ADAPTIVE MEMORY PROGRAMMING
FOR
SCHEDULING PROBLEMS

Éric Taillard

Fourth Workshop on Models and Algorithms for Planning and Scheduling Problems
Renesse, The Netherlands, June 14-18 1999

ADAPTIVE MEMORY PROGRAMMING
FOR SCHEDULING PROBLEMS

Adaptive memory programming (AMP)

Definition and general considerations

Flow shop sequencing

Taboo search

FANT

Genetic Hybrid

Vehicle routing and scheduling

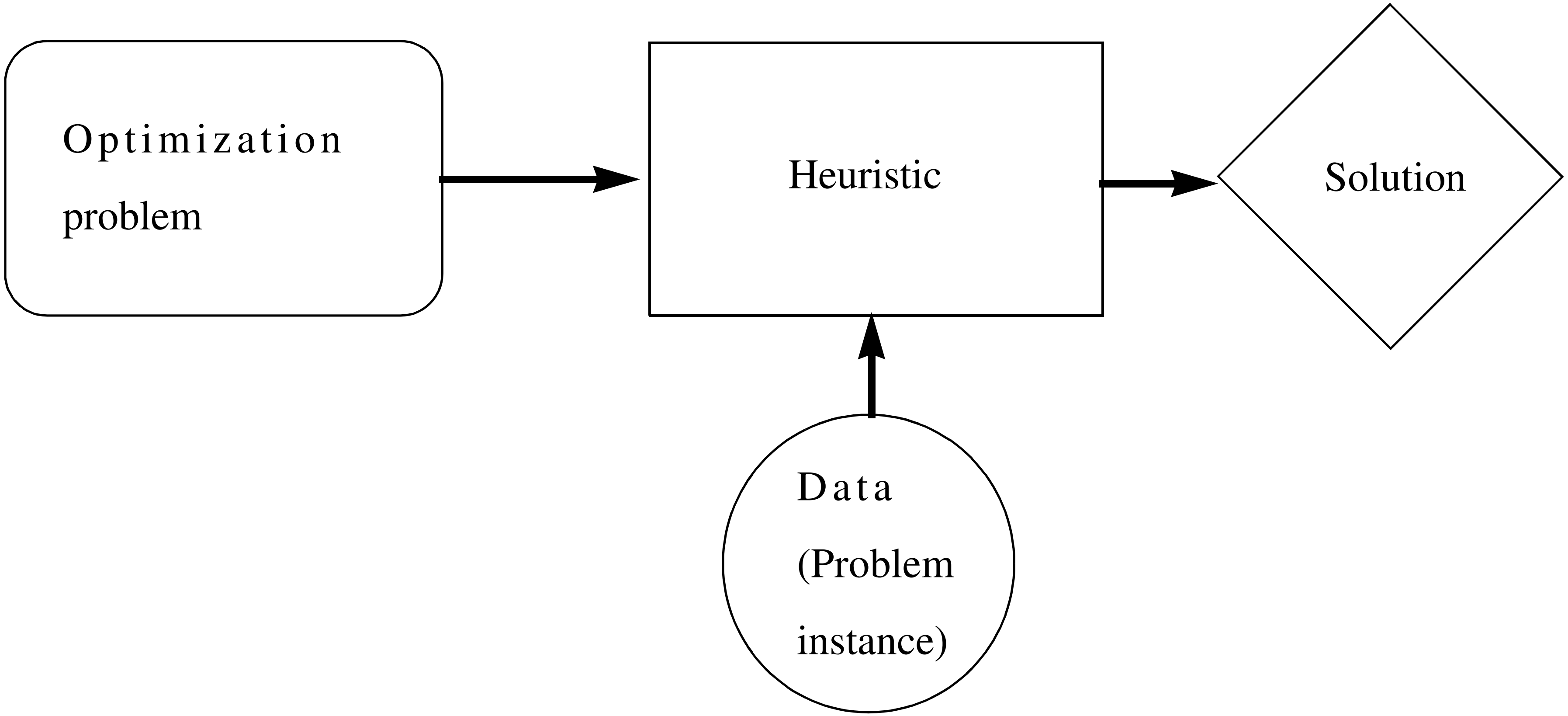
Large neighbourhood, Local OPTimization

Adaptive memory for VRPs

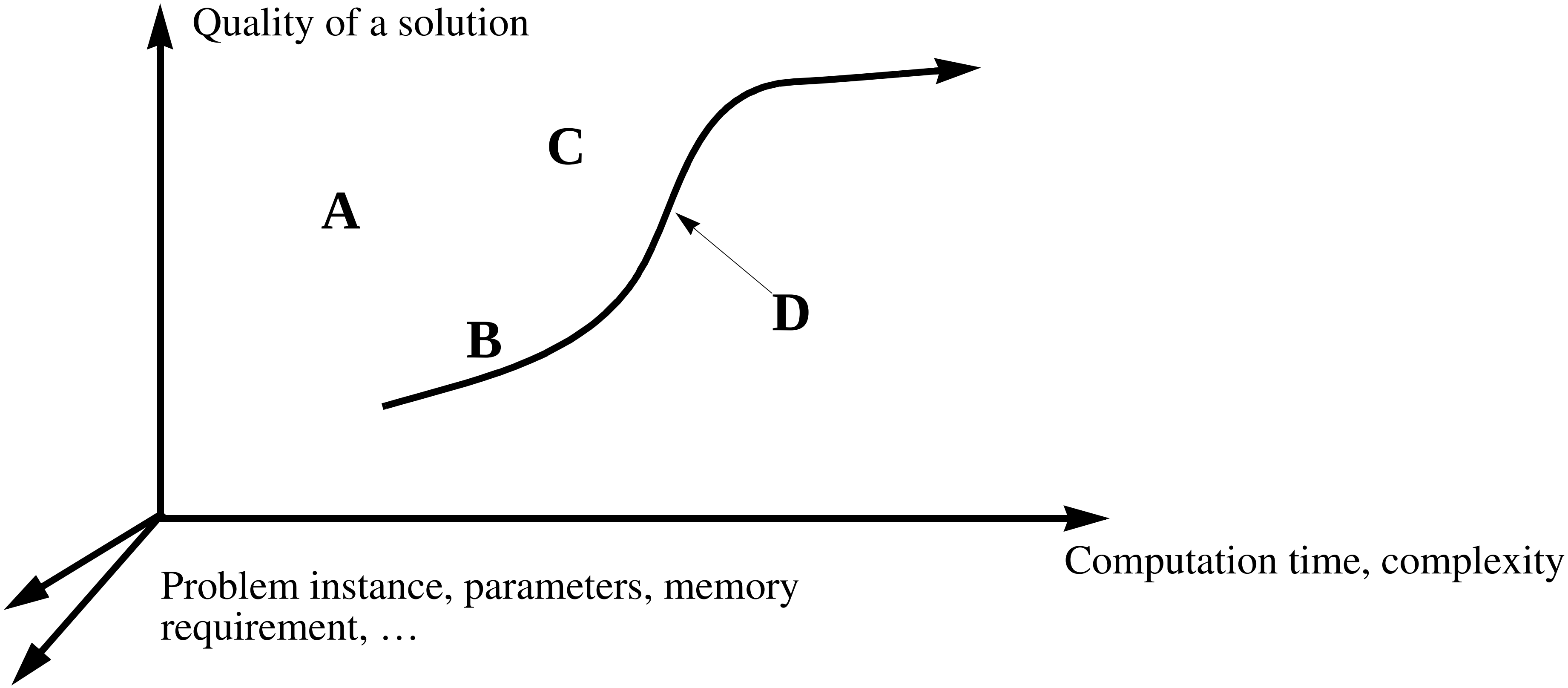
Dynamic problems

HEURISTIC :

Algorithm producing a solution non necessarily optimum for a generic combinatorial optimization problem.

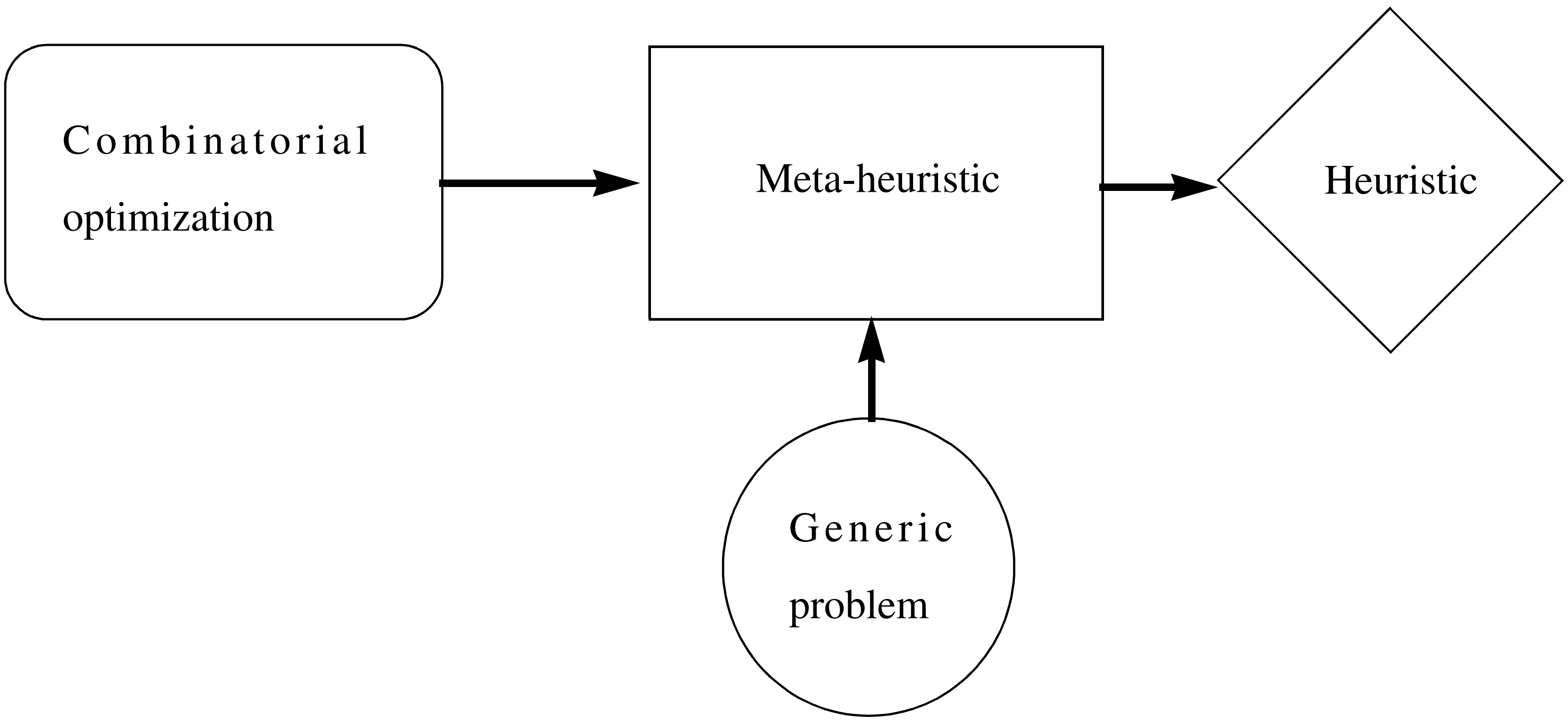


QUALITY OF A HEURISTIC



META-HEURISTIC :

Limited set of concepts that can be applied to a large set of combinatorial optimization problems for creating new heuristic methods.



QUALITY OF A META-HEURISTIC

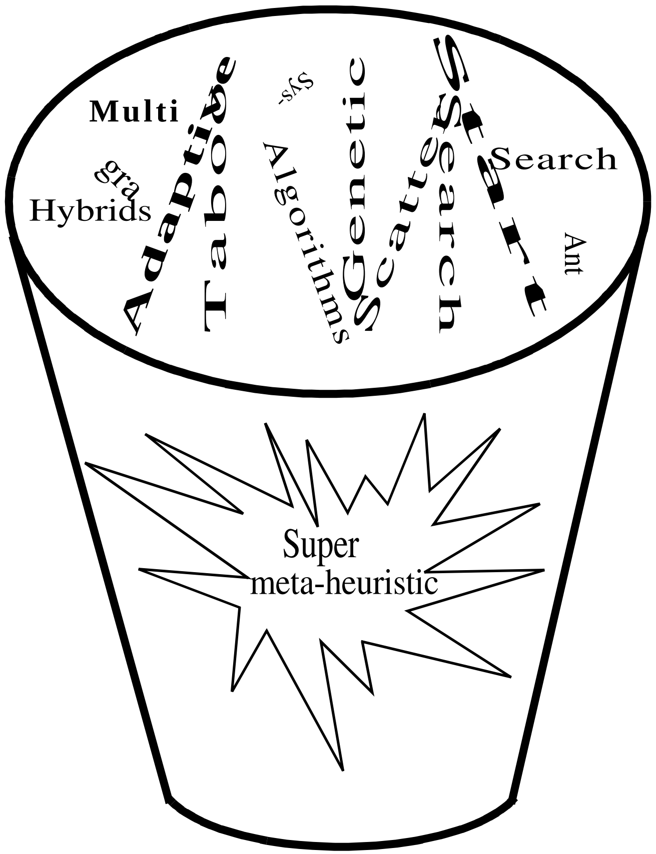
Can be applied to a large set of combinatorial optimization problems

Good heuristics suggesting power.

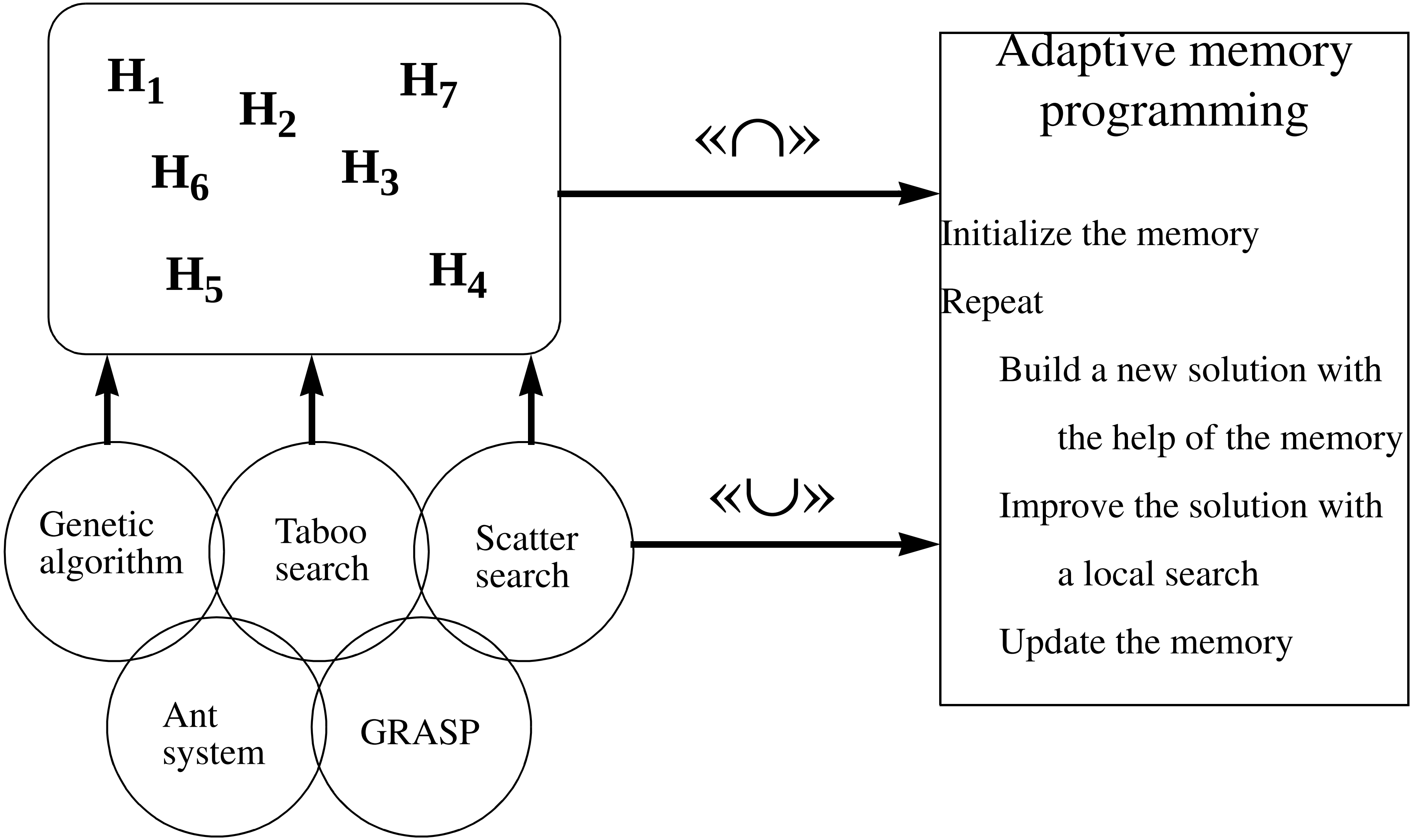
Simple design

Attractiveness of the metaphor

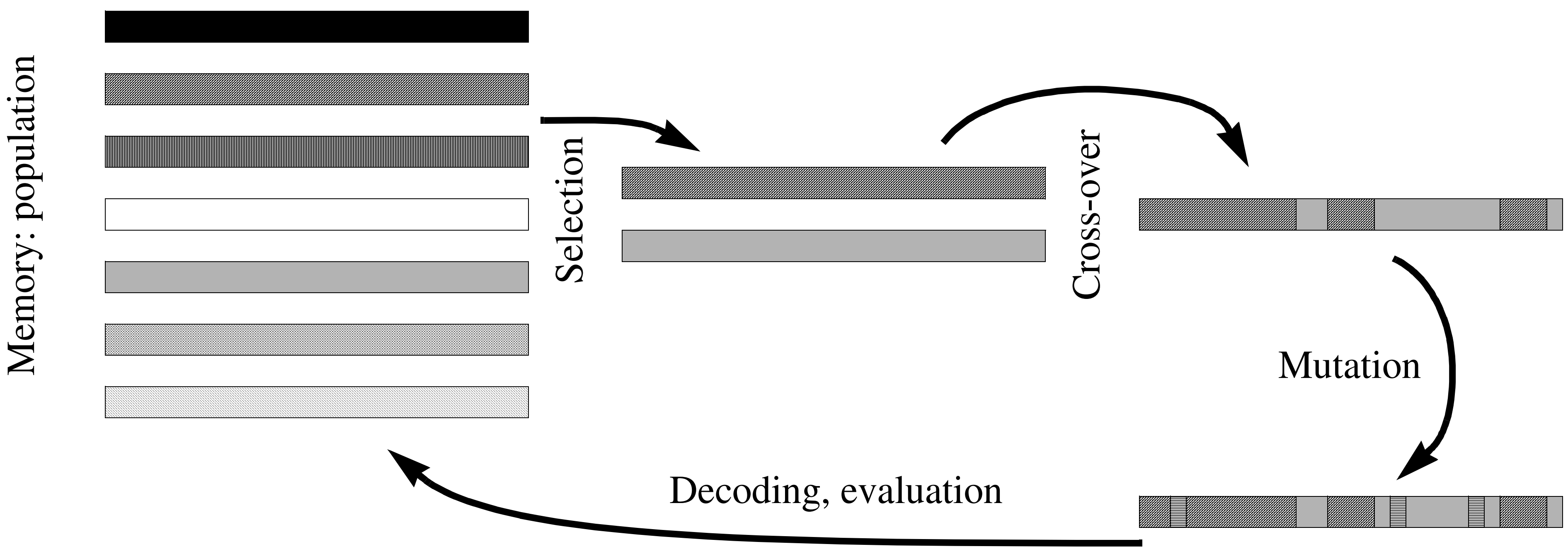
→ **Quality of the heuristic methods produced**



ADAPTIVE MEMORY PROGRAMMING



AMP I: Genetic algorithms



Main strength: Good cover of the solution space

Weaknesses:

Coding sensitive

→ Natural coding, “intelligent” cross-over

No local search ⇒ Low convergence rate, low quality of final solutions

→ Mutation = Local search

AMP II: TABOO SEARCH

Long term memory:

Very often embedded in the local search.

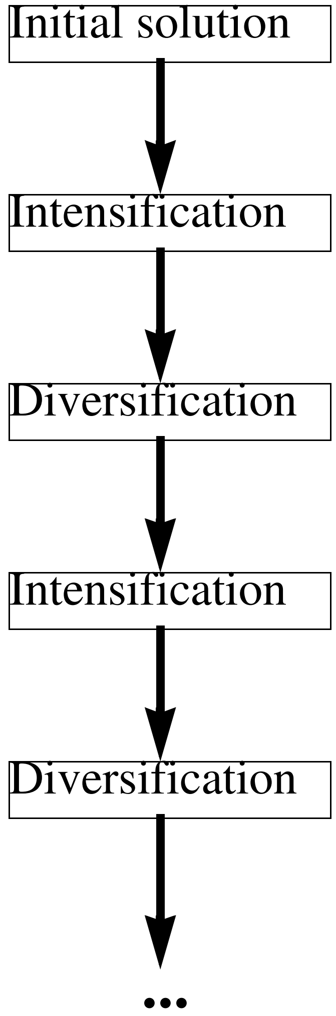
Used for a given number of iterations for jumping in solution space.

Intensification

- Basic taboo search
- Come back to a good solution
- Modify local search parameters
- Candidate list

Diversification

- Modify the objective
- Modify the neighbourhood
- Modify local search parameters



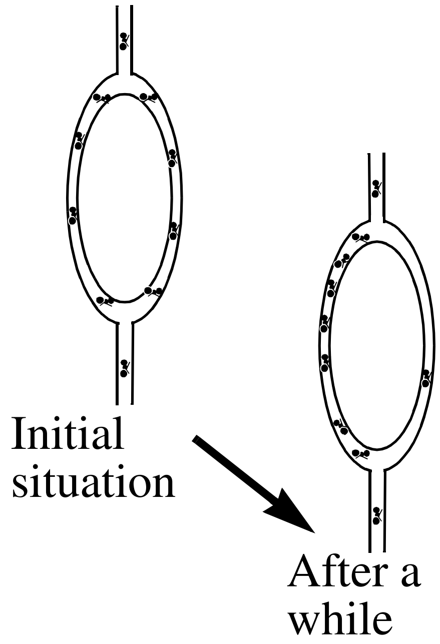
AMP III : ANT SYSTEM

Memory: Statistic on the use of given components in the solutions generated (pheromone trail).

(Parallel) construction of solution in a probabilistic way.

Statistic initially proposed (i : component of solution s, of quality q, $0 < \alpha < 1$ and β: parameters)

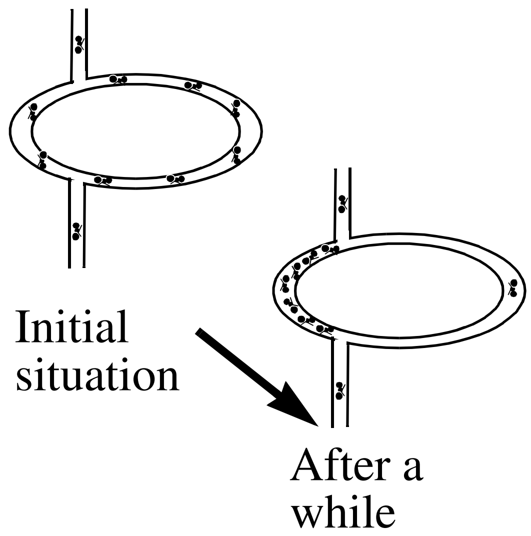
$$\tau_i(t+1) = \begin{cases} \alpha \cdot \tau_i(t), & (i \notin s) \\ \alpha \cdot \tau_i(t) + \beta \cdot q, & (i \in s) \end{cases}$$



Weakness: Very low convergence rate, local optimality not guaranteed.

Adaptation:

- Add a local search
- Co-ordination process
    - Selected re-enforcement
    - Search strategies



ANT SYSTEMS

Queen process

Initialize the memory *M*

Repeat (in parallel):

- Choose the parameters and activate an Ant process
- Receive a solution from an Ant process and update *M*

Return best solution found

Ant process

Build a solution *s*’ (with the help of *M* and parameters)

Apply a local search on *s*’ to obtain *s*

Return *s* to Queen process

Remark: View very close to AMP, allowing to integrate components of other meta-heuristics, not too far from the metaphor

Flow shop sequencing

n Jobs

m Machines

Processing time of job i on machine j: p_{ij} $1 \le i \le n$, $1 \le j \le m$

Objective:

$$
\begin{aligned}
&\mathrm{Min}_{\pi \in \Pi} && t_{\pi_n m} \\
&\text{s.t.} && t_{\pi_i j} \ge \max(t_{\pi_{i-1} j}, t_{\pi_i j-1}) + p_{\pi_i j},\ i, j > 1 \\
& && t_{\pi_1 j} \ge t_{\pi_1 j-1} + p_{\pi_1 j},\ j > 1 \\
& && t_{\pi_i 1} \ge t_{\pi_i 1} + p_{\pi_i 1},\ i > 1 \\
& && t_{\pi_1 1} \ge p_{\pi_1 1}
\end{aligned}
$$

LOCAL SEARCH FOR FLOW SHOP

Neighbourhood: Double shift

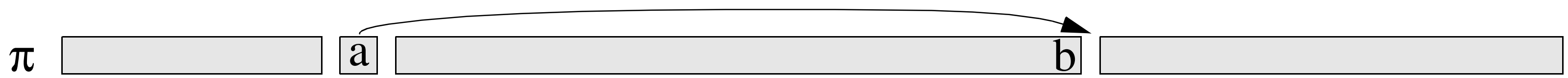
Change the position of one job $\pi \rightarrow \mu$ (job in position a goes in position b).

Forward ($a < b$)

$$\mu_k = \pi_k,\ k < a,\ k > b$$

$$\mu_k = \pi_{k+1},\ a < k \le b$$

$$\mu_b = \pi_a$$

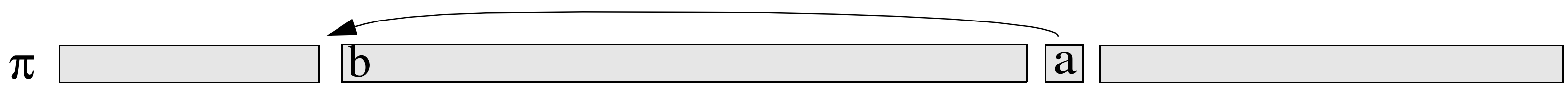


Backward ($a > b$)

$$\mu_k = \pi_k,\ k > a,\ k < b$$

$$\mu_k = \pi_{k-1},\ b < k \le a$$

$$\mu_b = \pi_a$$



Other neighbourhoods

Adjacent pairwise exchange

Inversion of a sub-part of the permutation (TSP 2-opt)

Exchange the position of two jobs

Forward shift

Backward shift.

Empirical studies (Taillard 1990, Reeves 1999):

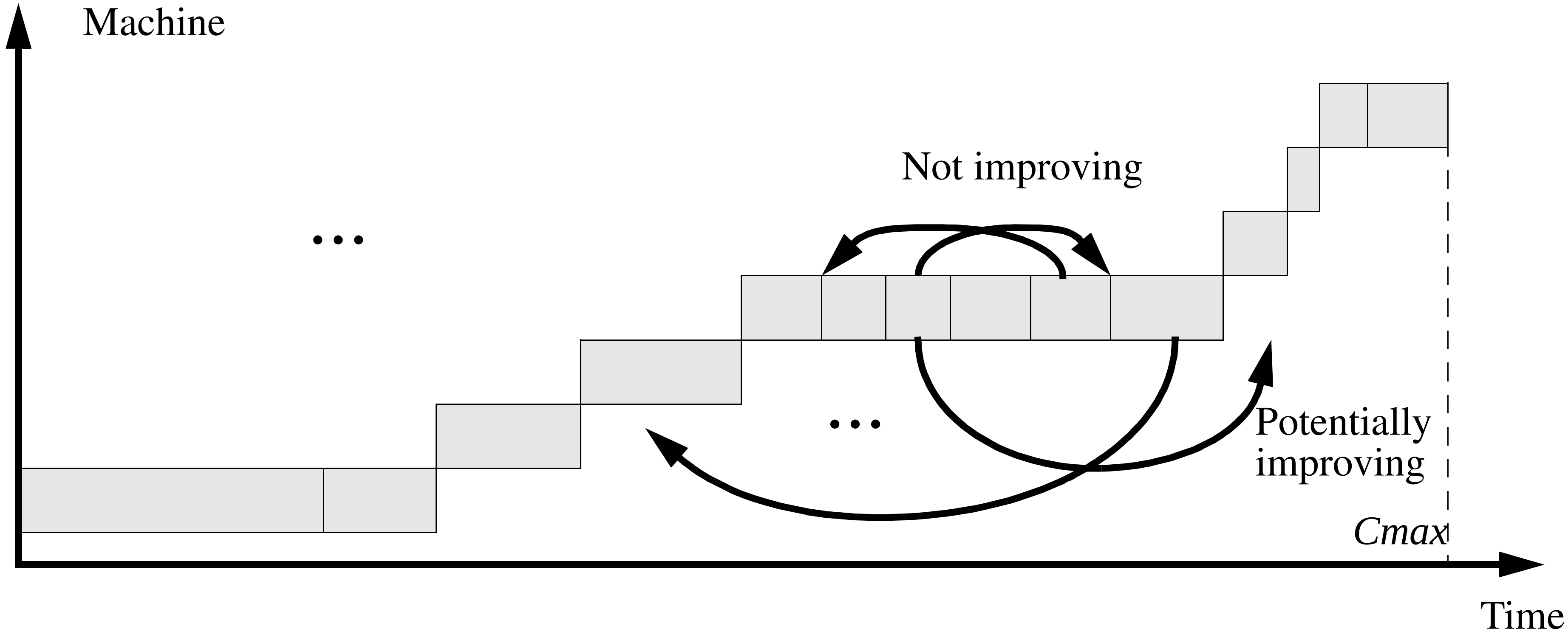
- Best: Double shift
- Then: exchange
- Then: one-direction shift
- Worst: Adjacent pairwise exchange.

EVALUATION OF THE NEIGHBOURHOOD

Complexity: $O(n^2m)$ (Taillard 1990)

Improvement: Nowicki & Smutnicki 1996

Restriction to potentially interesting shifts + further restrictions + computational improvements



LOCAL SEARCH: BASIC TABOO SEARCH

Taboo list: Hashing table of size L ($\cong$ 100'000).

Hashing function: $\sum \pi_i \cdot \pi_{i+1}$

Taboo condition: Prevent to come back to a solution with hashing value already computed in previous iterations (few% of the solutions forbidden).

For relatively small number of iterations

AMP FOR FLOW SHOP: FANT

Memory: Statistics τ_{jk}: (job k succeeds job j).

Constructive procedure: random with probability proportional to τ_{jk}.

Memory update:

$$\tau_{\pi_j \pi_{j+1}} = \tau_{\pi_j \pi_{j+1}} + r,\ 1 \le j < n$$

$$\tau_{\beta_j \beta_{j+1}} = \tau_{\beta_j \beta_{j+1}} + B,\ 1 \le j < n$$

π: current solution

β: best solution found

r: initially = 1; incremented if the constructive procedure produces β again, set to 1 if β improved.

B: fixed parameter (= n)

Memory reinitialized (τ_{jk} = r) each time r changes

Local search procedure: basic taboo search running for 4n iterations.

AMP for flow shop: Genetic hybrid

Memory: Population of P (= 20) solutions (survival of the fittest strategy)

Constructive procedure:

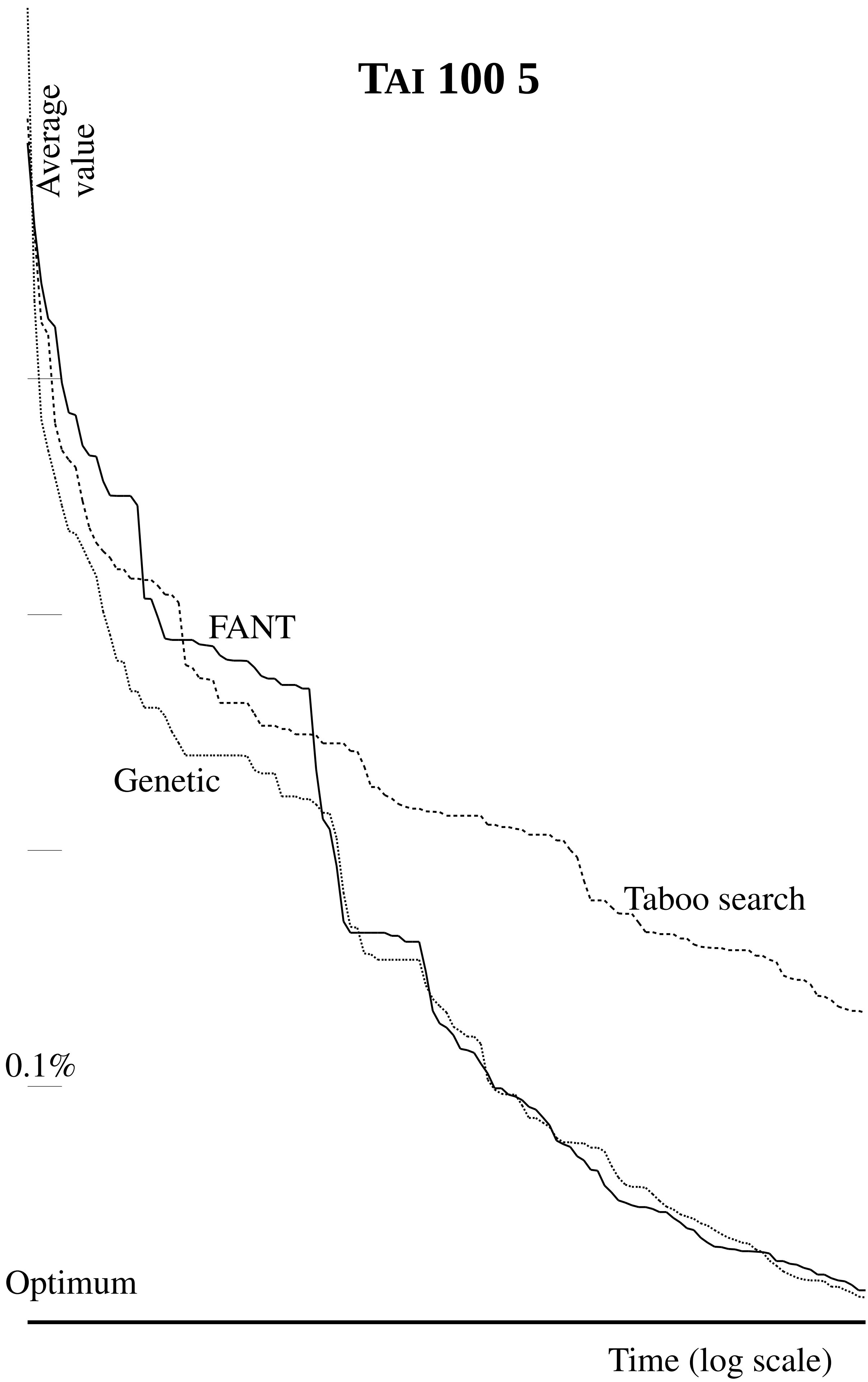
Selection: linear ranking

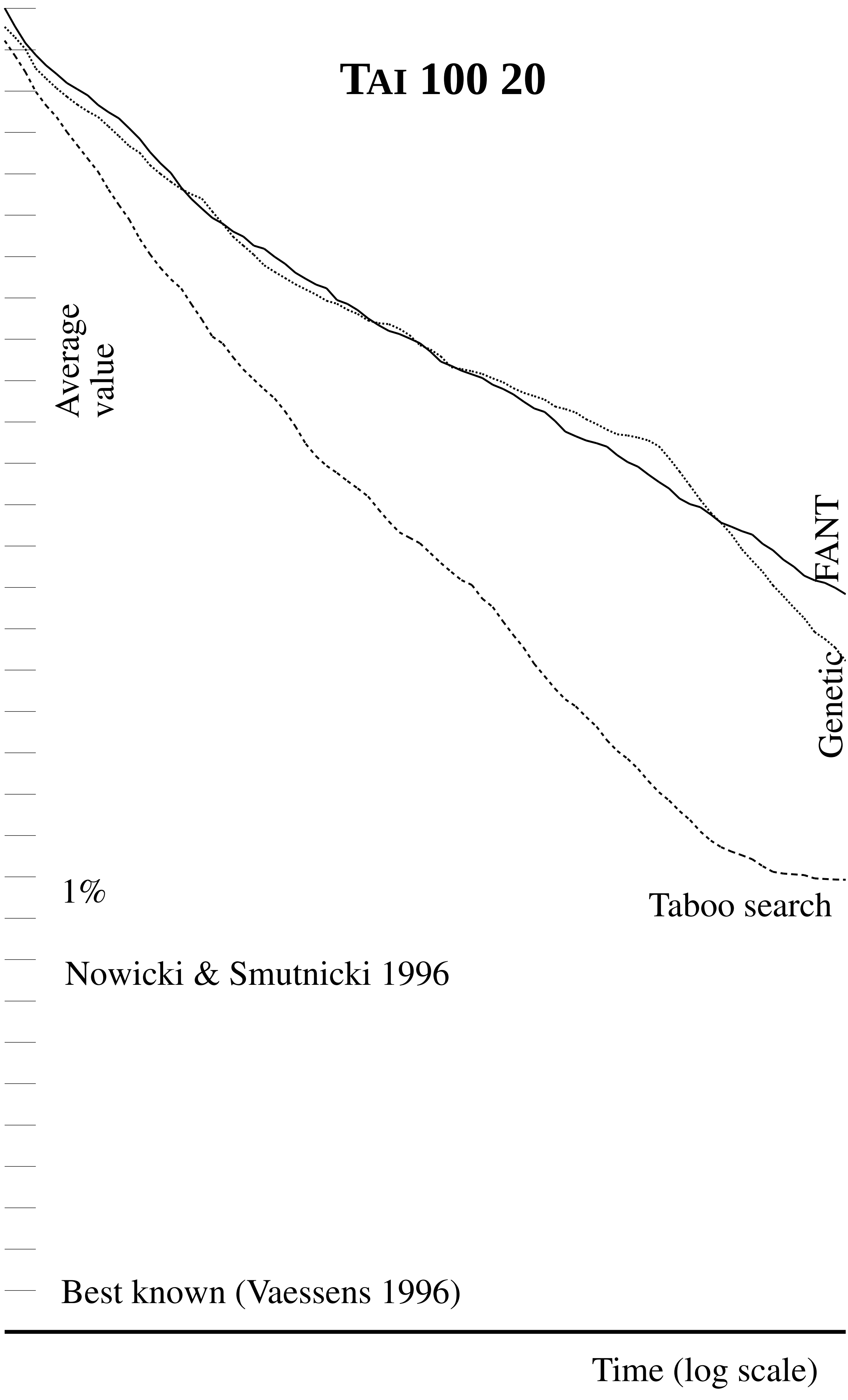
$$P(\text{choose } k^{\text{th}} \text{ best individual}) = 1 - \frac{2k}{P(P-1)}$$

Cross-over (Tate & Smith, 1995):

1	2	3	4	5	6	7	8	9	10	11	12	First parent
2	1	3	5	7	6	4	10	11	9	8	12	Second parent
2	1	3	4	7	6		8	11	9		12	Rancom copy
						10				5		Random fill
2	1	3	4	7	6	10	8	11	9	5	12	Child

“Mutation”: Basic taboo search running for $4n$ iterations.


TAI 100 5
Average value
FANT
Genetic
Taboo search
0.1%
Optimum
Time (log scale)

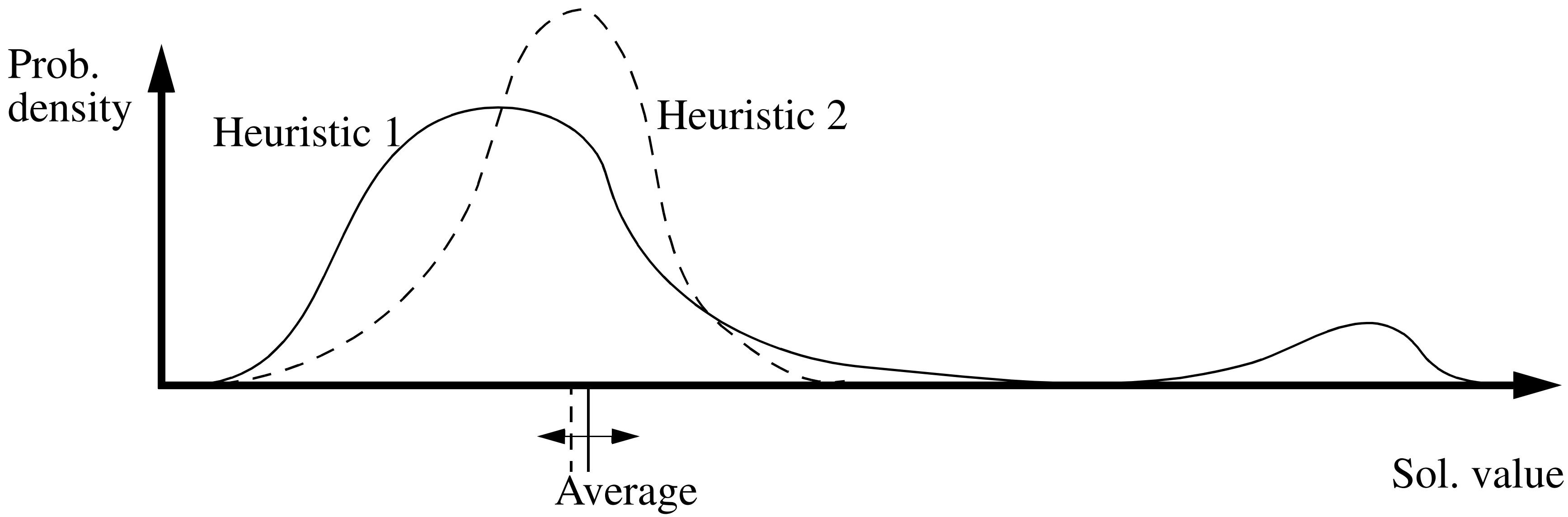

TAI 100 20
Average value
FANT
Genetic
1%
Taboo search
Nowicki & Smutnicki 1996
Best known (Vaessens 1996)
Time (log scale)

QUESTIONS :

Genetic paradigm better or worse than ant one ?

Is the difference significant ? Statistical test ?

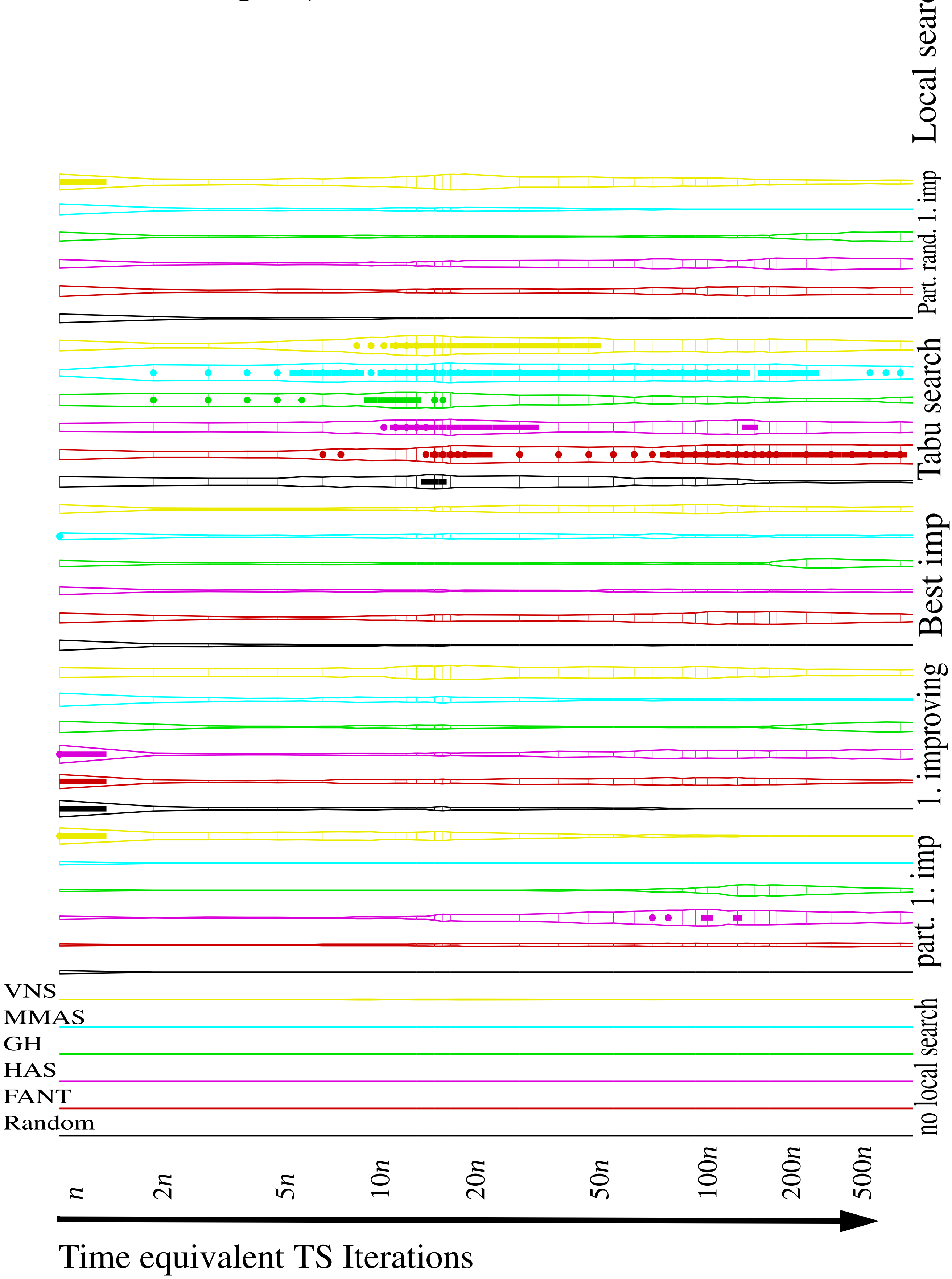
Influence of learning mechanism versus local search ?



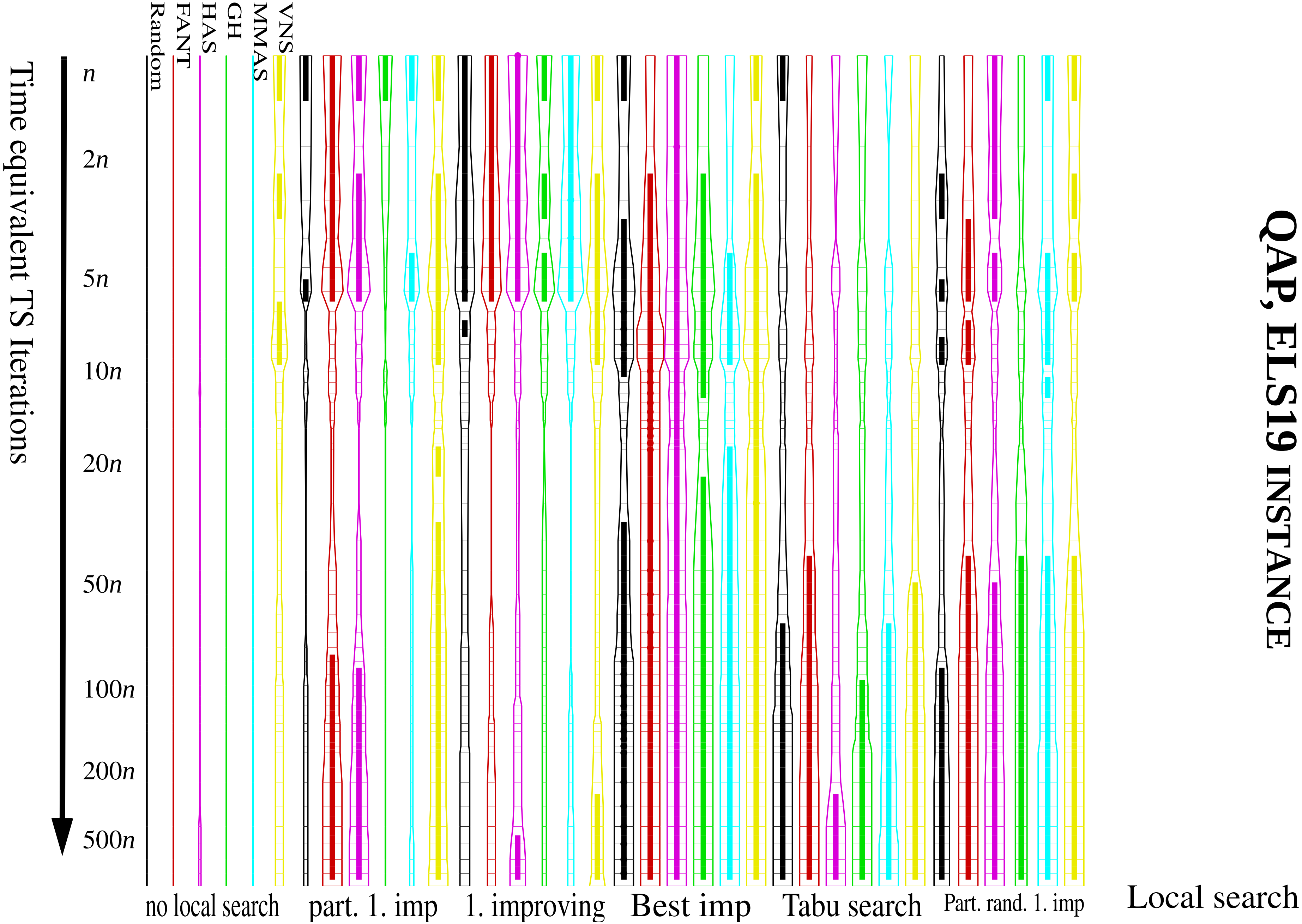
Answer trial : Non-parametric statistic tests, based on ranks

Mann-Withney : $P(H_1 < H_2) < 1/2$

QAP, SKO.. INSTANCES



QAP, ELS19 INSTANCE



CONCLUSIONS:

Local search mandatory (if good solutions are wanted)

Local search has a stronger influence than construction procedure

Memory important at long term for finding problem structure

Efficiency vastly depends on the instance

Instance over-fitting

VEHICLE SCHEDULING

1 depot (0)

n customers, with ordered quantities q_i, service times s_i, time windows $[b_i, e_i]$, ($0 \leq i, j, \leq n$)

Travel times t_{ij} between customers (depot) i and j ($0 \leq i, j, \leq n$)

Vehicles of capacity Q.

Solution: Set of vehicle tours $\{T_1, T_2, \ldots, T_m\}$ for the vehicles, specified by arrival hours a_i and departure hours d_i such that:

$$\sum_{i \in T_k} q_i \leq Q, \forall k$$

$$b_i \leq a_i \leq e_i, d_i \geq a_i + s_i, \forall \mathrm{i}$$

$$a_j \geq d_i + t_{ij}, \forall (i, j) \in T_k$$

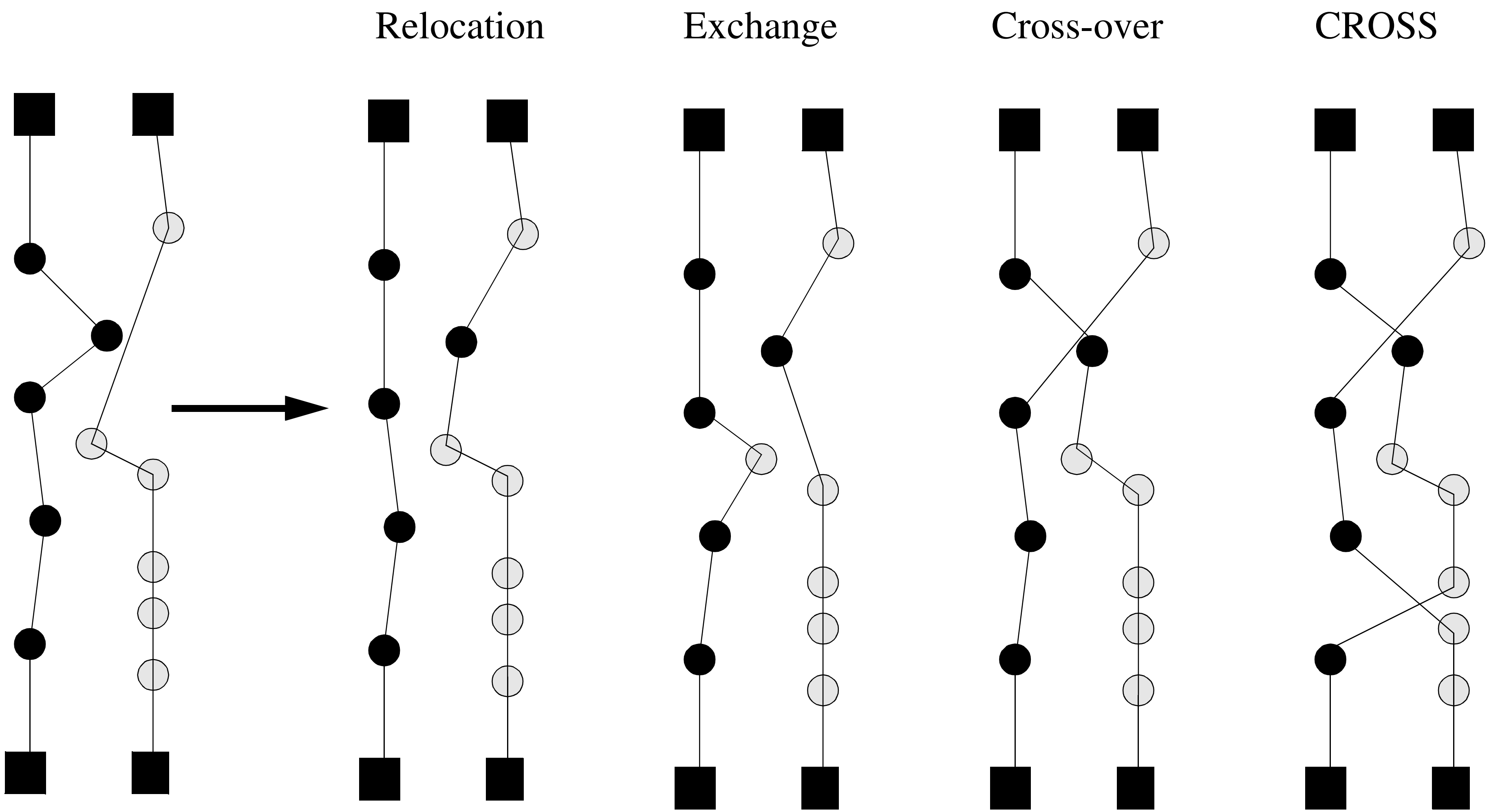
Objectives:

Minimize m

Minimize sum of travel time

Minimize sum of tour duration

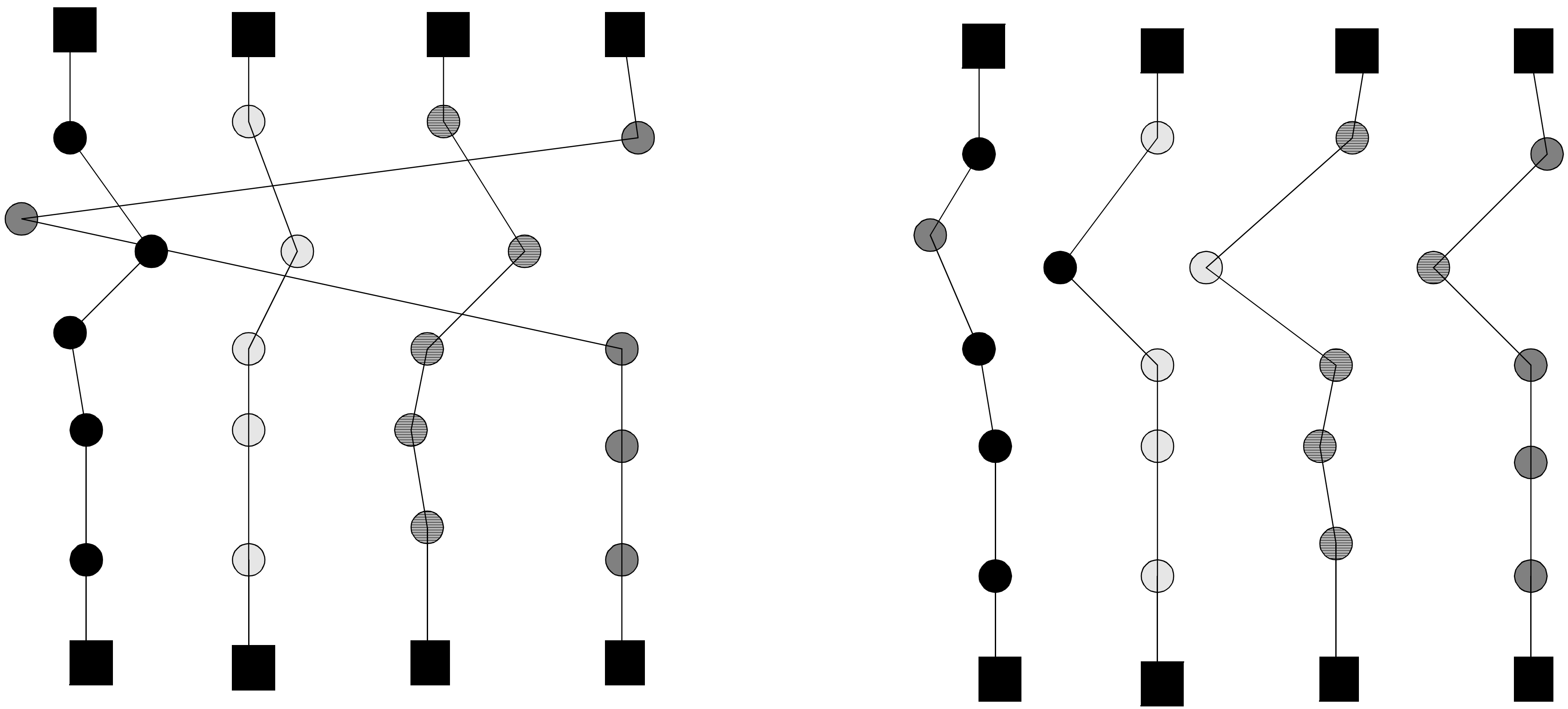
NEIGHBOURHOODS FOR VRPS



Generally good for small instances and basic local searches.

ADVANCED NEIGHBOURHOODS

Ejection chains (Rego & Roucairol, 1996)



Large neighbourhoods

Taillard (1993)

Choose k tours, optimize them with a basic taboo search.

Shaw (1998)

Choose k customers, find optimum insertion with constraint programming

GENERAL LOPT FRAME

0) Input : Solution s composed of parts $s_1, \ldots, s_p$.

1) $C = \{s_1, \ldots, s_p\}$

2) While $C \neq \emptyset$, repeat :

   2a) Choose $s_i \in C$; let sub-problem R composed of the set of the r "closest" sub-problems of s_i ($s_i \in R$)

   2b) Optimize R

   2c) If R improved, set $C \leftarrow C \cup R$;
   else, set $C \leftarrow C \setminus \{s_i\}$.

s_i: Customers, Vehicle tours, clusters

2b: Exact algorithm, taboo search

2a, 2c: Search strategy based on a candidate list

AMP FOR VRPs

Memory M: Population of tours

Construction procedure

a) Give a rank $r_T > 0 \ \forall T \in M$; $s \leftarrow \varnothing$; $M' \leftarrow M$.

b) Repeat, while $M' \neq \varnothing$:

b1) Choose a tour $T \in M'$, with probability $r_T / \sum_{k \in M'} r_k$.

b2) Set $s \leftarrow s \cup \{T\}$.

b3) For each tour $T' \in M'$ such that $T \cap T' \neq \varnothing$ set $M' \leftarrow M' \setminus \{T'\}$.

c) Complete partial solution s

Local search

Large neighbourhood with basic taboo search based on relocation + exchange

Large neighbourhood with basic taboo search based on CROSS

Question: Inspired by genetic algorithms, scatter search, taboo search, ant systems ?

Remark: Also efficient for solving other problems (e. g. Graph colouring, Zufferey 1998)

Other VRPs tackled with AMP

1) **Multiple tours VRP (with Gendreau & Laporte)**

   Run a standard AMP

   Solve a kind of bin-packing with tours in memory

2) **VRP with MinMax objective (with Golden & Laporte)**

   Set $L_{max} = \infty$

   Run an AMP with:

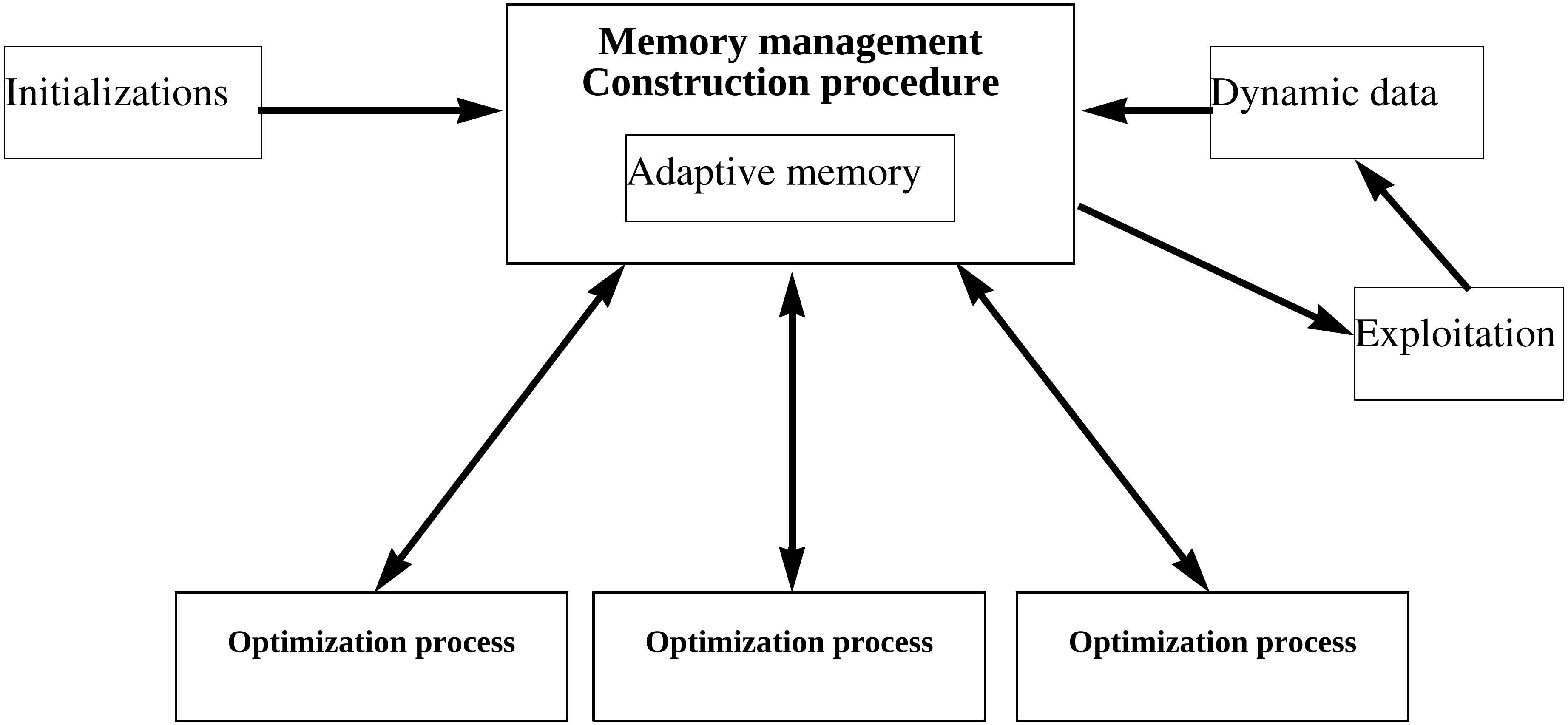
   - Memory containing only tours $< L_{max}$
   - Local search allowing only tours $< L_{max}$
   - Update L_{max} when a feasible solution is found

3) **VRP with heterogeneous fleet**

   Run a standard AMP for each vehicle type $1,\dots,k$

   Solve a set partitioning problem with additional constraints with tours in $M_1 \cup M_2 \cup \dots \cup M_k$

DYNAMIC VRP SOLVED WITH PARALLEL AMP



CONCLUSIONS

Analysis of memory-based methods based on meta-heuristics

Difficult to see the metaphor

Synthesis: Adaptive memory programming

Conceptually simple

Not restrictive

Easy to parallelize

Practical applications

Dynamic problems

Local search: First component to optimize, theoretical properties.

Memory: Useful for long runs, structured problems.