Carthagène

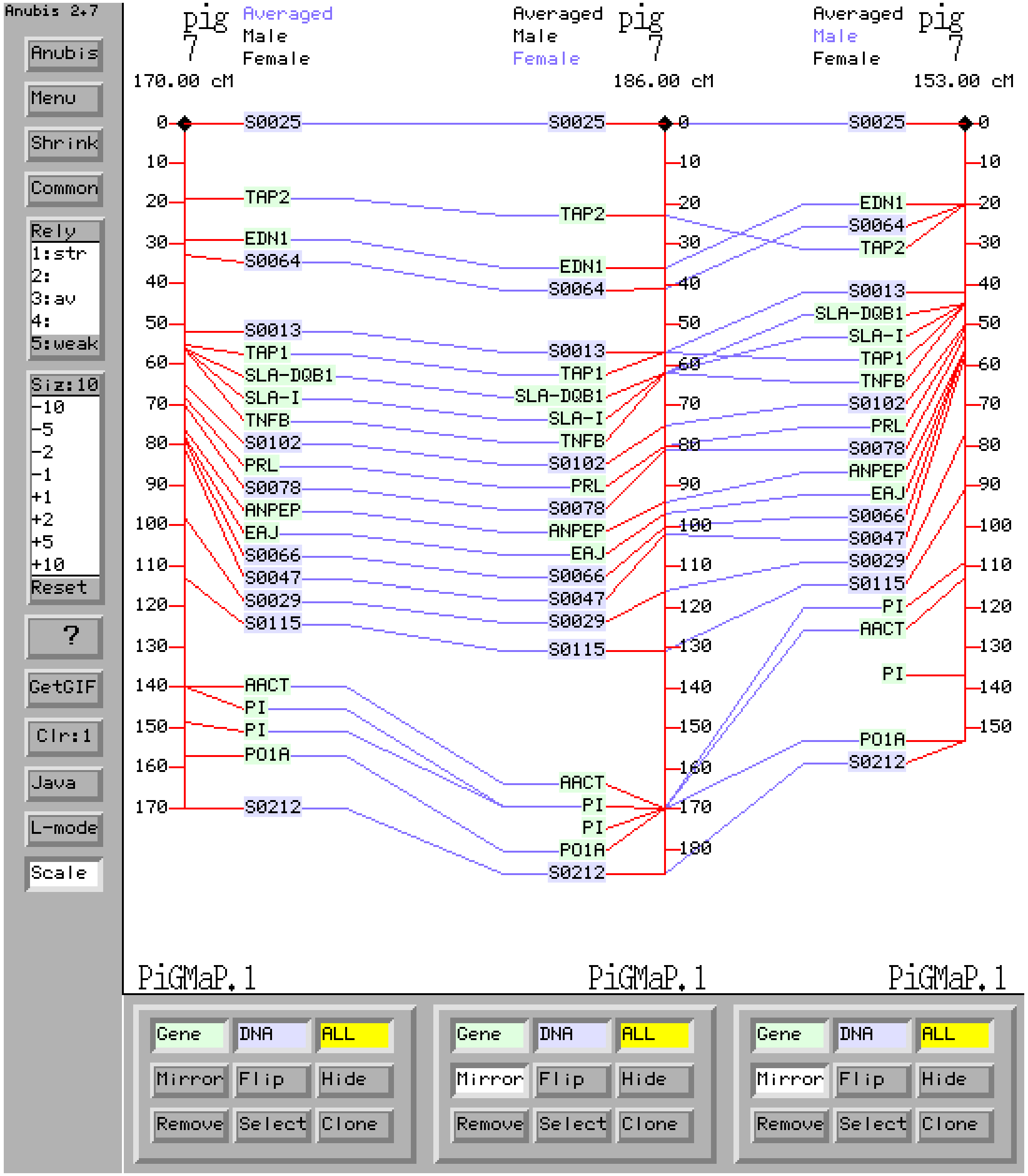
A brief introduction to combinatorial optimization:
The Traveling Salesman Problem

Simon de Givry

Find a tour with minimum distance, visiting every city only once

Distance matrix (miles)

Distances	**Camara**	**Caniço**	**Funchal**	**…**
Camara	0	15	7	…
Caniço	15	0	8	…
Funchal	7	8	0	…
…	…	…	…	…

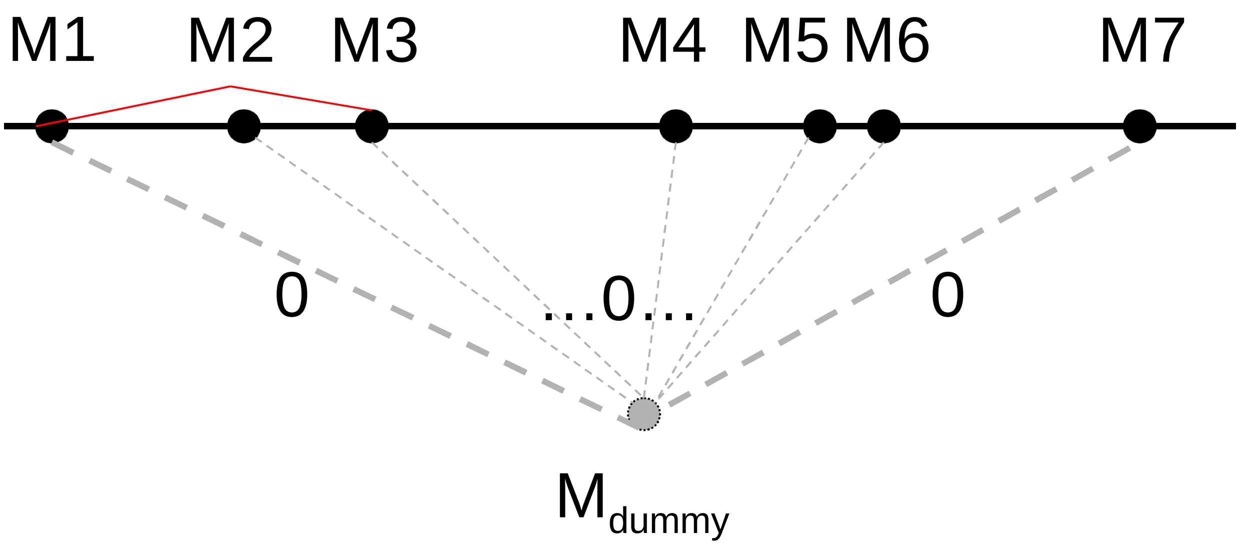


Find an order of all the markers with maximum likelihood

2-point distance matrix (Haldane)

Distances	**M1**	**M2**	**M3**	**…**
M1	0	14	7	…
M2	14	0	8	…
M3	7	8	0	…
…	…	…	…	…

Link



$\forall i,j,k$ distance(i,j) ? distance(i,k) + distance(k,j)

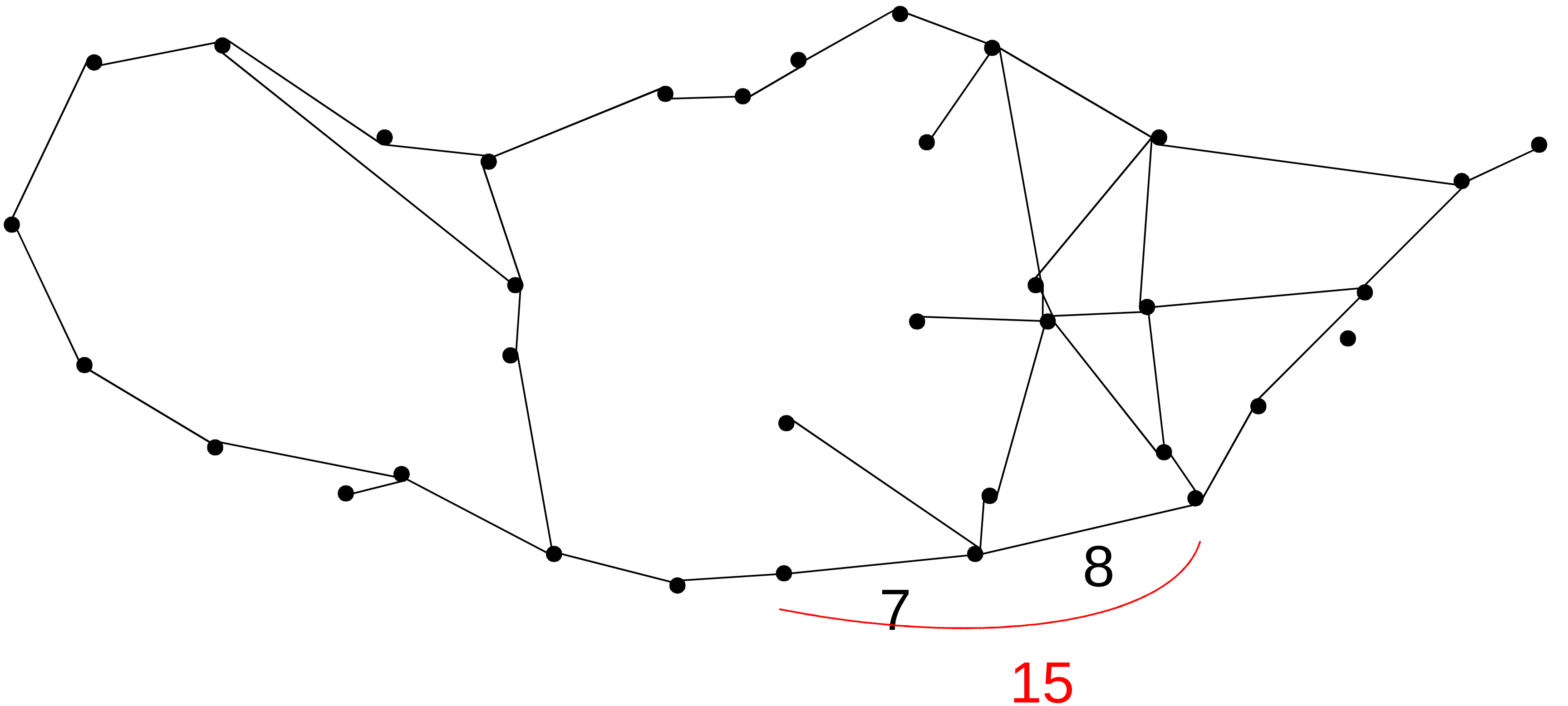
=,≤

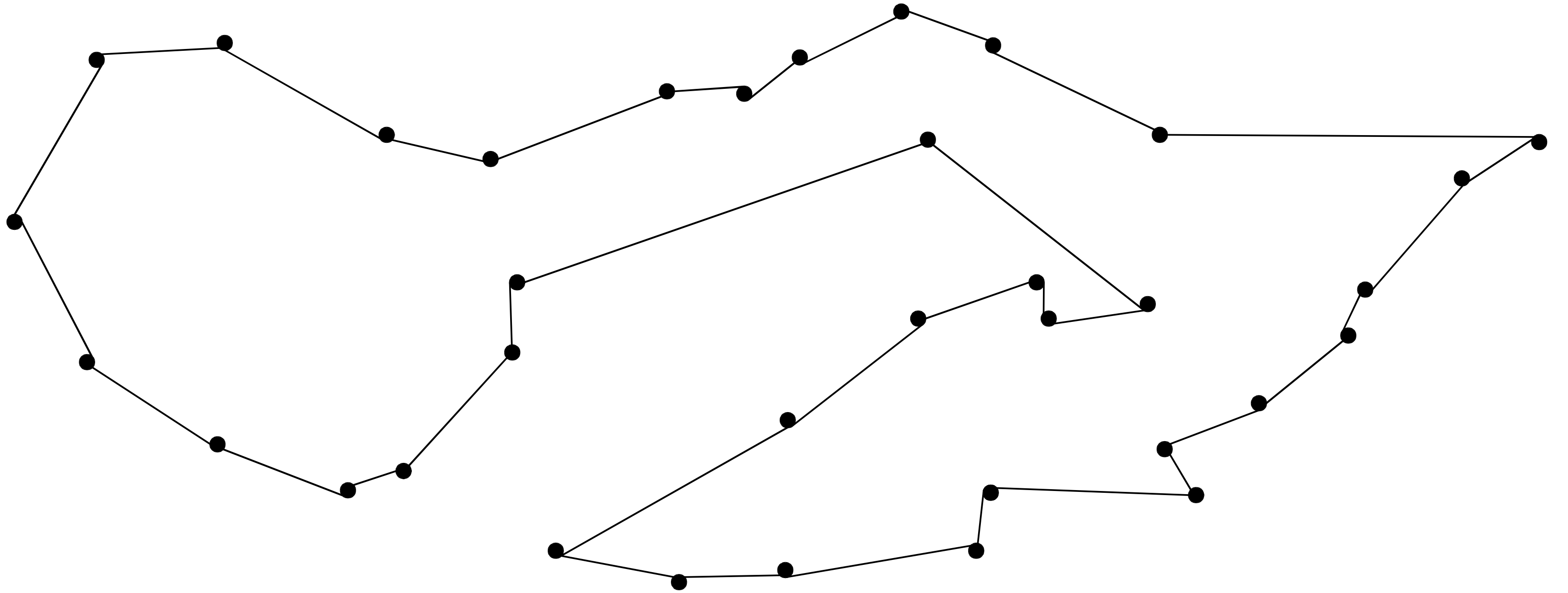
Multi-point likelihood (with unknowns) ⇔

the distance between two markers depends on the order

Traveling Salesman Problem

- Complete graph
- Positive weight on every edge
  - Symmetric case: dist(i,j) = dist(j,i)
  - Triangular inequality: dist(i,j) ≤dist(i,k) + dist(k,j)
  - Euclidean distance
- Find the shortest Hamiltonian cycle


8
7
15



Total distance = xxx miles

Traveling Salesman Problem

- Theoretical interest
  - NP-complete problem
  - 1993-2001: +150 articles about TSP in INFORMS & Decision Sciences databases
- Practical interest
  - Vehicle Routing Problem
  - Genetic/Radiated Hybrid Mapping Problem
    - NCBI/Concorde, Carthagène, ...

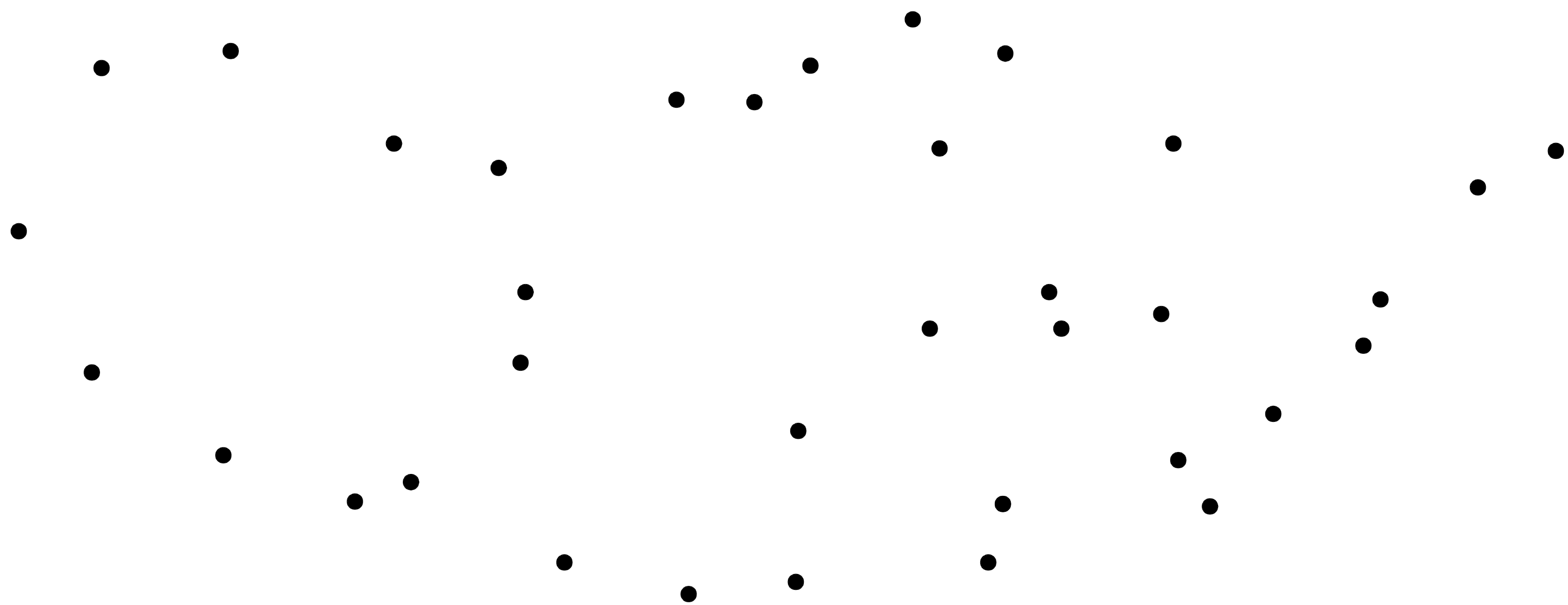
Variants

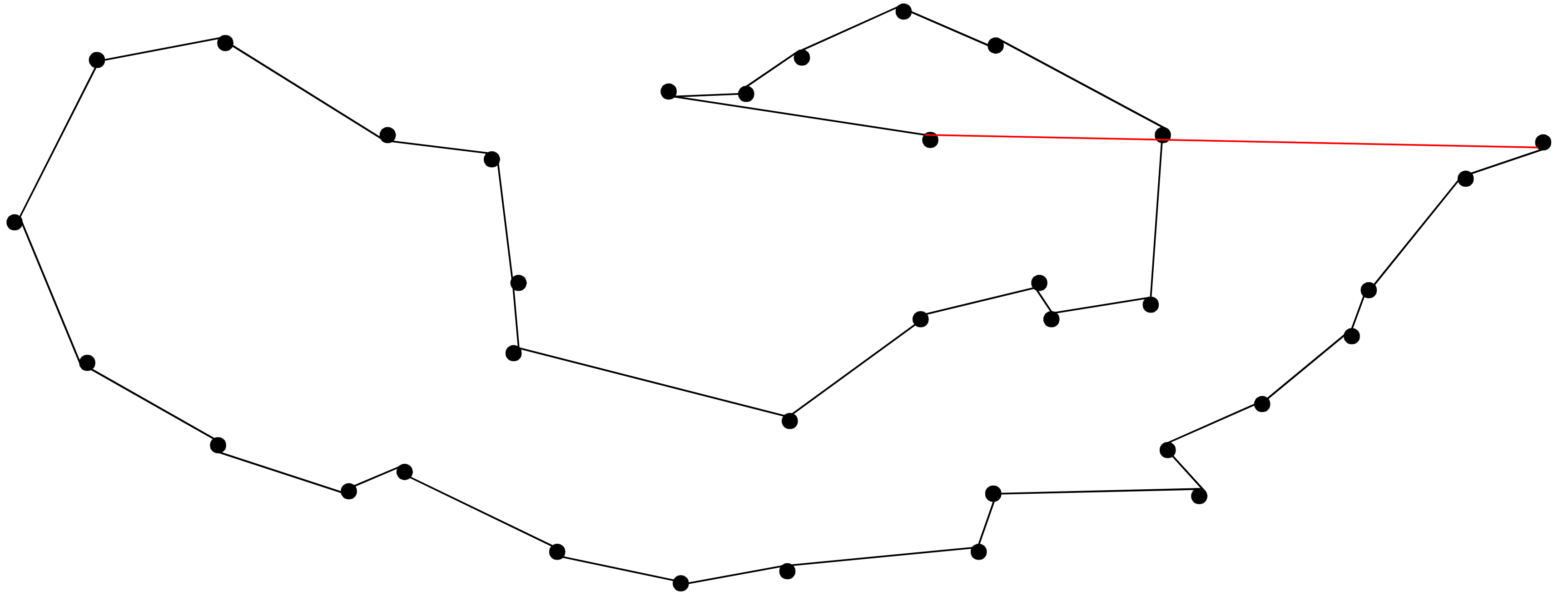
Euclidean Traveling Salesman Selection Problem
Asymmetric Traveling Salesman Problem
Symmetric Wandering Salesman Problem
Selective Traveling Salesman Problem
TSP with distances 1 and 2, TSP(1,2)
K-template Traveling Salesman Problem
Circulant Traveling Salesman Problem
On-line Traveling Salesman Problem
Time-dependent TSP
The Angular-Metric Traveling Salesman Problem
Maximum Latency TSP
Minimum Latency Problem
Max TSP
Traveling Preacher Problem
Bipartite TSP
Remote TSP
Precedence-Constrained TSP
Exact TSP
The Tour Cover problem

Plan

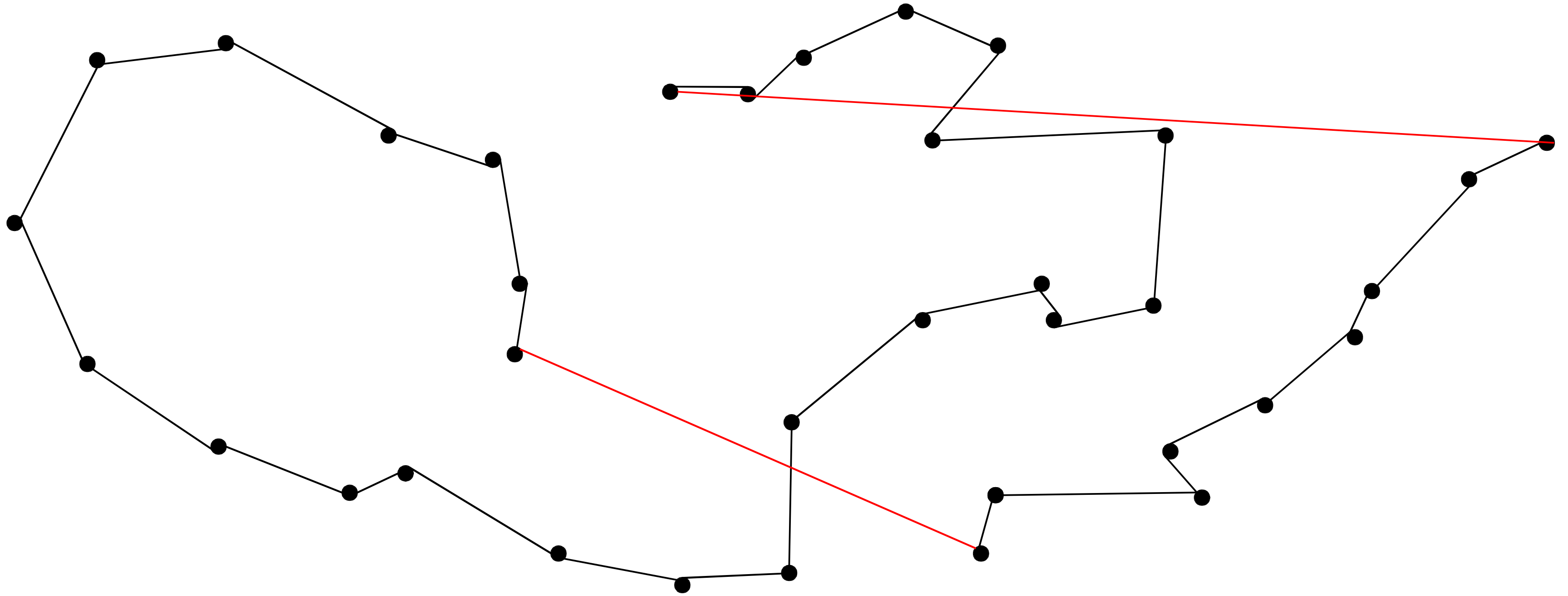
- Introduction to TSP
- Building a new tour
- Improving an existing tour
- Finding the best tour

Building a new tour

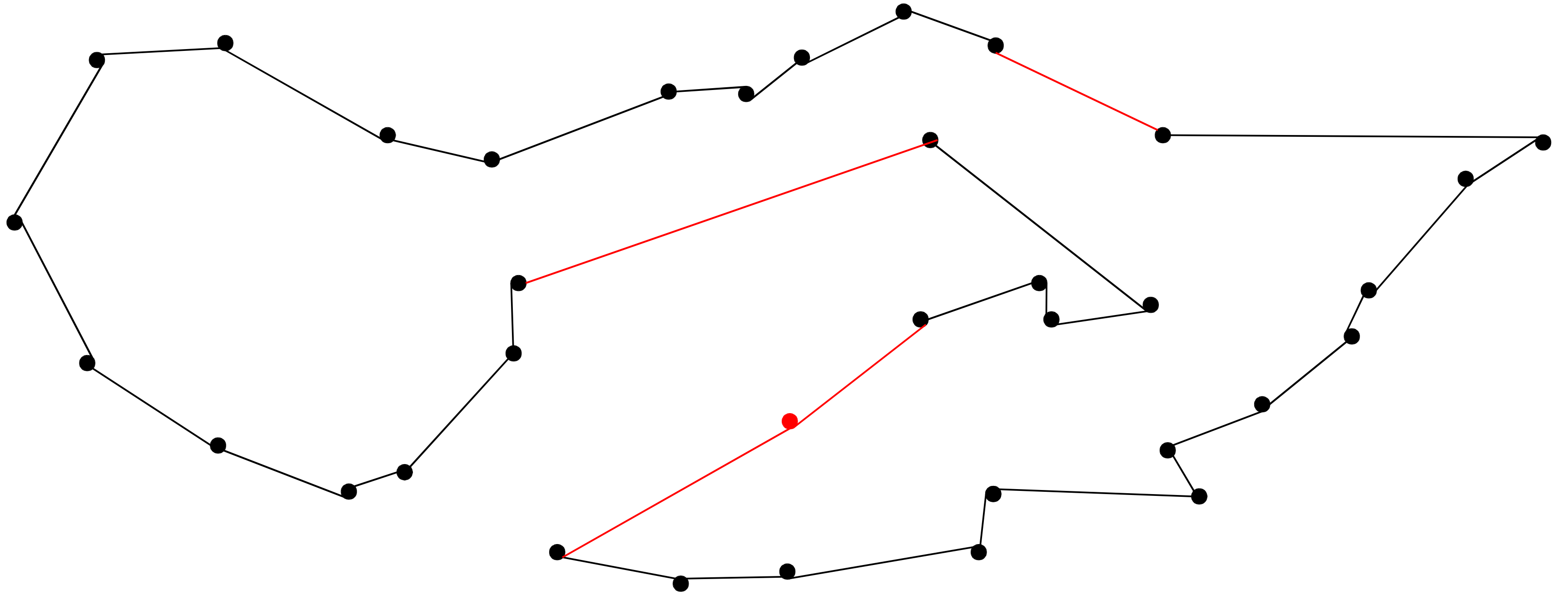




Nearest Neighbor heuristic



Greedy (or multi-fragments) heuristic

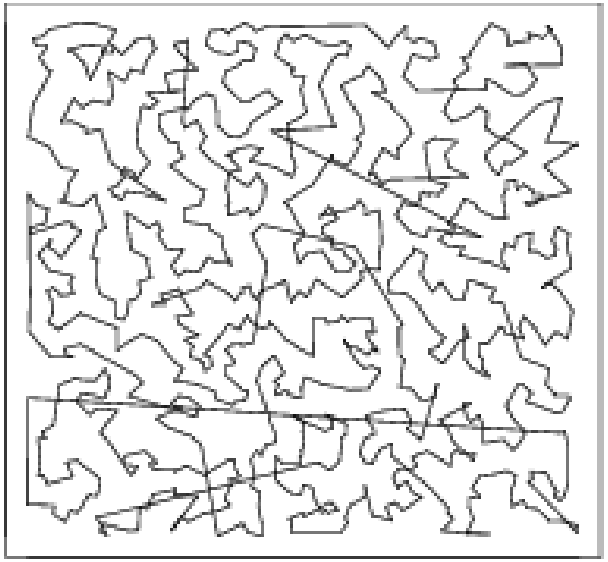


Savings heuristic (Clarke-Wright 1964)

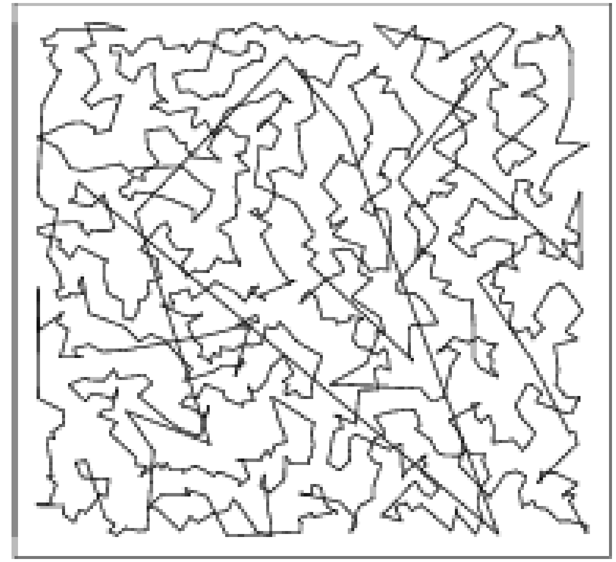
Heuristics

Mean distance to the optimum

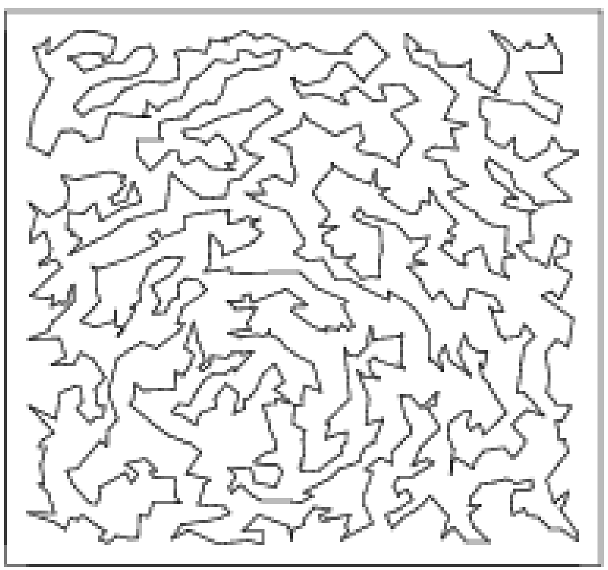
Savings:	11%
Greedy:	12%
Nearest Neighbor:	26%



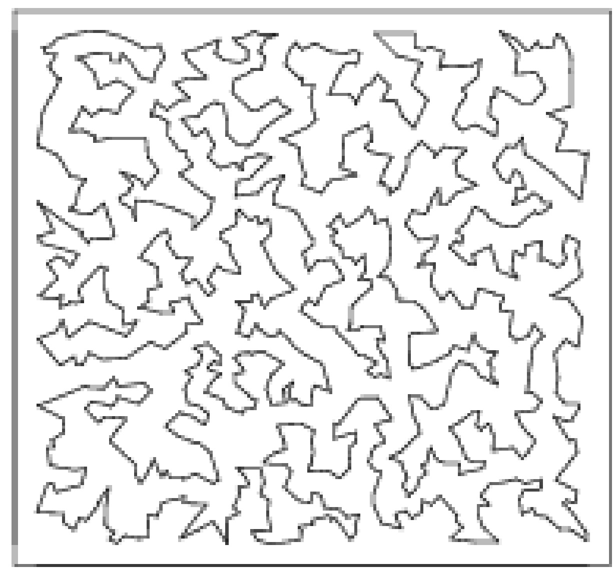
Greedy Tour



Nearest Neighbor Tour

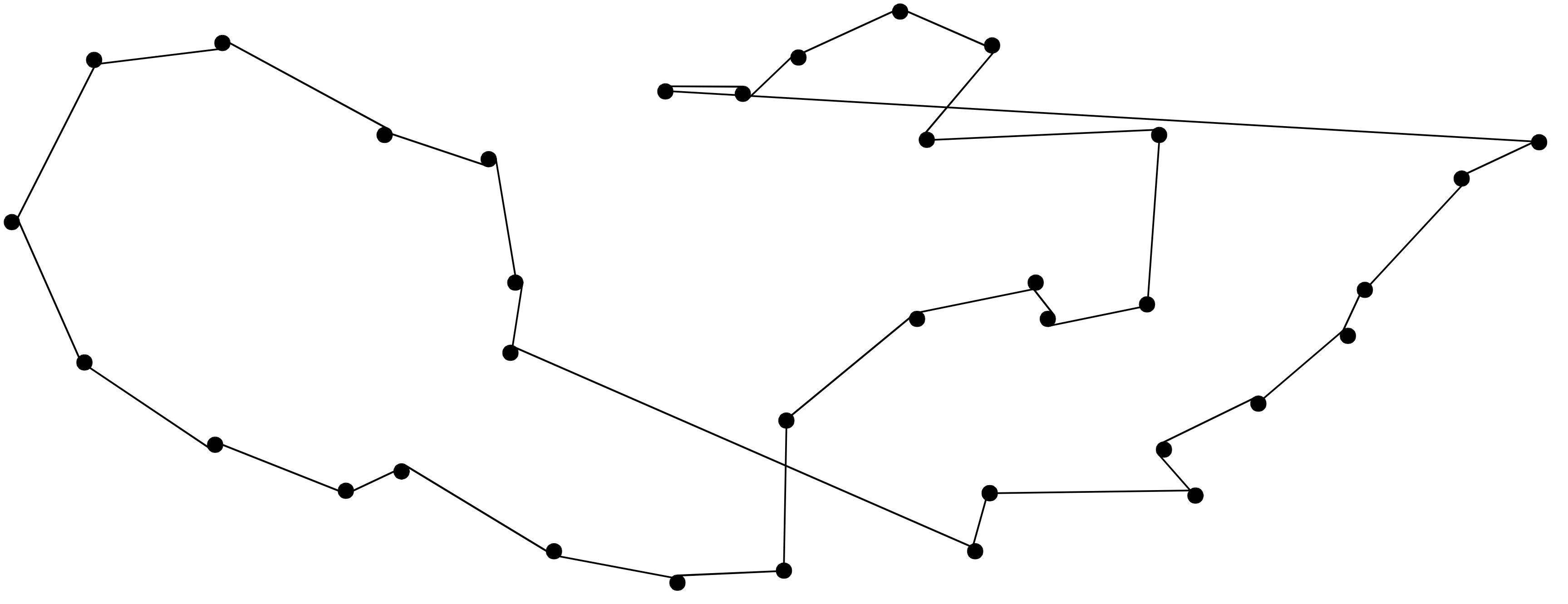


Savings Tour

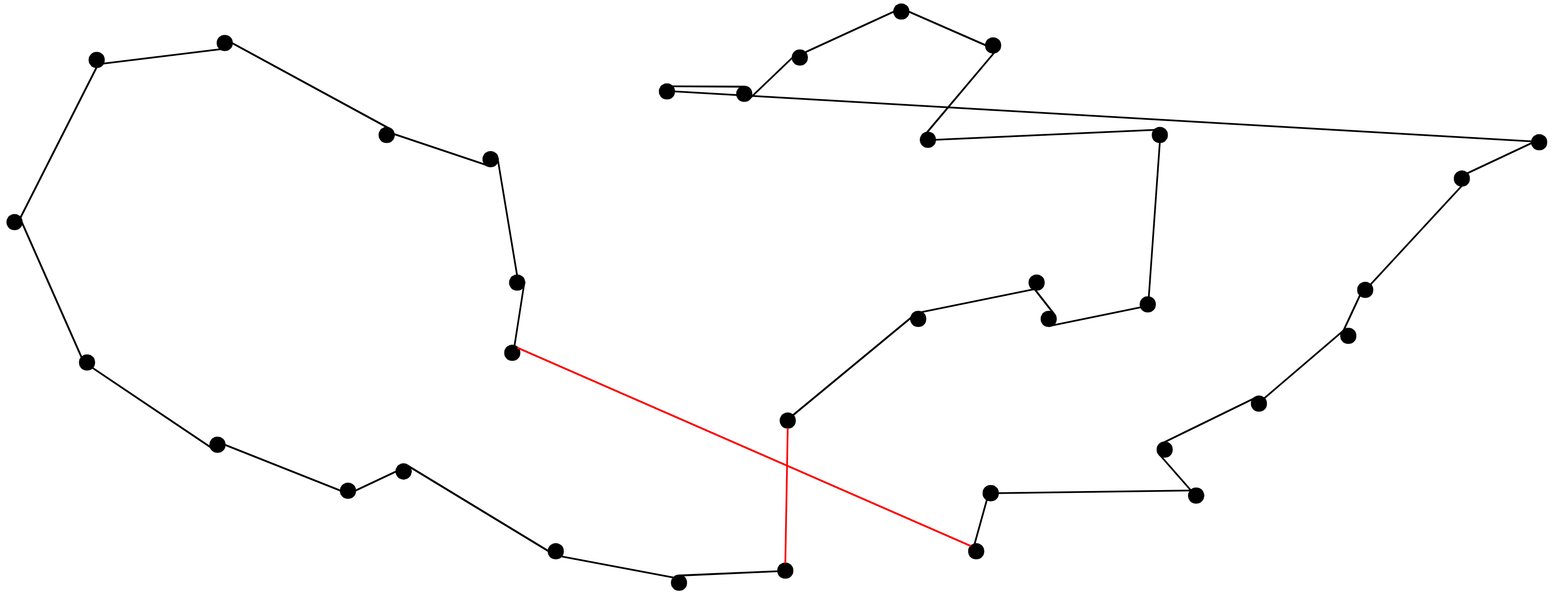


Optimal Tour

Improving an existing tour



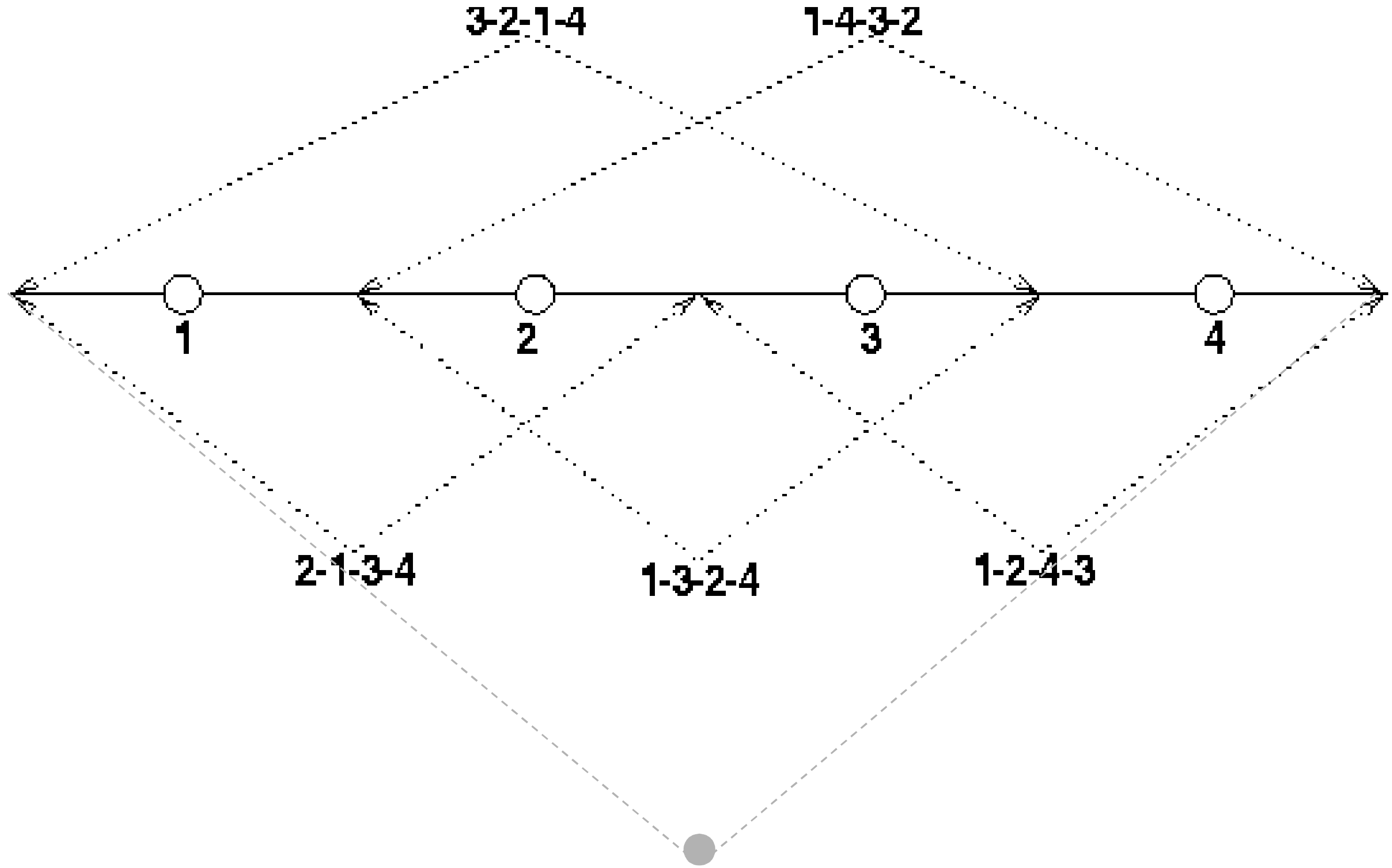
Which local modification can improve this tour?

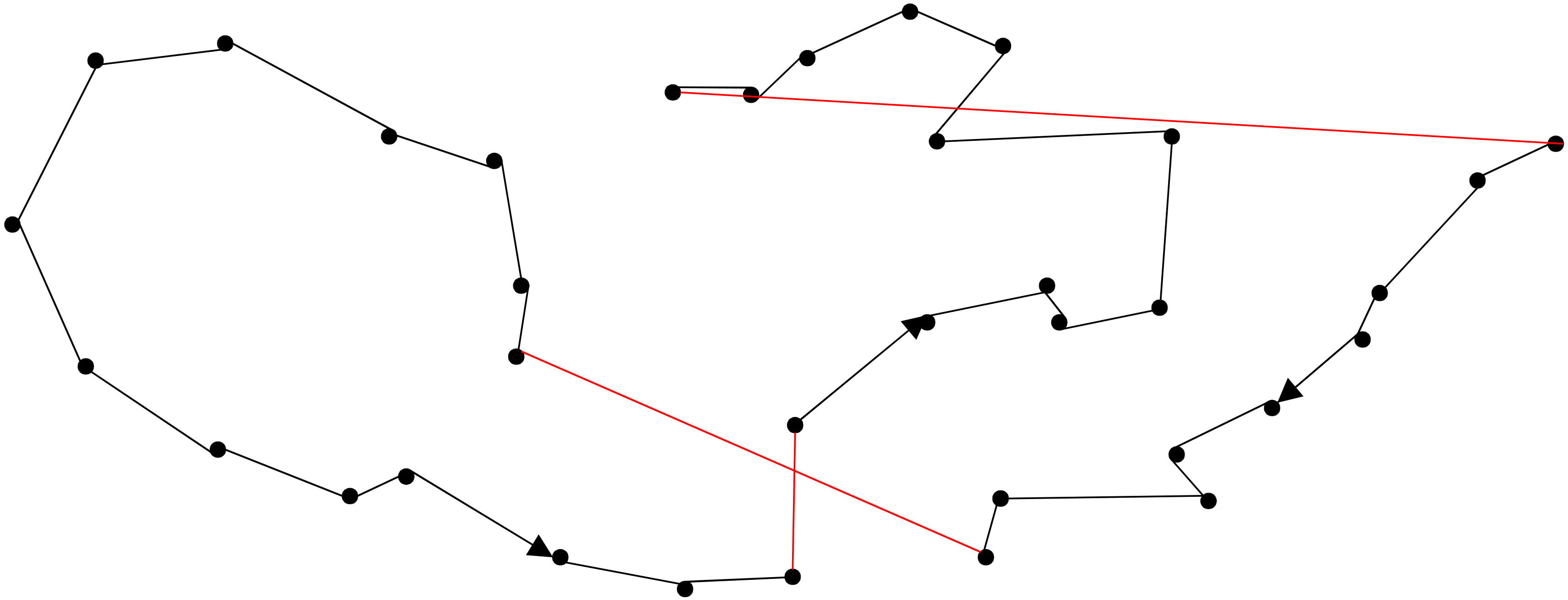


Remove two edges and rebuild another tour

⇔ Invert a given sequence of markers

2-change



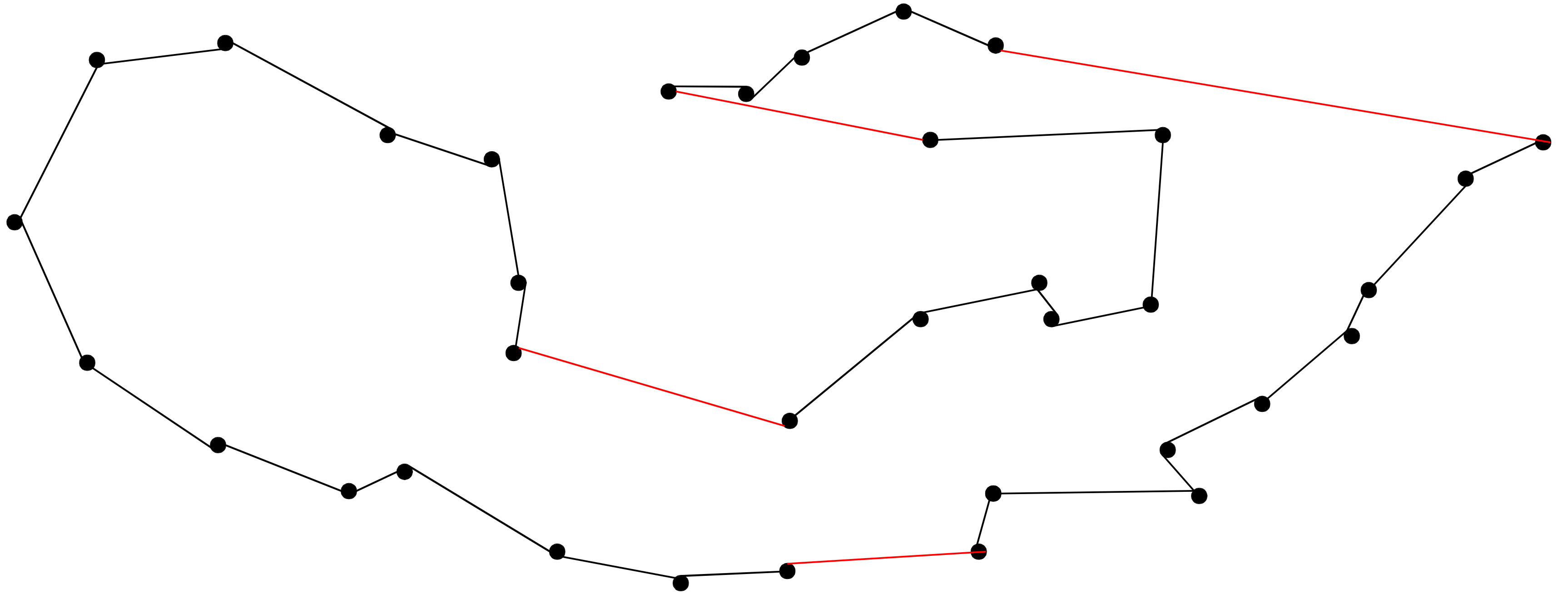


Remove three edges and rebuild another tour (7 possibilities)

⟹ Swap the order of two sequences of markers

« greedy » local search

- 2-opt
  - Note: a finite sequence of « 2-change » can reach any tour, including the optimum tour
  - Strategy:
    - Select the best 2-change among N*(N-1)/2 neighbors (2-move neighborhood)
    - Repeat this process until a fix point is reached (i.e. no tour improvement was made)



2-opt

Greedy local search

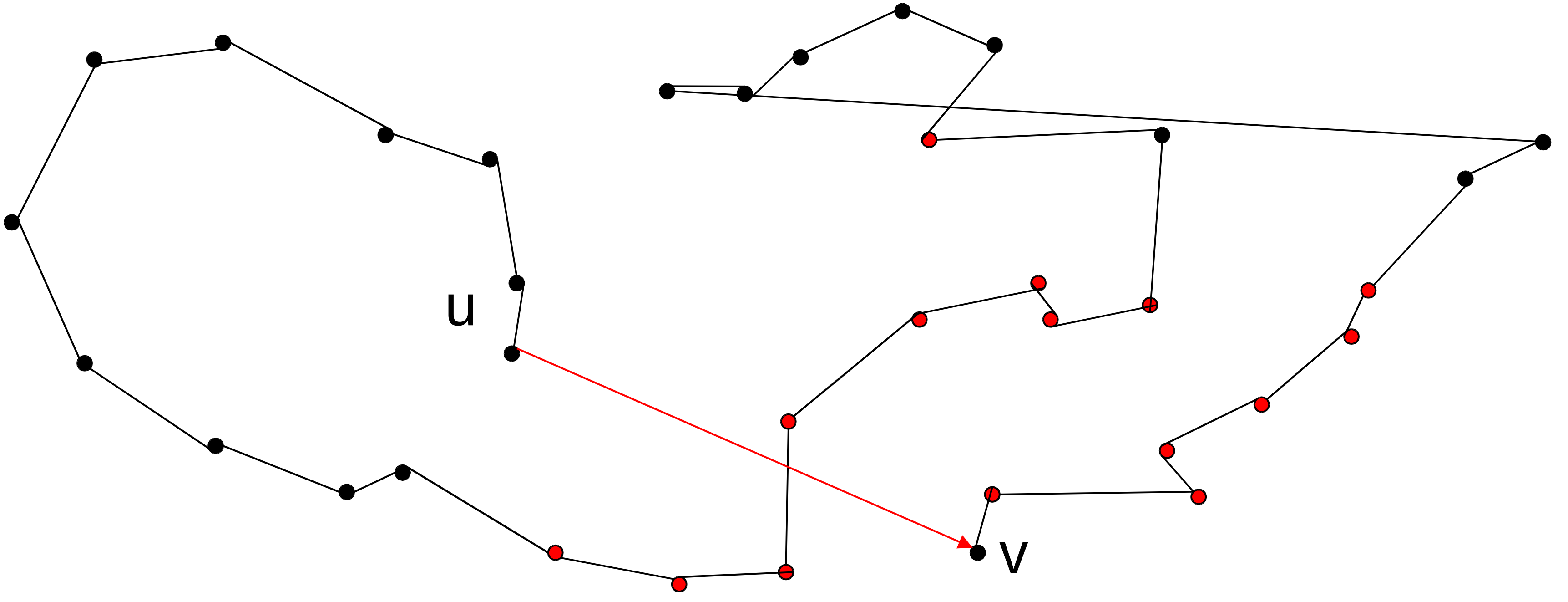
- Mean distance to the optimum
  - 2-opt : 9%
  - 3-opt : 4%
  - LK (limited k-opt) : 1%
- Complexity
  - 2-opt : $\sim N^3$
  - 3-opt : $\sim N^4$
  - LK (limited k-opt) : $< N^4$?

Complexity

n = number of vertices

Algorithm	Complexity
A-TSP	(n-1)!
S-TSP	(n-1)! / 2
2-change	1
3-change	7
k-change	$(k-1)! \cdot 2^{k-1}$
k-move	$(k-1)! \cdot 2^{k-1} \cdot n! / (k! \cdot (n-k)!)$
	$\sim O(n^k)$ k << n
In practice:	o(n)
2-opt et 3-opt	$\sim O(n^{k+1})$
In practice:	$o(n^{1.2})$
	time(3-opt) ~ 3 x time(2-opt)

2-opt implementation trick:



For each edge (uv), maintain the list of vertices w such that dist(w,v) < dist(u,v)

Lin & Kernighan (1973)

k-change : e_1->f_1,e_2->f_2,...

=> $Sum^k_{i=1}$(dist(e_i) - dist(f_i)) > 0

There is an order of i such that all the partial sums are positives:

S_l = $Sum^l_{i=1}$(dist(e_i) - dist(f_i)) > 0

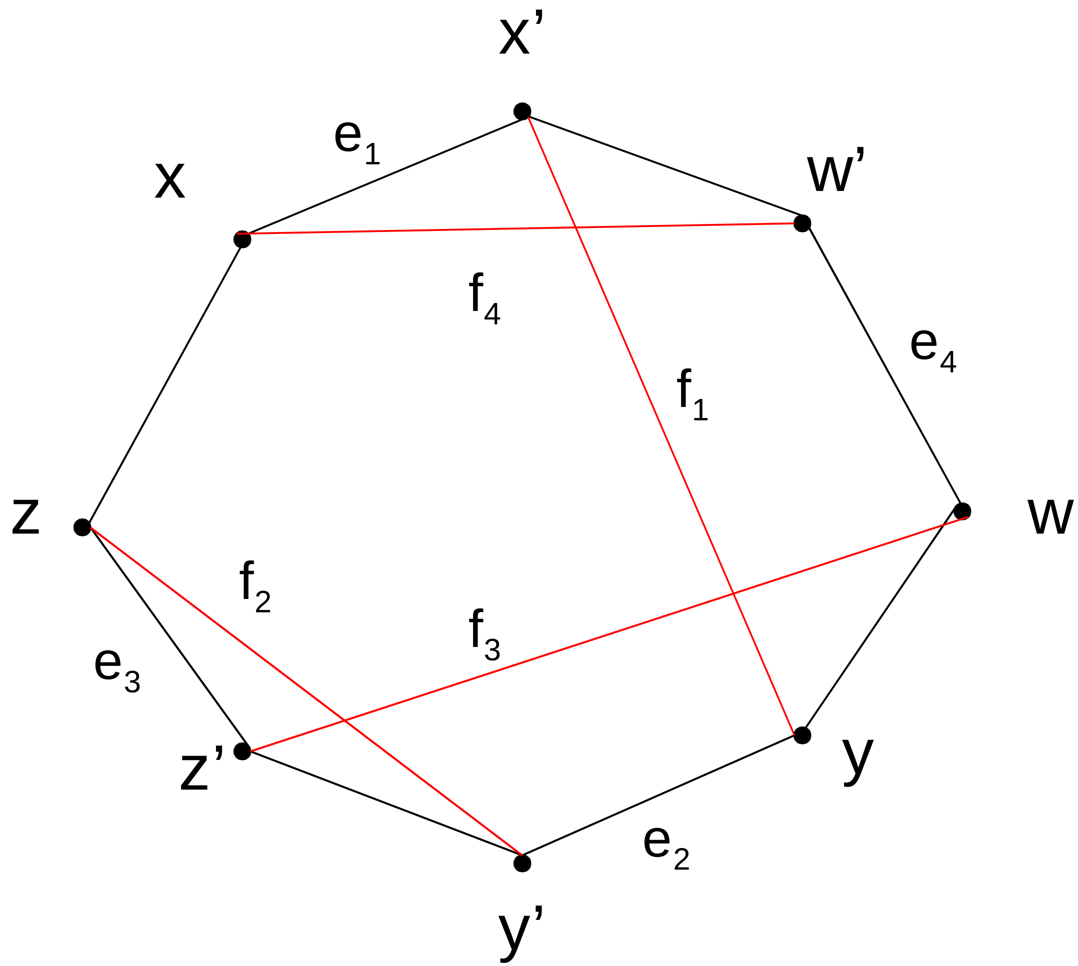
=> Build a valid increasing alternate cycle:

xx '->yx ', yy' -> zy', zz' -> wz', etc.

dist(f_1)<dist(e_1),dist(f_1)+dist(f_2)<dist(e_1)+dist(e_2),..

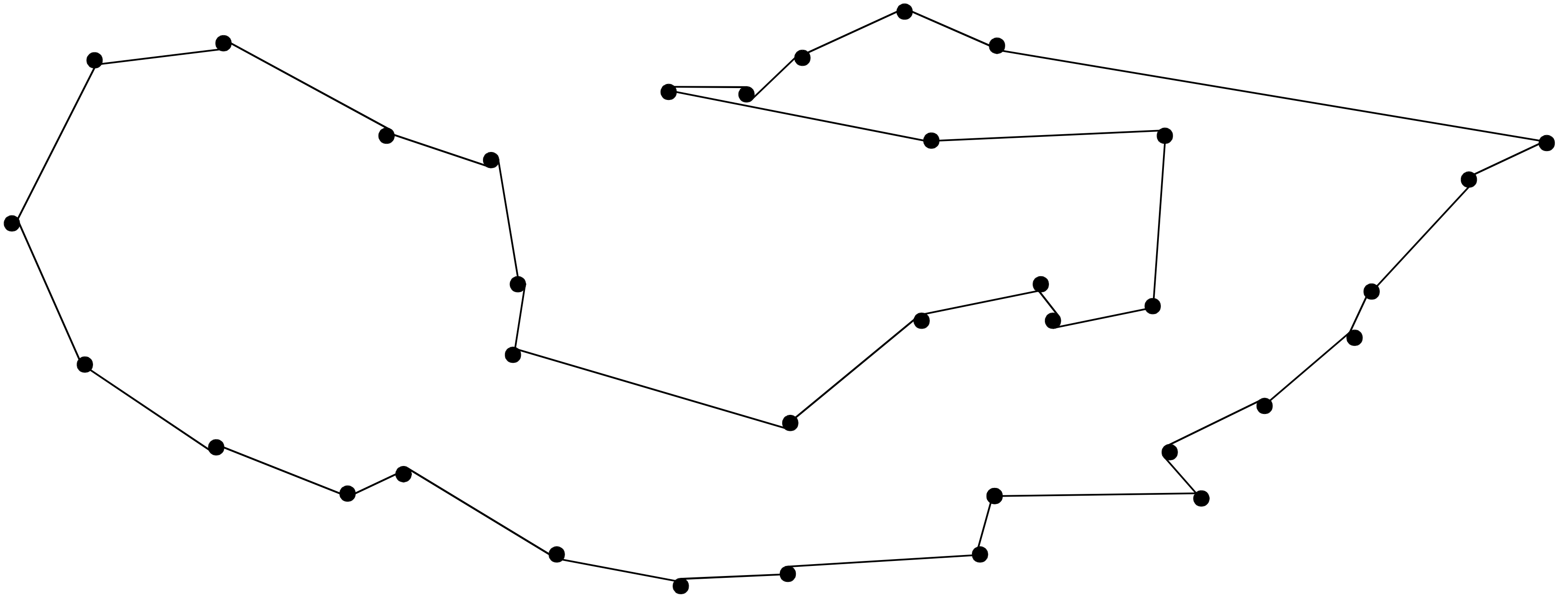
+ Backtrack on y and z choices + Restart

(in maximization)

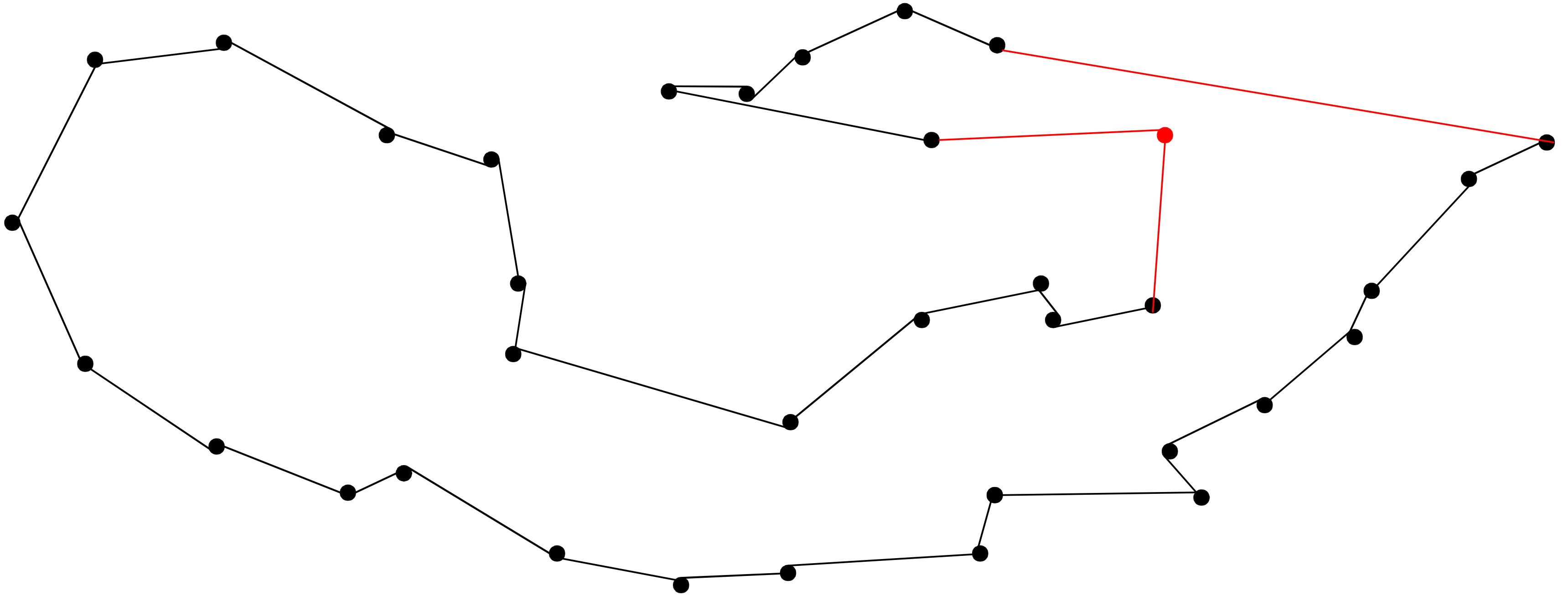


{x,y,z,w,..} ^ {x',y',z',w',..} = 0

y is among the 5 best neighbors of x', the same for z' and w

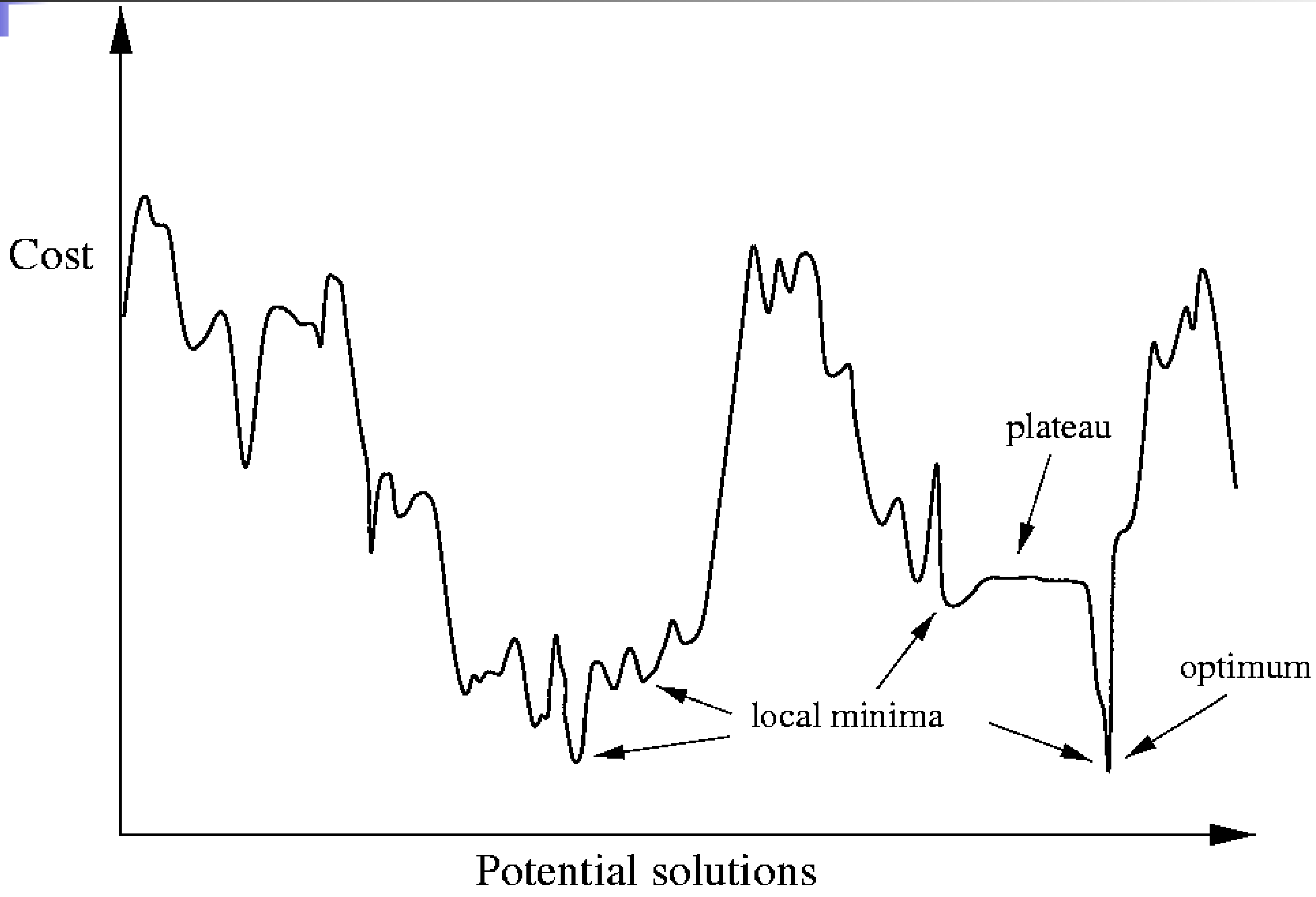


Is this 2-opt tour optimum?



2-opt + vertex reinsertion

local versus *global* optimum

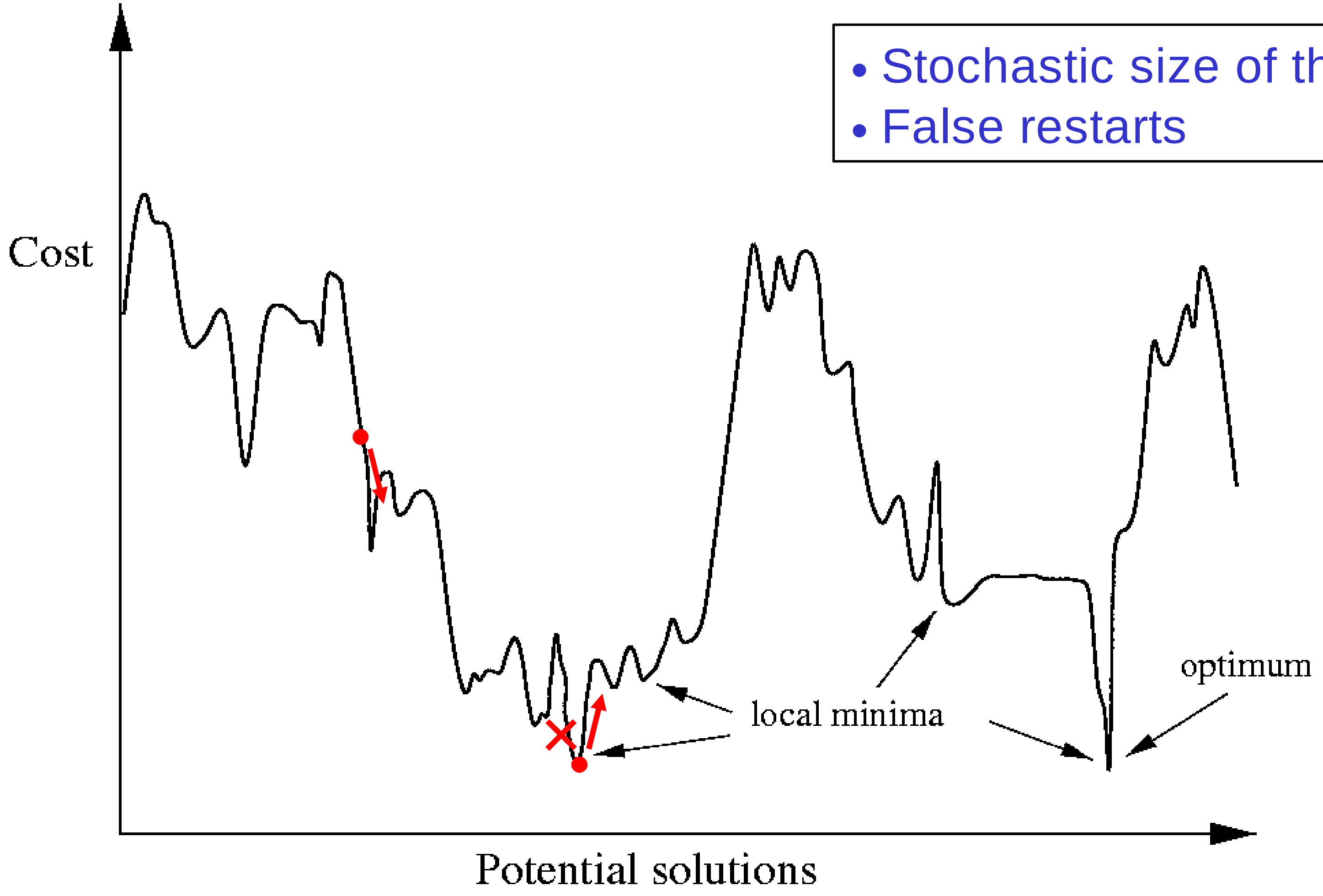


Local search & « meta-heuristics »

- Tabu Search
  - Select the best neighbor even if it decreases the quality of the current tour
  - Forbid previous local moves during a certain period of time
    - List of tabu moves
  - Restart with new tours
    - When the search goes to a tour already seen
    - Build new tours in a random way

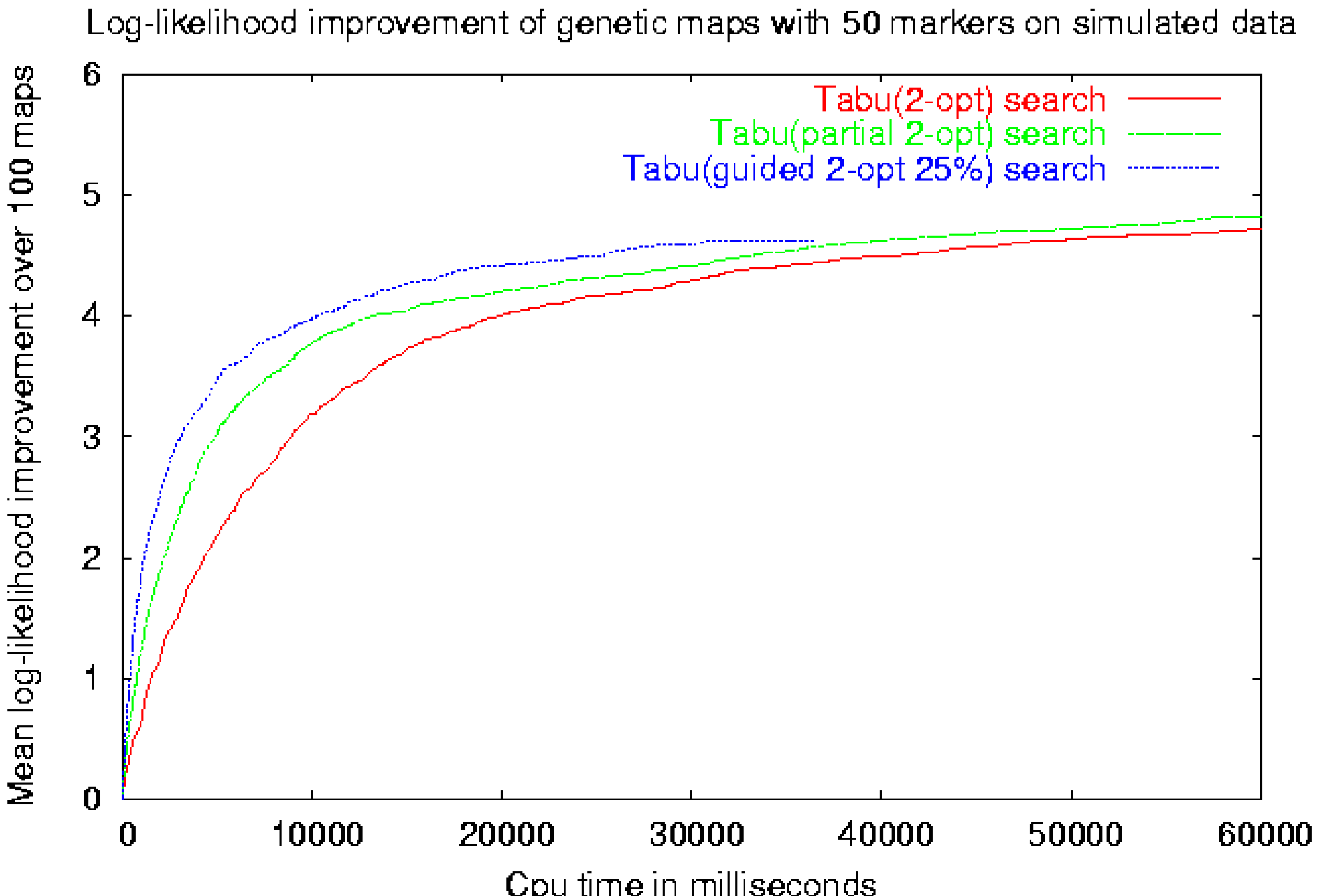
Tabu Search

- Stochastic size of the tabu list
- False restarts



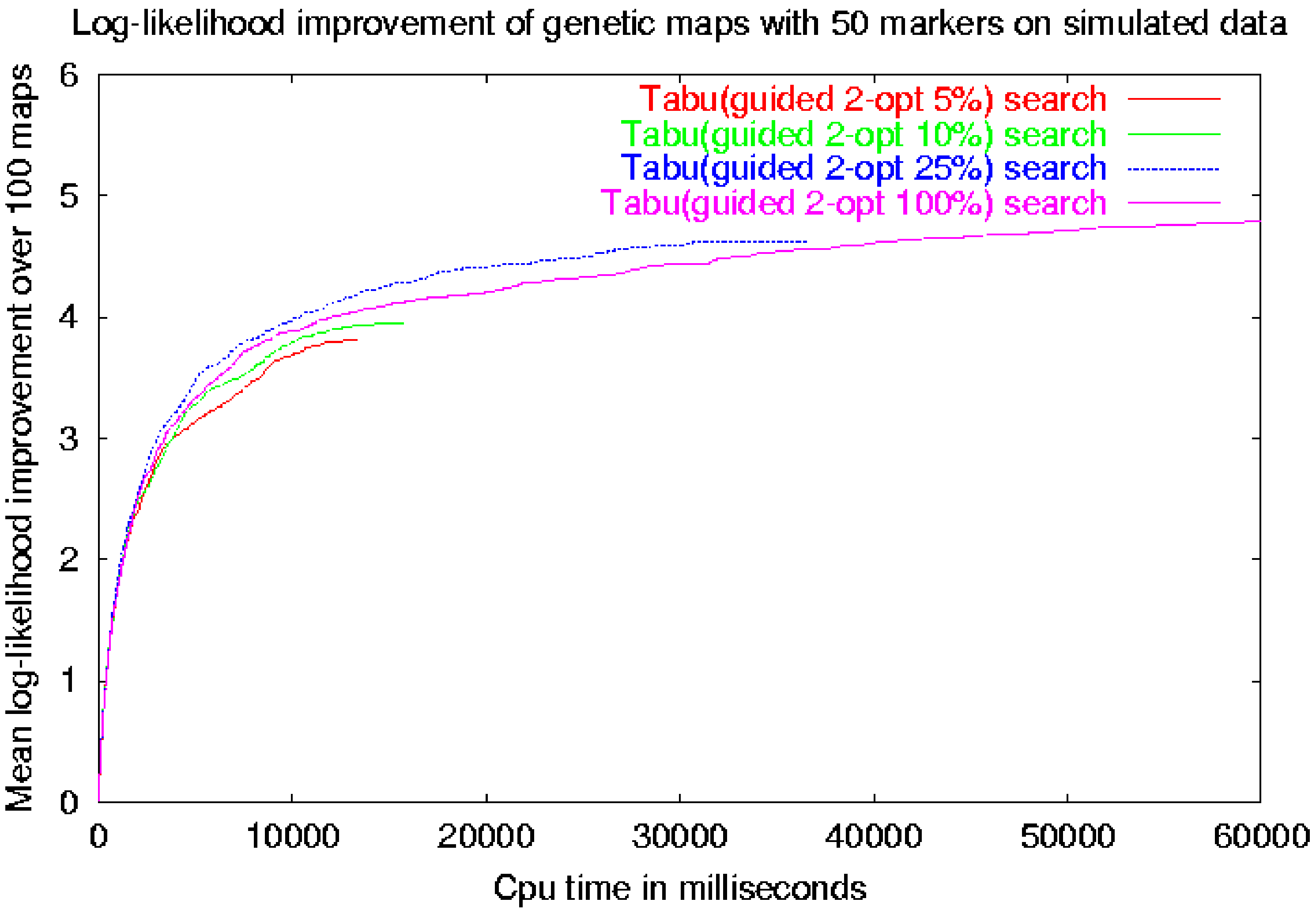
Experiments with CarthaGène

N=50 K=100 Err=30% Abs=30%



Legend: partial 2-opt = early stop , guided 2-opt 25% = early stop & sort with X = 25%

Experiments - next

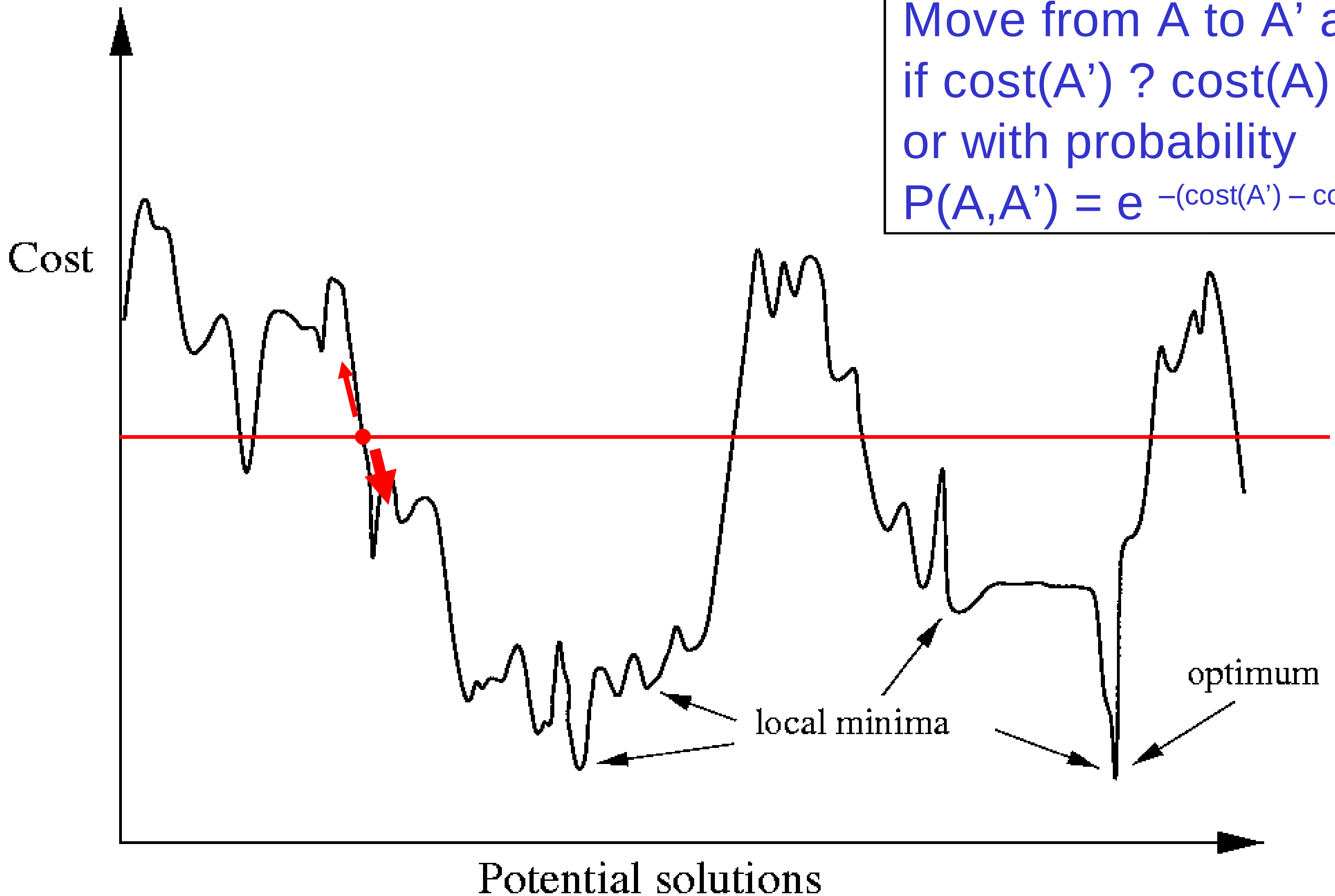


Other meta-heuristics

- Simulated Annealing
  - Local moves are randomly chosen
  - Neighbor acceptance depends on its quality
  - Acceptance process is more and more greedy
- Genetic Algorithms
  - Population of solutions (tours)
  - Mutation, crossover,…
- Variable Neighborhood Search
- …

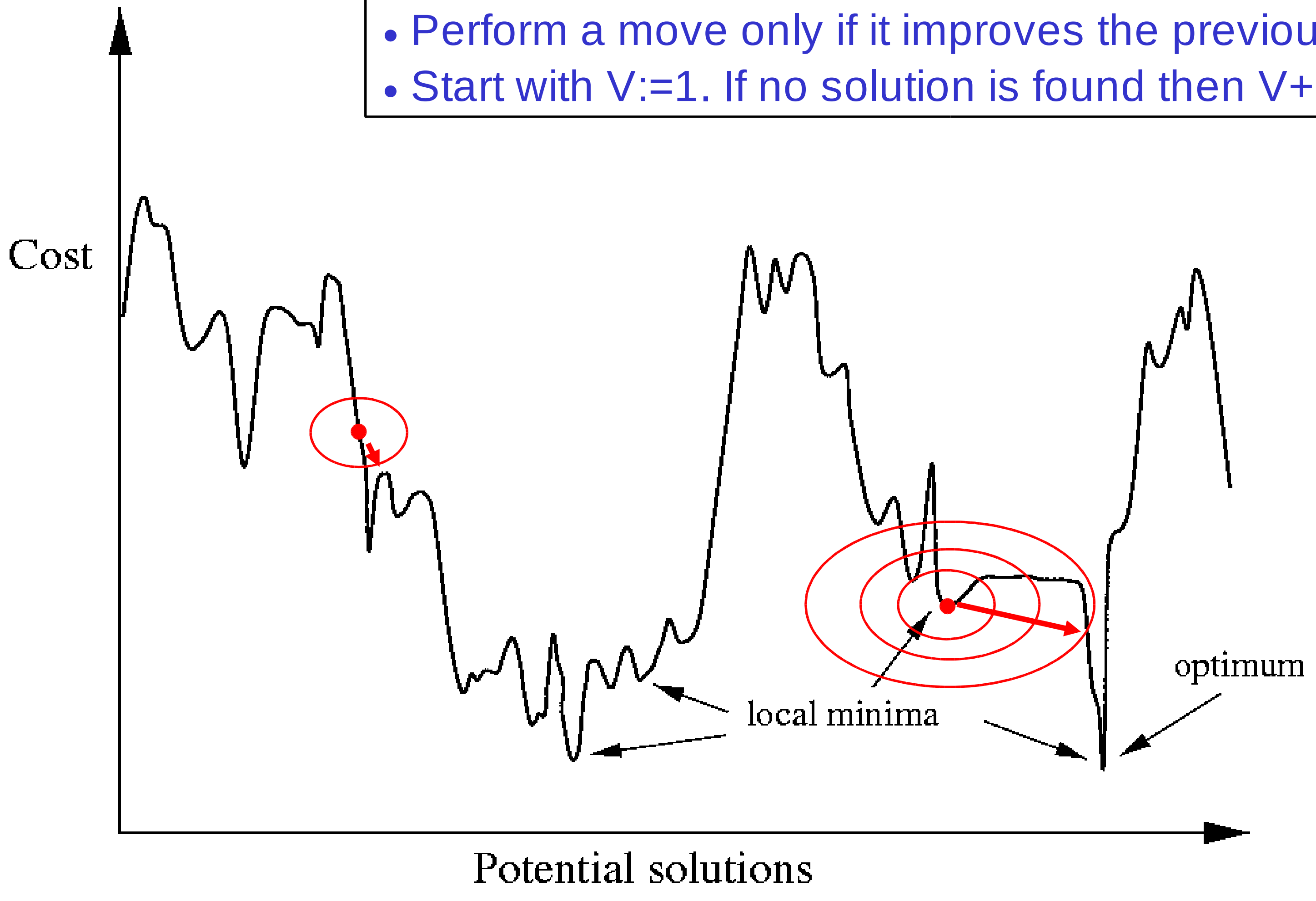
Simulated Annealing

Move from A to A' accepted
if cost(A') ? cost(A)
or with probability
$P(A,A') = e^{-(cost(A') - cost(A))/T}$



Variable Neighborhood Search

- Perform a move only if it improves the previous solution
- Start with V:=1. If no solution is found then V++ else V:=1



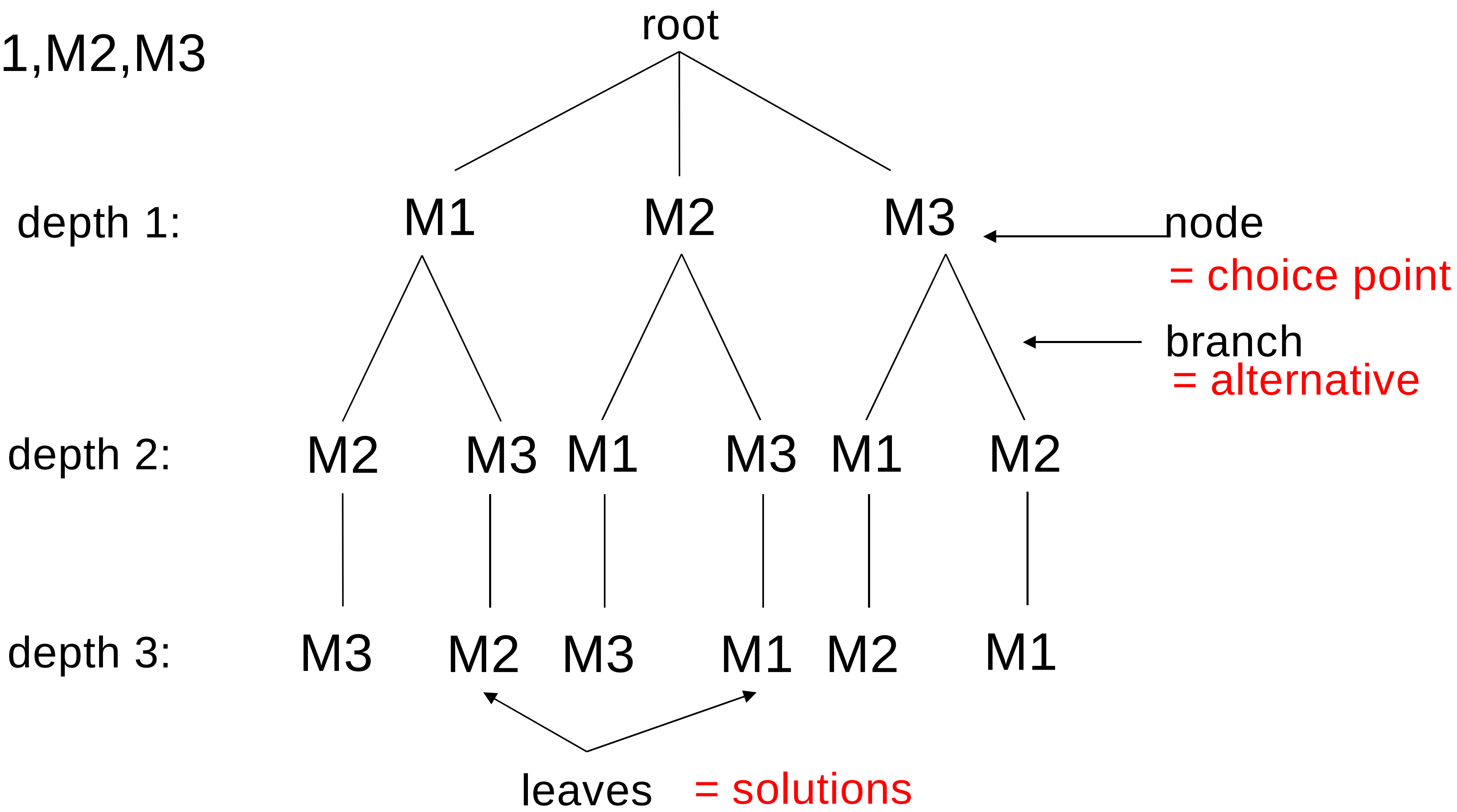
Local Search

Demonstration

Finding the best tour

Search tree

M1,M2,M3

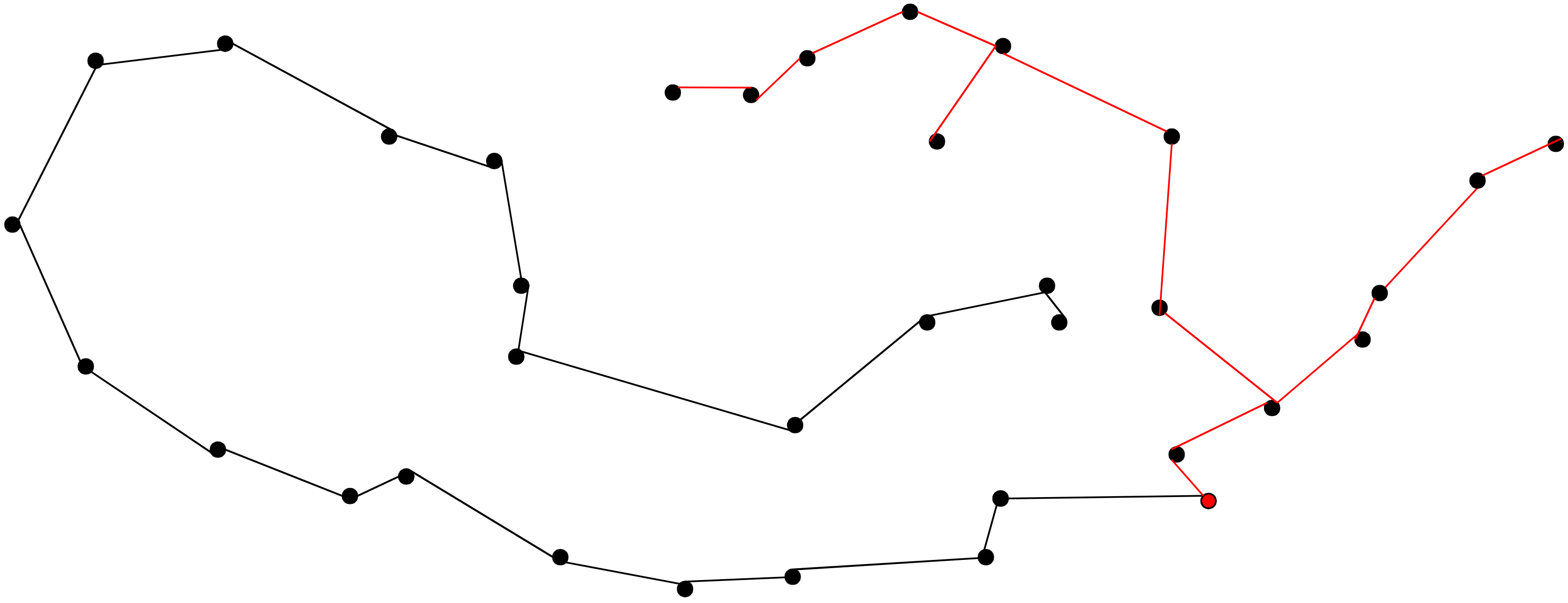


Tree search

- Complexity : n!/2 different orders
  - Avoid symmetric orders (first half of the tree)
- Can use *heuristics* in choice points to order possible alternatives
- *Branch and bound* algorithm
  - Cut all the branches which cannot lead to a better solution
- Possible to combine local search and tree search

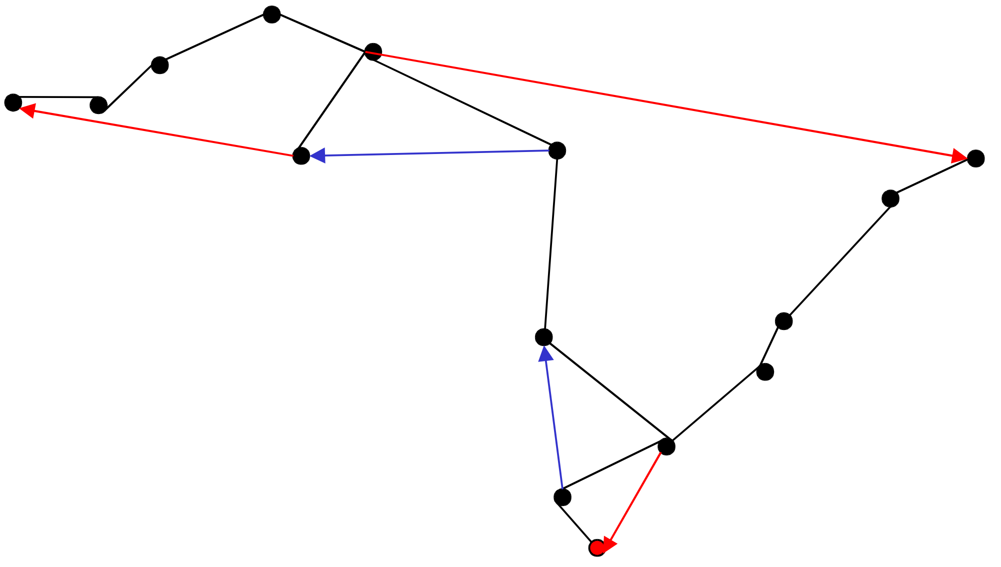
Branch and bound

Minimum weight spanning tree



Prim algorithm (1957)

Held & Karp algorithm (better spanning trees) (1971)
≤linear programming relaxation of TSP, LB(I)/OPT(I) ≥2/3



Christofides heuristic (1976)

=> A(I) / OPT(I) ≤3/2   (with triangular inequalities)

Complexity

Complexity	Standard computer	Computer 100 times faster	Computer 1000 times faster
N	N1	100*N1	1000*N1
N^2	N2	10*N2	31,6*N2
N^3	N3	4,64*N3	10*N3
2^N	N4	N4+6,64	N4+9,97
3^N	N5	N5+4,19	N5+6,29

Complete methods

1954 : 49 cities

1971 : 64 cities

1975 : 100 cities

1977 : 120 cities

1980 : 318 cities

1987 : 2,392 cities

1994 : 7.397 cities

1998 : 13.509 cities

2001 : 15.112 cities (585936700 sec. ≈19 years of CPU!)