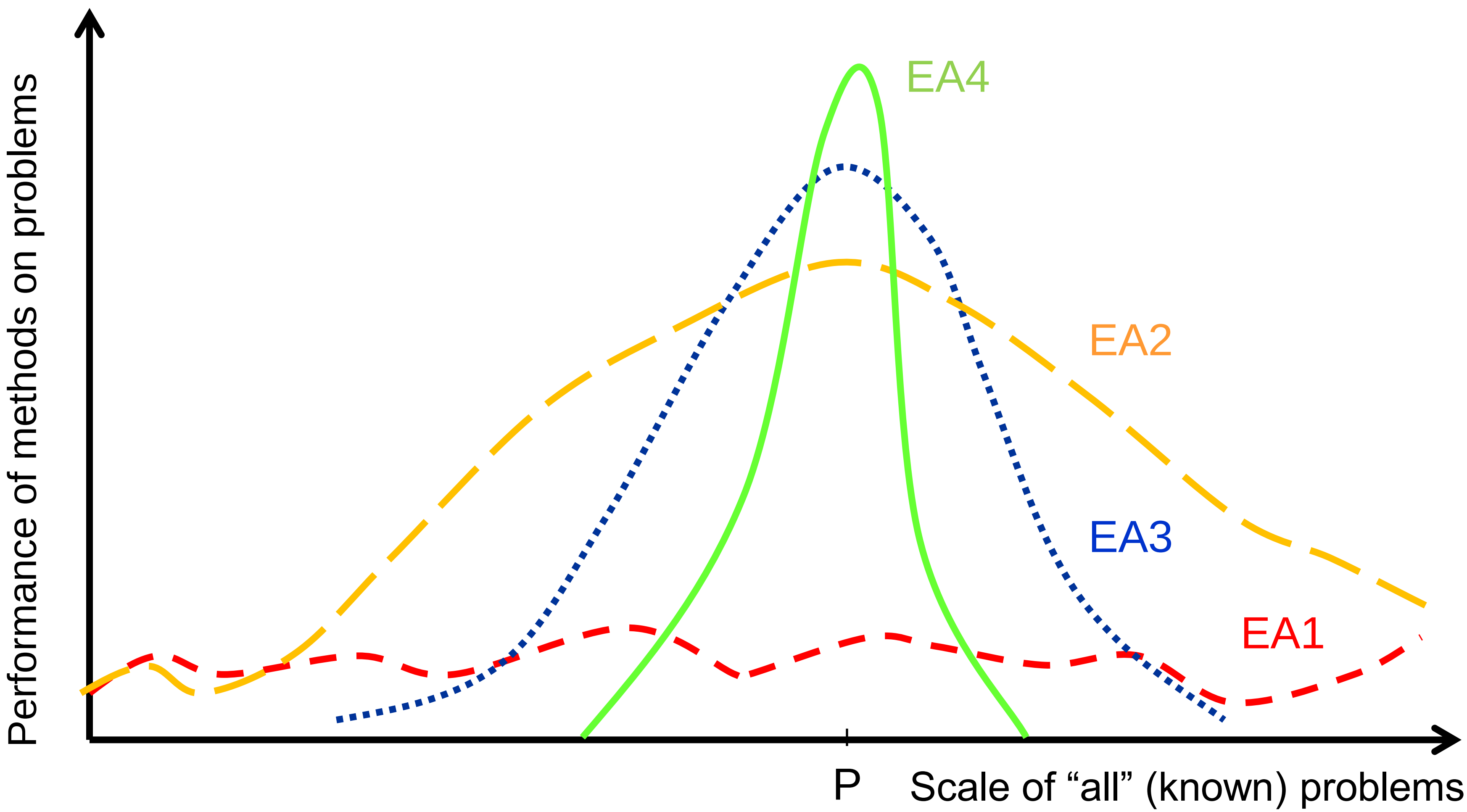
- **Why are then a lot of people applying EAs and obtaining good results?**
  - In fact, most of the “possible” problems are not interesting
  - In general, points in search spaces of real-world problems have continuous (and smooth) neighborhood
  - In such cases, experience has shown that EAs (and other search techniques) perform well

1.6.1. Is interesting to apply EAs to my problem?



1.6.1. Is interesting to apply EAs to my problem?

Michalewicz´s Vision (1996)



1.6.1. Is interesting to apply EAs to my problem?

- **HOWEVER, BE CAREFUL!**
- **When EAs (generally) should not be used?**
  - Instances of problems that can be solved by exact methods in “reasonable” time
    - Example: The researcher develops a “new” EA and test it in instances of the Travelling Salesman Problem (TSP) with *N*=100, 200 and 300 cities
      - However, the algorithm Concorde deterministically solves instances of the TSP with hundreds of cities in seconds

1.6.1. Is interesting to apply EAs to my problem?

- **When EAs (generally) should not be used?**
  - Problems where it is known that a global optimum can be found in “reasonable” time
    - Examples
      - Many of the problems with polynomial time complexity, i.e., $O(n^k)$

1.6.1. Is interesting to apply EAs to my problem?

- **When EAs (generally) should not be used?**
  - Problems where other optimization algorithms (traditional optimization algorithms, other metaheuristics, heuristics, …) perform better
    - For the given constraints to solve the problem
    - How do we know this?

i. **Experimental comparison**

- traditional method adopted by most of the researches

ii. **Theory** (when possible)

1.6.2. Introduction to the Theory of EAs

- **Is it possible to "predict" if applying a given EA to my problem will be interesting?**
  - In order to answer this question, we should understand how the EA works from a theoretical point of view

- **When applicable, theory can**
  - Provide performance guarantees for the algorithm
    - Examples: runtime analysis
  - Help designing new algorithms, operators, or modifications of the known algorithms
  - Help understanding the influence of the algorithm´s parameters
  - Eventually be used to explain phenomena in other areas, e.g., Biology

1.6.2. Introduction to the Theory of EAs

- **Some criticisms to EAs…**
  - *"There is no guarantee of convergence to global optimum!"* (?)
  - "*It is not possible to understand how EAs work!*" (?)
  - "*It is not possible to understand how the parameter´s setting influence the performance!*" (?)

- **In fact, there is a lot of questions that must be answered**
  - **We must not rely (only) on the inspiration of evolution by natural selection to justify the use of EAs**
  - However, some of the criticisms are not exclusive to EAs
    - Example: For all known optimization algorithms, we should be very careful when we speak about convergence for algorithms applied to problems in the NP class

1.6.2. Introduction to the Theory of EAs

- **So, why apparently the theory is much well understood in other algorithms?**

- **Difficulties with (a) Theory for EAs**
  - EAs are vast and complex dynamical systems with many degrees of freedom
  - EAs are generally applied to complex problems with fitness landscapes that are difficult to be properly modeled
  - EAs involve probabilistic operators
    - We often need statistical tools to analyze them
    - Results are many times described over average behavior
  - Parallel: a predictive model for biological evolution

1.6.2. Introduction to the Theory of EAs

- **In Evolutionary Computation, there are more theoretical studies for Evolution Strategies**
  - Real codification
  - Most of the papers on Theory deals with well-defined search-spaces

- **Runtime analysis (time complexity)**
  - Important question: How many iterations until global optima (or very good solutions) found?

    - http://www.cs.nott.ac.uk/~psxld/seminars/seminar_slides/pkl_seminar.pdf

1.6.3. Theory of GAs: Some Approaches

- **Here, we will discuss theory for Genetic Algorithms (GA)**

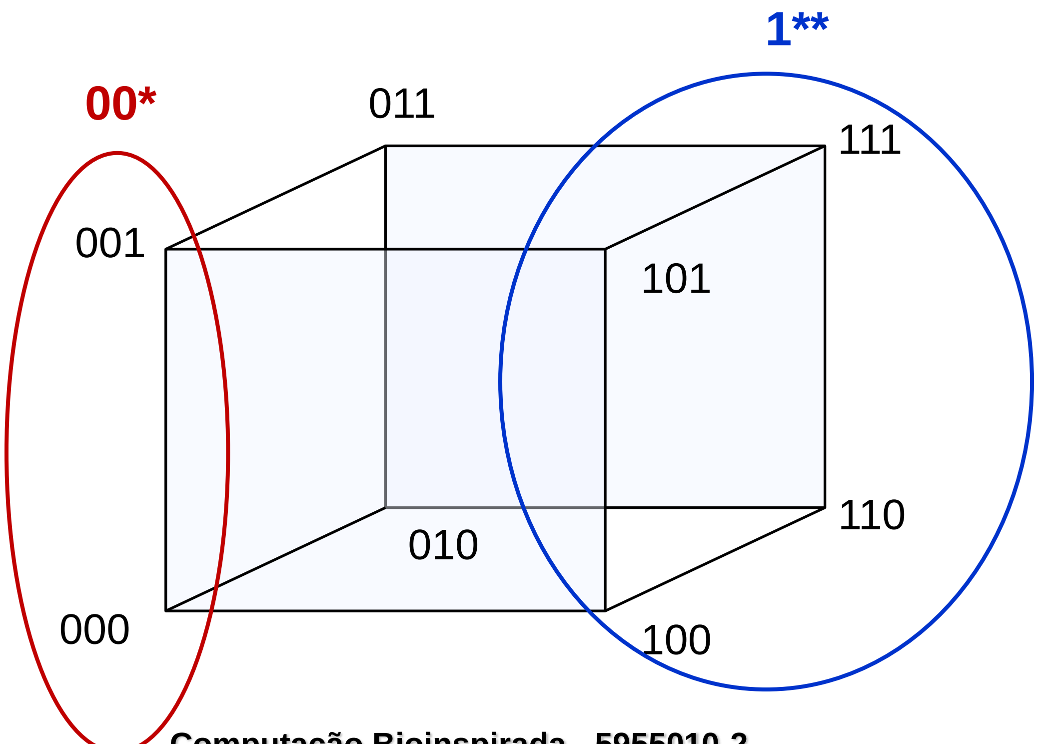
- **Some approaches:**
  - Schema Theorem
  - GA process investigated as a Markovian Process
  - GA seen as a dynamical system
    - Exact model
  - Mechanical statistics approach
  - Fitness landscapes approach

1.6.3.1. Theory of GAs: Schema Theorem

- **How do GAs work?**
  - One explanation for the operation of GAs is the **Building Blocks Hypothesis**
    - Building block
      - Short low-order **schema** with good fitness
    - Schema
      - Template describing a subset of solutions (strings) with similarity in some positions
        - In other words, a schema is a **hyperplane** in the search space

1.6.3.1. Theory of GAs: Schema Theorem

- **Schemata (considering the binary representation)**
  - *Strings* composed by 0, 1, * ("don't care")
    - Examples



1.6.3.1. Theory of GAs: Schema Theorem

- **Why do we use shemata?**
  - They represent a subset of solutions (chromosomes) instead of only one solution
    - The analysis of the population becomes easier
  - Some positions of the solutions (genes) may be more important to the optimization process than others
  - Reproduction operators can change a chromosome and not change a schema

1.6.3.1. Theory of GAs: Schema Theorem

- **Properties of a schema *H***
  - **Order** $o(H)$
    - Number of fixed positions
    - Examples: $o($ 011*1** $) = 4$, $o($ 0*1*1** $) = 3$
  - **Defining lenght** $\delta(H)$
    - Distance between first and last fixed positions
    - Examples: $\delta($ *011**1* $) = 5$, $\delta($ 0*1***1 $) = 6$

1.6.3.1. Theory of GAs: Schema Theorem

- **Holland´s formulation for the Standard GA**
  - Standard GA: fitness proportionate parent selection, one point crossover, and bit-flip mutation
  - Considering a chromosome of length *l* that contains a schema *H*. The probability of disrupting the schema
    - By crossover is:

$$P_{crossover}(H) = \frac{\delta(H)}{l-1}$$

    - By mutation is:

$$P_{mutation}(H) = o(H) p_m$$

1.6.3.1. Theory of GAs: Schema Theorem

- **Holland´s formulation for the standard GA**
  - Combining with proportionate selection, we have the equation for the expected number of individuals representing schema *H* in next generation

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{l-1} - o(H)p_m\right]$$

Schema Theorem

The number of instance within the population of short low-order schemata of above-average fitness will increase exponentially in subsequent generations

1.6.3.1. Theory of GAs: Schema Theorem

<u>Building Blocks Hypothesis</u>

Short low-order schemata of above-average fitness (building blocks) are combined and recombined to form strings with potentially better fitness

- **In this way, the complexity of the problem would be reduced**
  - Instead of building strings with high fitness directly, a procedure that requires trying all possible combinations of genes, strings each time better would be built by combining the best partial solutions found by various strings with lower order

1.6.3.1. Theory of GAs: Schema Theorem

- **The Two-Armed Bandit Problem**
  - Consider a machine with two independent arms
    - One arm pays an average reward of μ_1 (with variance σ_1^2) while the other arm pays an average reward of μ_2 (with variance σ_2^2)
    - Which arm should we explore?
    - An “optimal” strategy is to exponentially increase the number of trials in the best observed arm

1.6.3.1. Theory of GAs: Schema Theorem

- **The K-Armed Bandit Problem**
  - In the GAs, we are not solving the previous problem, but a problem where *K* hyperplanes (schemata) are explored simultaneously
  - According to Holland, the GA approaches the "optimal" strategy to exponentially increase the number of attempts of the current best hyperplanes (schemata)

1.6.3.1. Theory of GAs: Schema Theorem

- **Arguing pro...**
  - The building block hypothesis “can” be used to explain why some problems are difficult for GAs
    - Examples
      - Deceptive Problems: when low-order schemata, rather than combining to generate higher-order promising schemata, combine to form schemata that result in suboptimal solutions
      - Scaling: when some schemata have very higher fitness when compared to others

1.6.3.1. Theory of GAs: Schema Theorem

- **Arguing con…**
  - There are criticisms about the schema theorem and the building block hypothesis. Some of them:
    - It does not consider the constructive effects of crossover and mutation
    - Due to the use of the estimated fitness of a given schema, the theorem says nothing about the future generations from $t + 2$
    - Problems designed to be easier for the GA according to the building blocks hypothesis (e.g., Royal Road Functions) are sometimes easier for other algorithms, e.g., hill-climbing with random mutation

1.6.3.1. Theory of GAs: Schema Theorem

- **Arguing pro (again)...**
  - Algorithms and operators can be designed to explicitly explore the building blocks
    - Examples: some **Estimation of Distribution Algorithms** (**EDAs**) explicitly identify and recombine building blocks (gene linkage)
      - Messy GA
      - Population-based incremental learning (PBIL)
      - Compact Genetic Algorithm (cGA)

1.6.3.1. Theory of GAs: Schema Theorem

- **Arguing pro (again)...**
  - One legacy of the interest in schemata: application of Walsh transforms to binary GAs
    - **Walsh Transforms**
      - Allow to perform function decomposition for binary representation
        - Parallel: Fourier Transform decomposes a continuous function (signal dependent on time)
        - Parallel: Walsh coefficients similar to Fourier coefficients (frequencies)
      - Can be used in theoretical studies

1.6.3.2. Theory of GAs: Exact Model

- **Initially proposed by M. Vose**
- **Simple GA is seen as a discrete dynamical system**
  - The Exact Model Approach is also known as Dynamical Systems Approach
  - Dynamical system
    - Mathematical concept in which fixed rules describe the dependence on time of a state in a geometric space (state space)

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

1.6.3.2. Theory of GAs: Exact Model

- **GA as a discrete dynamical system**
  - Let *n* be the (finite) size of the search space
    - If the chromosome has *l* elements, then $n = 2^l$
  - In the exact model, *all possible candidate solutions* are represented in a discrete space with *n* dimensions
    - Thus, the current population of the GA can be described as an *n*-dimensional vector
      - Each element defines the proportion of each candidate solution in the population, i.e., $p(k) = v(k) / N$, where
        - The *k*-th element of **v** indicates the number of copies of the *k*-th candidate solution in the population of size *N*

1.6.3.2. Theory of GAs: Exact Model

- **GA as a discrete dynamical system**
  - As the sum of elements in **p** is equal to 1, the vector population may be described as belonging to a simplex

$$\Lambda = \left\{ \mathbf{p} \in \mathbb{R}^n : p_k \geq 0, \text{ for } k = 0, 1, \ldots, n-1 \text{ and } \sum_{k=0}^{n-1} p_k = 1 \right\}$$

  - Thus, the GA's behavior is seen as a trajectory in a simplex

1.6.3.2. Theory of GAs: Exact Model

- **GA as a discrete dynamical system**
  - The model considers infinite population (exact model)
    - However, GAs with finite population can be analyzed
      - The deviation (relative to the trajectory for the infinite population) is inversely proportional to the population size

1.6.3.2. Theory of GAs: Exact Model

- **Dynamical System of the GA**

$$\mathbf{p}(t) = \mathcal{G}\left(\mathbf{p}(t-1), t\right)$$

$$\mathbf{p}(t) = \mathcal{G}^t\left(\mathbf{p}(0)\right)$$

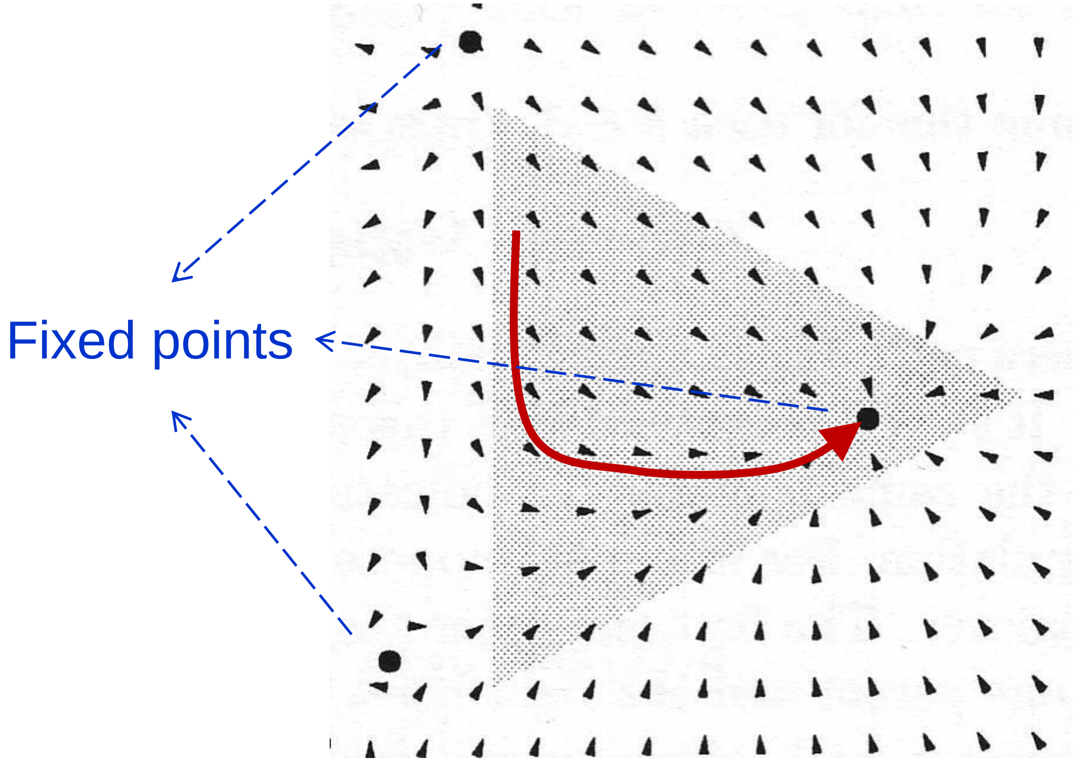
- **For the GA with (bit-flip) mutation and proportional selection**

$$\mathcal{G}(\mathbf{p}) = \frac{UF\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}}$$

  - ***U***: *nxn* matrix representing the transitions due to mutation
  - ***F***: *nxn* diagonal matrix with the fitness of each candidate solution

1.6.3.2. Theory of GAs: Exact Model

- **Dynamical System of the GA**
  - Example
    - figure adapted from [REEVES & ROWE, 2004]



1.6.3.2. Theory of GAs: Exact Model

- **Some observations**

  - The existence, location and stability of fixed points and attractors can be defined by the analysis of the generational operator

  - For GAs with proportional selection and bit-flip mutation, fixed points and attractors are given by the eigenvectors of ***UF***

  - All population trajectories converge to the main fixed point (in which part of the population is in a global optimum)
    - In other words: the system is <u>asymptotically stable</u>

1.6.3.2. Theory of GAs: Exact Model

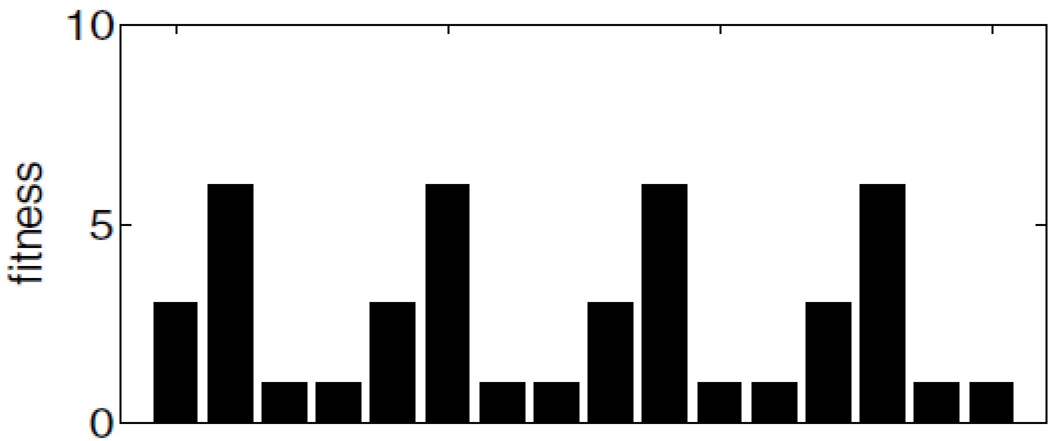
- **Some observations**
  - The other eigenvectors are related to <u>metastable states</u>
    - They play very important roles in the evolutionary process
      - They can change the trajectory in the simplex and trap the population for generations
        - Local optima
  - Similar analysis can be made for the case with crossover and with other operators
  - The exact model allow to understand the effects of parameters like population size and mutation rate

1.6.3.2. Theory of GAs: Exact Model

- **Problems**
  - A very large number of equations to be analyzed for practical problem
    - In general, applicable for small solution spaces
  - The fitness of all candidate solutions must be known

1.6.3.3. Theory of GAs: Fitness Landscapes

- **Fitness landscape**
  - What is a fitness landscape?



0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

1.6.3.3. Theory of GAs: Fitness Landscapes

- **Fitness landscape**
  - The landscape observed for a particular function is an artifact of the algorithm used
    - In other words: of the neighborhood induced by the operators
  - Can be defined by the triple (S,n,f) where
    - S: search space
    - $n(\mathbf{x})$: neighborhood function
    - $f(\mathbf{x})$: fitness function
  - In this way, when analyzing the fitness landscape, it is essential to analyze the neighborhood structure induced by the operators

1.6.3.3. Theory of GAs: Fitness Landscapes

- **Fitness landscape**
  - How the neighborhood relations are defined?
    - We can use an adjacency matrix **A**
    - Using the diagonal matrix, **D**, containing the degrees of each vertex, we can still define the graph Laplacian

$$\boldsymbol{\Delta} = \mathbf{A}-\mathbf{D}$$

1.6.3.3. Theory of GAs: Fitness Landscapes

- **Elementary landscapes**
  - Fitness landscapes that satisfy for all points **s** the following equation

$$\boldsymbol{\Delta}\, f(\mathbf{s}) + (C\,/\,m)\, f(\mathbf{s}) = 0$$

    - C is a problem-specific parameter
    - m is the size of the problem instance
  - Several combinatorial optimization problems, e.g., TSP, have elementary landscapes
    - All minima are lower, all maxima are higher than the averaged fitness (f_m) for all candidate solutions in the space
    - Cost to find a local optimum in a maximization problem using neighborhood search (under mild conditions on the nature of the fitness): $O(\ m \log_2(f_{max}/f_m)\ )$

1.6.4.1. Example: Dynamical Systems Approach Applied to DOPs

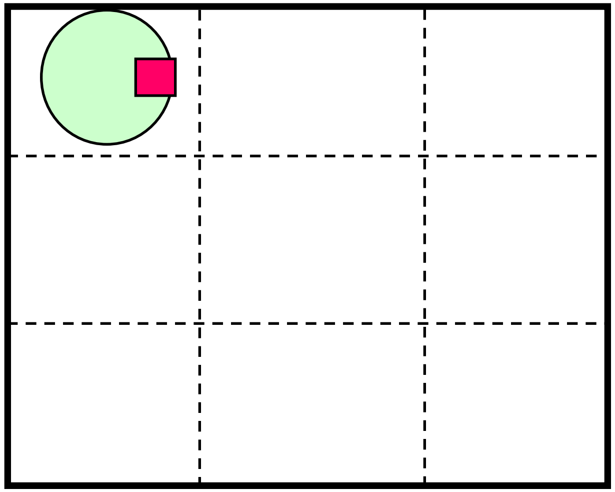
- **Dynamic Evolutionary Optimization**
  - EAs applied to Dynamic Optimization Problems (DOPs)
    - The fitness landscape changes during the optimization process
    - Example: Evolutionary Robots
      - Robots totally or partially designed by EAs
      - In general, when the fitness of the solutions are experimentally obtained (e.g., when the individual of the EA defines a control law that is tested during a period of time in a real robot), days are required for the optimization process
        - During this long period, changes often occur:
          - \- In the robot. Examples: Battery charge oscillation, faults, …
          - \- In the environment. Examples: illumination variation, …

1.6.4.1. Example: Dynamical Systems Approach Applied to DOPs

- **Problem**
  - Analysis of the fitness modifications in a dynamic problem with evolutionary robots (simulations)
    - Problem: simple navigation task
      - DOP:
        - Faults occur in the robot during the optimization process
    - The analysis of the fitness modifications in the problems studied here, and in other problems too, can help the development and analysis of benchmark DOP generators

1.6.4.1. Example: Dynamical Systems Approach Applied to DOPs

- **Problem**
  - Mobile robot with a frontal sensor
  - Controller: *l*-dimensional binary vector (control vector)
    - Indicates the action for each possible state
      - State: input from sensor and memory of last action
  - Fitness of the individual (control vector)
    - number of positions occupied by the robot during 10 iterations or until the robot hits a wall



1.6.4.1. Example: Dynamical Systems Approach Applied to DOPs

- **Problem**
  - Model 1 (*l*=4 bits): two actions
    - Move forward and rotate 90 degrees
  - Model 2 (*l*=8 bits): four actions
    - Move forward, rotate 90 or -90 degrees, wait
  - Three faults can occur in the robot
    - Fault 1: sensor inputs always equal to zero
    - Fault 2: sensor inputs always equal to one
    - Fault 3: wrong sensor inputs
  - Each fault represents a different change (DOP)
  - The effects of the changes on the fitness vector were analysed and the dynamical system was simulated

Simulation: GA with mutation and proportional selection

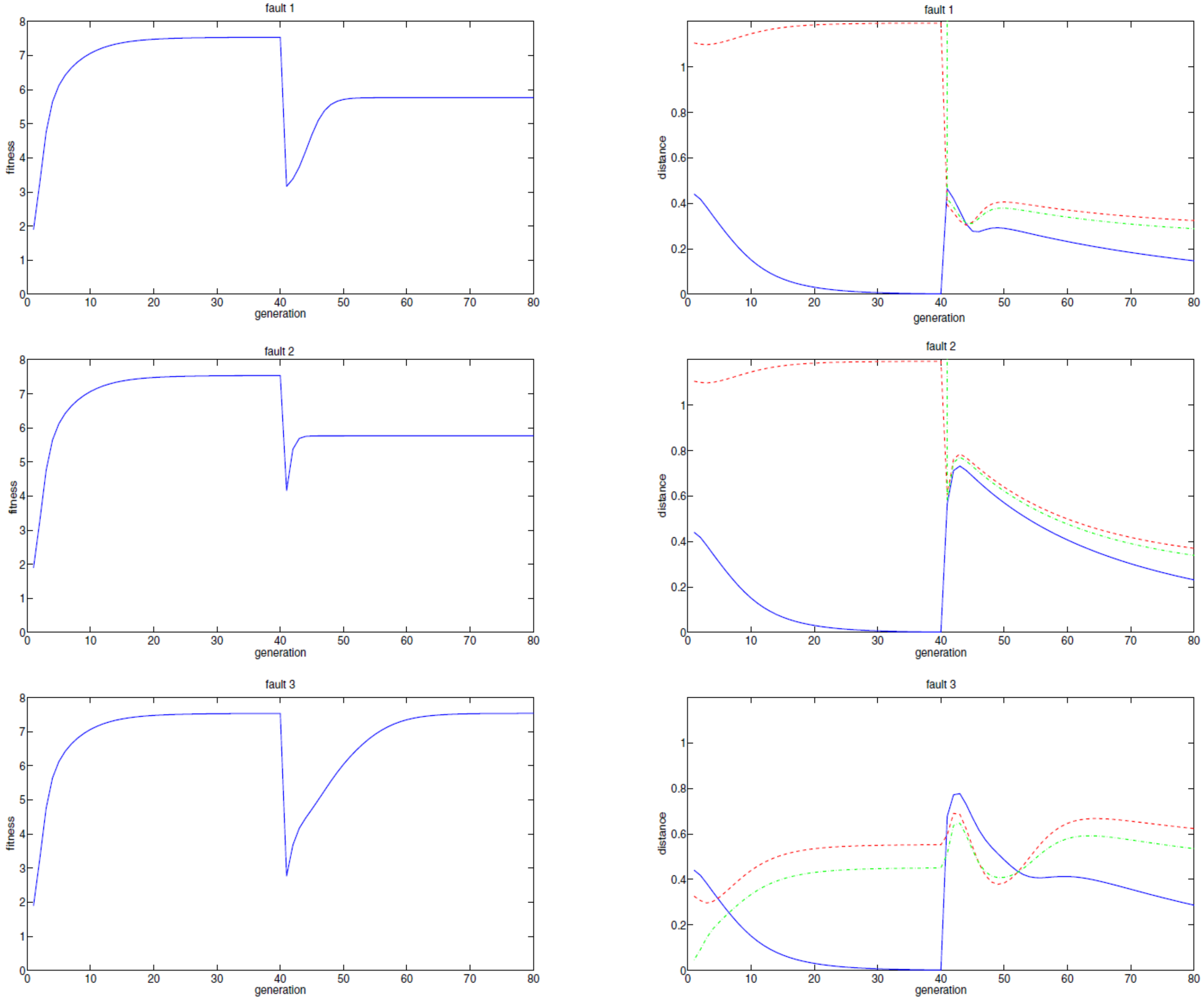


Figure 3: Mean fitness and distances from the population to three metastable states for problem 1 with $l = 8$. The solid line shows the distance to the main metastable state.

1.6.4.2. Example: Partition Crossover

- **k-bounded pseudo-Boolean optimization**
  - Example: NK Landscape Model
    - Cost function given by

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}, \mathbf{m}_i)$$

x: solution with size *N*

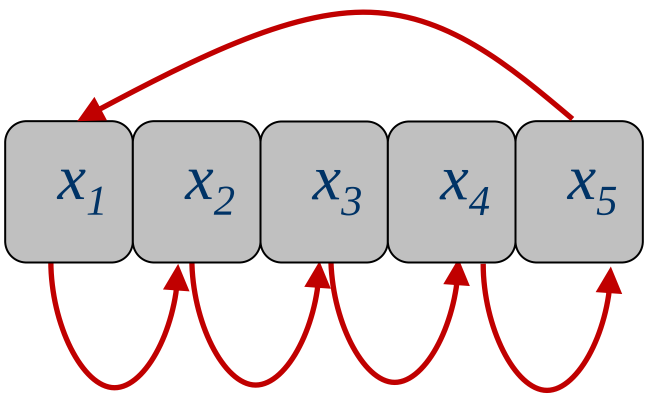
$\mathbf{m}_i$: binary mask with size *N* and *K*+1 ones

K: integer controlling the epistasis degree

1.6.4.2. Example: Partition Crossover

- **Example**
  - N=5, K=1, adjacent neighborhood

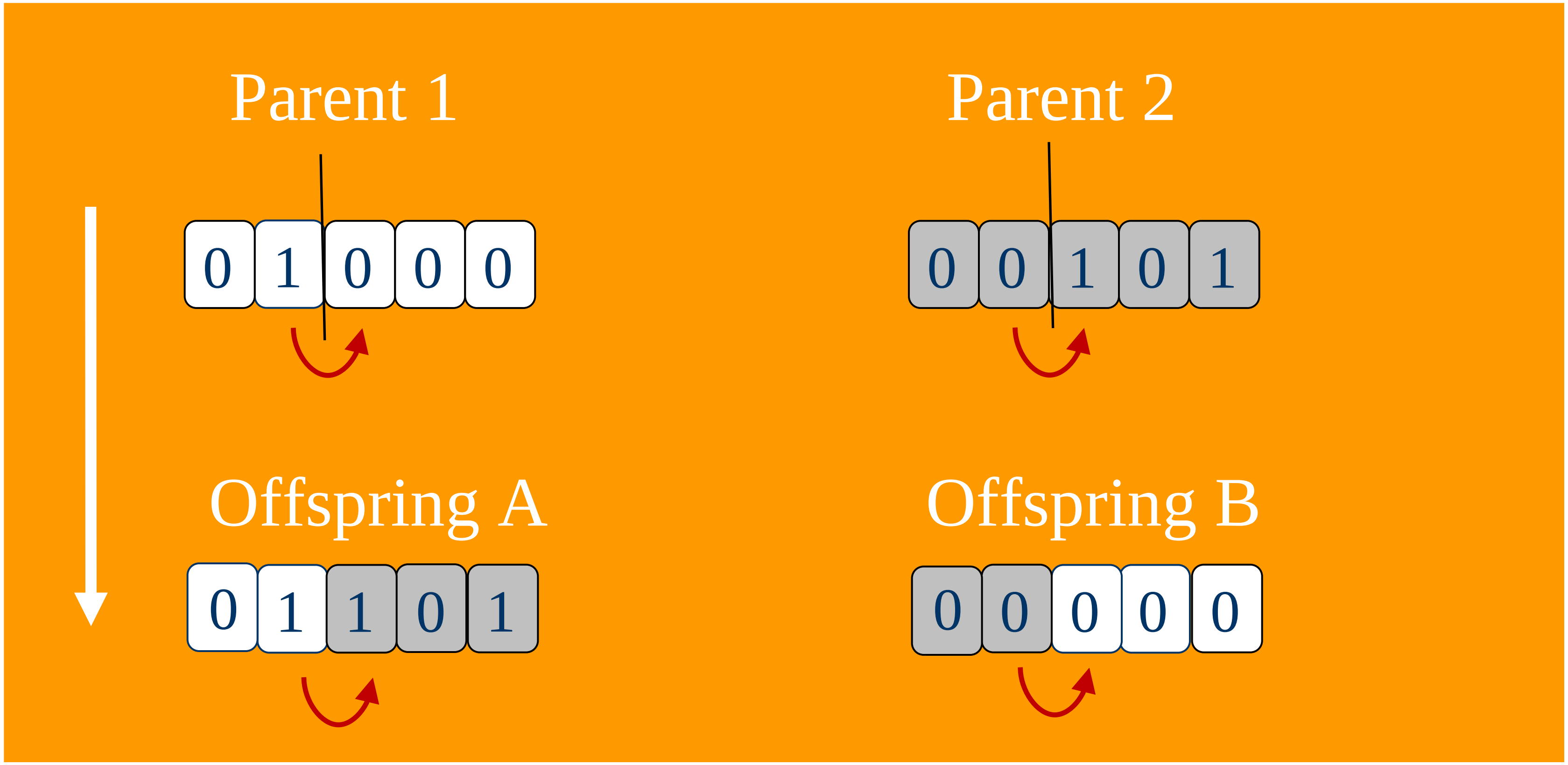
$$f(\mathbf{x}) = \frac{1}{5}\left(f_1(x_1, x_5) + f_2(x_2, x_1) + f_3(x_3, x_2) + f_4(x_4, x_3) + f_5(x_5, x_4) \right)$$



1.6.4.2. Example: Partition Crossover

$$f(\mathbf{x}) = \frac{1}{5}\left(f_1(x_1, x_5) + f_2(x_2, x_1) + f_3(x_3, x_2) + f_4(x_4, x_3) + f_5(x_5, x_4) \right)$$

1-point crossover



1.6.4.2. Example: Partition Crossover

- **How do we preserve the interaction of the solution components in order to allow the linear decomposition of the cost function?**

$$f(\mathbf{x}) = \frac{1}{5}\left(f_1(x_1, x_5) + f_2(x_2, x_1) + f_3(x_3, x_2) + f_4(x_4, x_3) + f_5(x_5, x_4)\right)$$

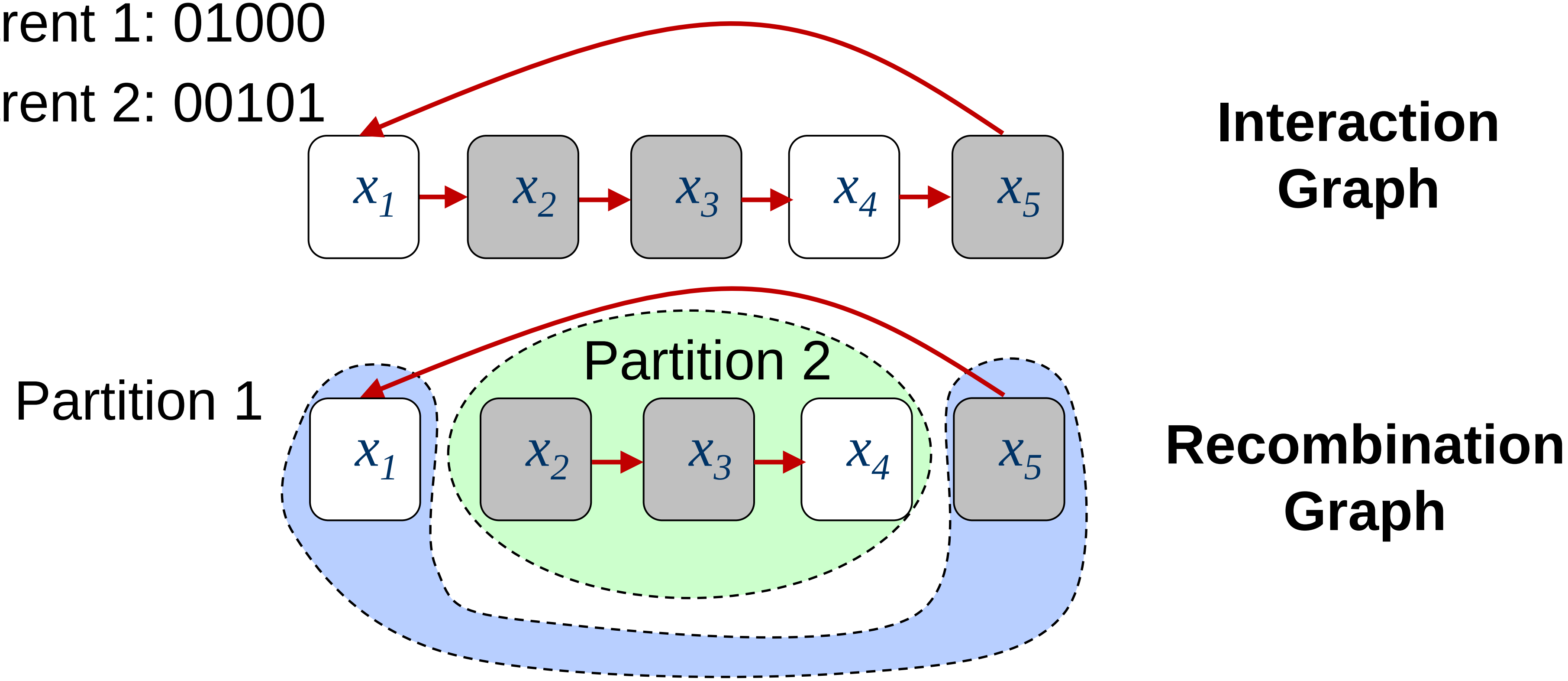
  - **Solution:** we partition the solutions according to the interactions and common characteristics of the parents - *Recombination by Decomposition*
    - Example:

Partition Crossover

1.6.4.2. Example: Partition Crossover

Parent 1: 01000

Parent 2: 00101



$$f(\mathbf{x}) = \left(f_1(x_1, x_5) + f_2(x_2, x_1) + f_3(x_3, x_2) + f_4(x_4, x_3) + f_5(x_5, x_4)\right)/N$$

$$f(\mathbf{x}) = \left(f_{p1}(\mathbf{x}) + f_{p2}(\mathbf{x})\right)/N$$

$$f_{p1}(\mathbf{x}) = f_1(x_1, x_5) + f_5(x_5, x_4)$$

$$f_{p2}(\mathbf{x}) = f_2(x_2, x_1) + f_3(x_3, x_2) + f_4(x_4, x_3)$$

1.6.4.2. Example: Partition Crossover

- **Analysis**

q is the number of partitions found

  - The decomposition of the evaluation function allows to deterministically find the best among 2^q offspring at the cost of evaluating only two solutions

➢ Thus, the cost of finding the best among a number of offspring that grows <u>exponentially</u> with the number of partitions is <u>linear</u> (if the cost of evaluating one solution is linear)

1.6.4.2. Example: Partition Crossover

Table 1: Percentage over 50 runs where the global optimum was found (Found) in the experiments of the hybrid GA with the adjacent model. The average percentage difference (% Difference) with respect to the global optimum evaluation is also given.

		2-point crossover		Uniform crossover		PX		PX fit/dist selection	
N	K	Found	% Difference	Found	% Difference	Found	% Difference	Found	% Difference
100	1	100	0.000 ±0.000	88	0.011 ±0.051	100	0.000 ±0.000	100	0.0000 ±0.0000
100	2	90	0.007 ±0.029	24	0.294 ±0.347	100	0.000 ±0.000	100	0.0000 ±0.0000
100	3	62	0.086 ±0.192	4	0.657 ±0.432	100	0.000 ±0.000	100	0.0000 ±0.0000
300	1	18	0.090 ±0.087	0	0.859 ±0.262	100	0.000 ±0.000	100	0.0000 ±0.0000
300	2	0	0.611 ±0.212	0	2.157 ±0.431	100	0.000 ±0.000	100	0.0000 ±0.0000
300	3	0	1.503 ±0.402	0	3.464 ±0.506	80	0.009 ±0.023	98	0.0001 ±0.0007
500	1	0	0.364 ±0.153	0	1.371 ±0.307	100	0.000 ±0.000	100	0.0000 ±0.0000
500	2	0	1.398 ±0.327	0	3.261 ±0.377	98	0.001 ±0.004	98	0.0011 ±0.0078
500	3	0	2.791 ±0.467	0	4.851 ±0.518	40	0.029 ±0.042	70	0.0079 ±0.0183

1.6.4.2. Example: Partition Crossover

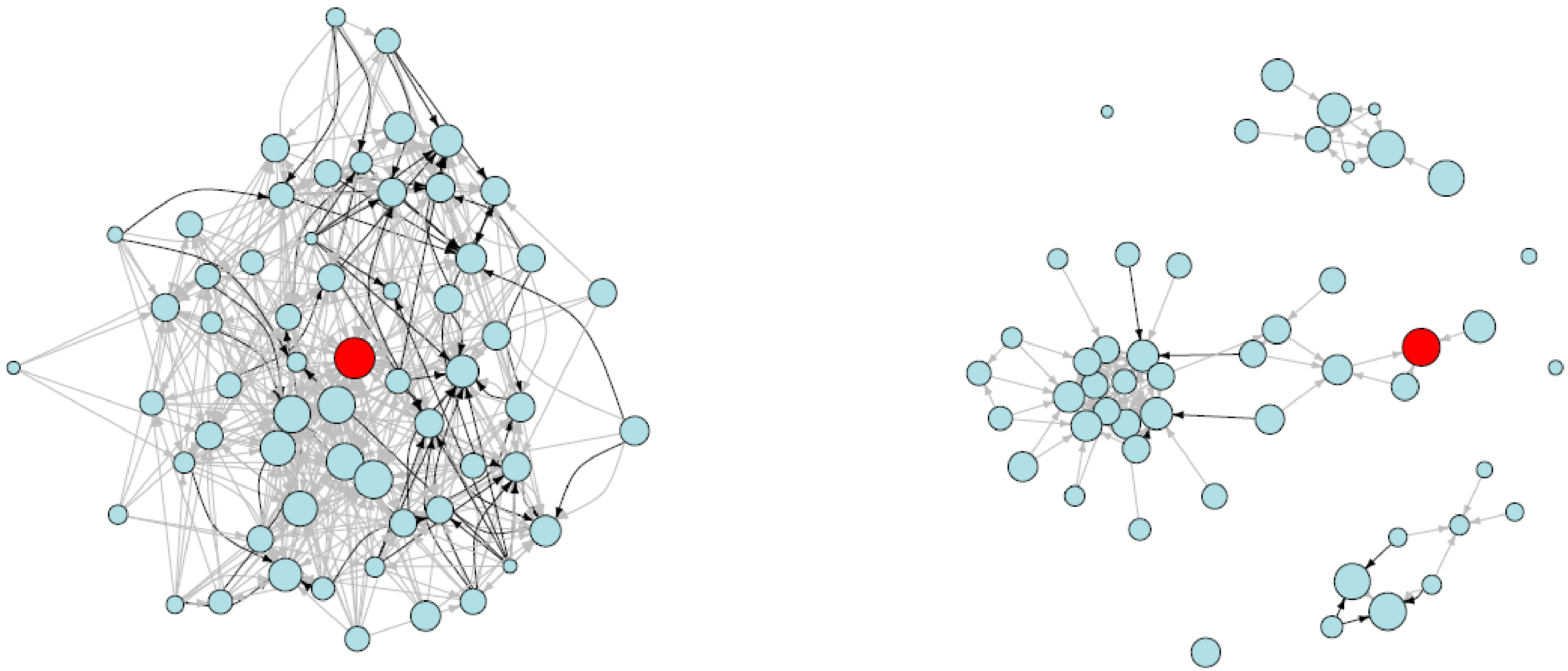


Figure 2: PX local optima networks for two selected instances with values $N = 20$, $K = 2$, $q = 100$. Nodes are local optima and edges connect parents to offspring after partition crossover (black edges indicate PX followed by hill-climbing). Vertex area is proportional to their fitness and the global optimum is highlighted in red (darker color). Left: Adjacent model, the network has 60 nodes and features a single connected component. Right: Random model, the network has 50 nodes (local optima) and features 7 connected components, 4 of which are isolated nodes.

1.6.5. Conclusions

- **We must not rely (only) on the inspiration of evolution by natural selection to explain how EAs work**
- **Theoretical studies are necessary**
- **There are few theoretical studies in Evolutionary Computation**
  - Main difficulties
    - EAs are vast, non-determistic, complex dynamical systems
    - There is not a general method applicable to all situations
      - We do not know completely the fitness landscape in most of the real-word problems

1.6.5. Conclusions

- **However, there are several cases of success**
  - Example: runtime analysis for several problems
- **Theory, when applicable, can**
  - Provide performance guarantees for the algorithm
  - Help designing new algorithms and operators
  - Help understanding the influence of the algorithm´s parameters
  - Eventually be used to explain phenomena in other areas
- **Anyway, experimental comparison is essential**

References

- **Theory of EAs**
  - Books
    - M. D. Vose (1999). "*The Simple Genetic Algorithm: Foundations and Theory*", MIT Press.
    - Reeves, C. R. & Rowe, J. E. (2003). "*Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory*", Kluwer Academic Publishers.
    - Jansen, T. (2013). "*Analyzing Evolutionary Algorithms: The Computer Science Perspective*", Springer.
  - Chapter
    - Chapter 11: Eiben, A. E. & Smith, J. E. (2003). "*Introduction to Evolutionary Computation*", Springer.

References

- **Examples cited here**

- TINÓS, R. (2012). “Analysing Fitness Landscape Changes in Evolutionary Robots*”. The 1st Understanding Problems Workshop (GECCO-UP), In: Companion Publication of the 2012 Genetic and Evolutionary Computation Conference (GECCO'12)*, July 7–11, 2012, Philadelphia, PA, USA. ACM 2012, ISBN 978-1-4503-1178-6, pp. 385-392.
- TINÓS, R. & YANG, S. (2014). “Analysis of fitness landscape modifications in evolutionary dynamic optimization”, *Information Sciences*, vol. 282, p. 214-236.
- TINÓS, R.; WHITLEY, D. & CHICANO, F. (2015). “Partition Crossover for Pseudo-Boolean Optimization”. *In: Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII* (FOGA '15). ACM Press, pp. 137-149.
- OCHOA, G.; CHICANO, F.; TINÓS, R. & WHITLEY, D. (2015). “Tunnelling Crossover Networks”. *In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO’2015)*, ACM Press, pp. 449-456.