CMU 15-781

Lecture 2a: Local Search

Teacher:
Gianni A. Di Caro

PATH SEARCH VS. LOCAL SEARCH

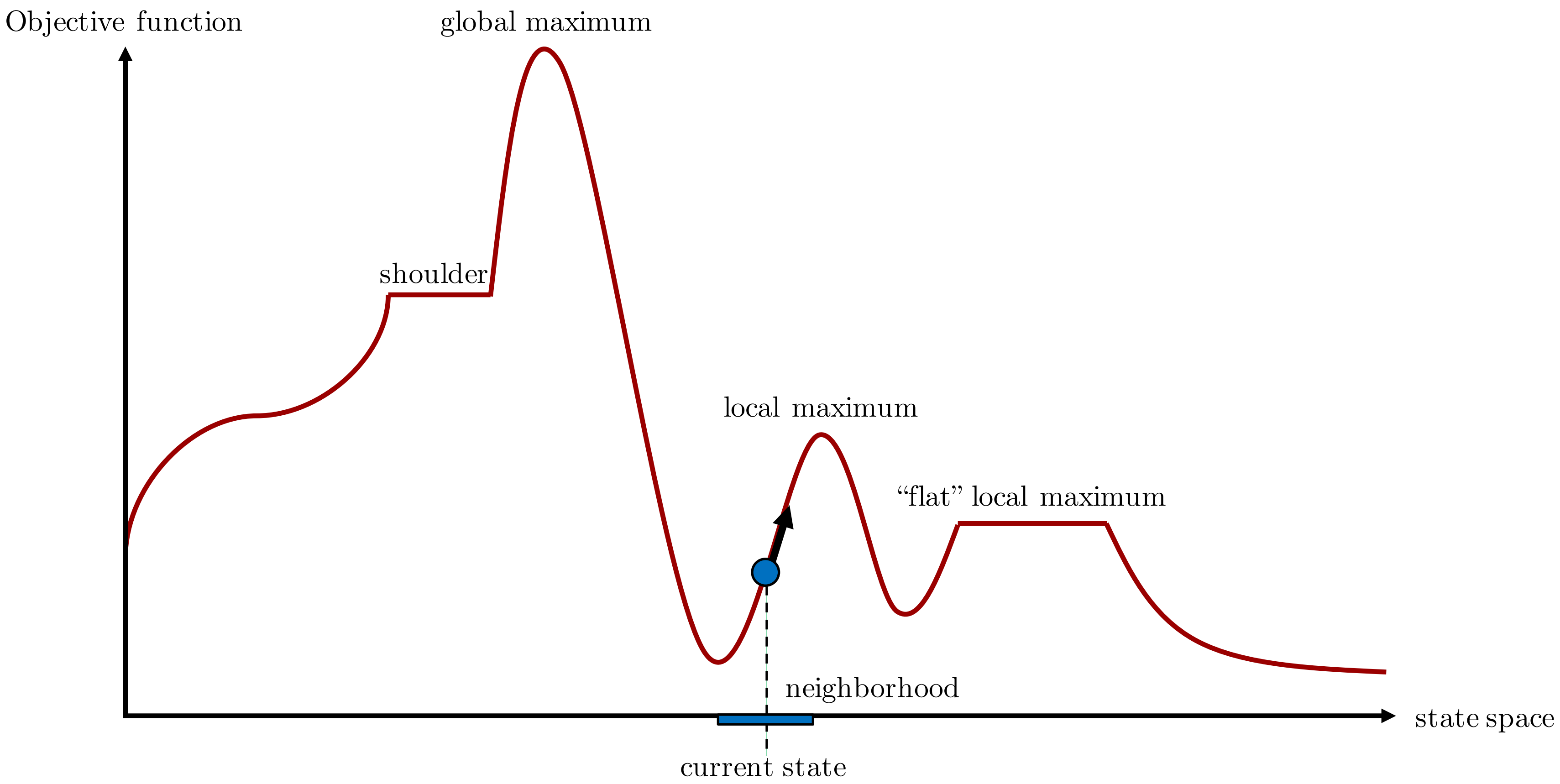
- The algorithms discussed so far are designed to find a goal state from a start state: the *path* to the goal constitutes a *solution* to the search problem
- In many problems the path doesn't matter:
  the goal state itself is the solution

- **State space** = set of "complete" configurations
  - **Optimization problems:** Find *optimal* configuration (objective or cost function)
  - **Constraint Satisfaction Problems:** Find configurations satisfying (all or the highest number of) *constraints*

PATH SEARCH VS. LOCAL SEARCH

- Local search algorithms at each step consider a *single "current" state*, and try to improve it by moving to one of its neighbors ➔ **Iterative improvement algorithms**
- <u>Pros and cons</u>
  - No complete (no optimal), except with *random restarts*
  - Space complexity $\mathcal{O}(b)$
  - Time complexity $\mathcal{O}(d)$, d can be ∞!
  - Can perform well also in large (infinite, continuous) spaces
  - Relatively easy to implement

State-space landscape



Hill-Climbing Search

Like climbing Everest in thick fog with amnesia

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
       neighbor ← a highest-valued successor of current
       if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
       current ← neighbor
   end
```

- Move in the direction of increasing value (up the hill)
- Terminate when no neighbor has higher value
- Greedy (myopic) local search

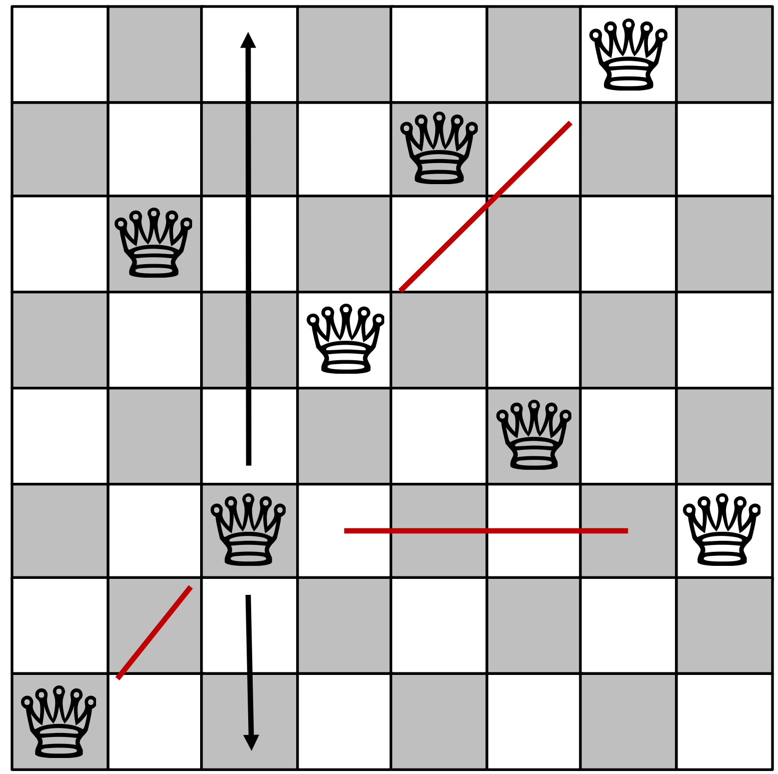
CSP Example: N-Queens

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

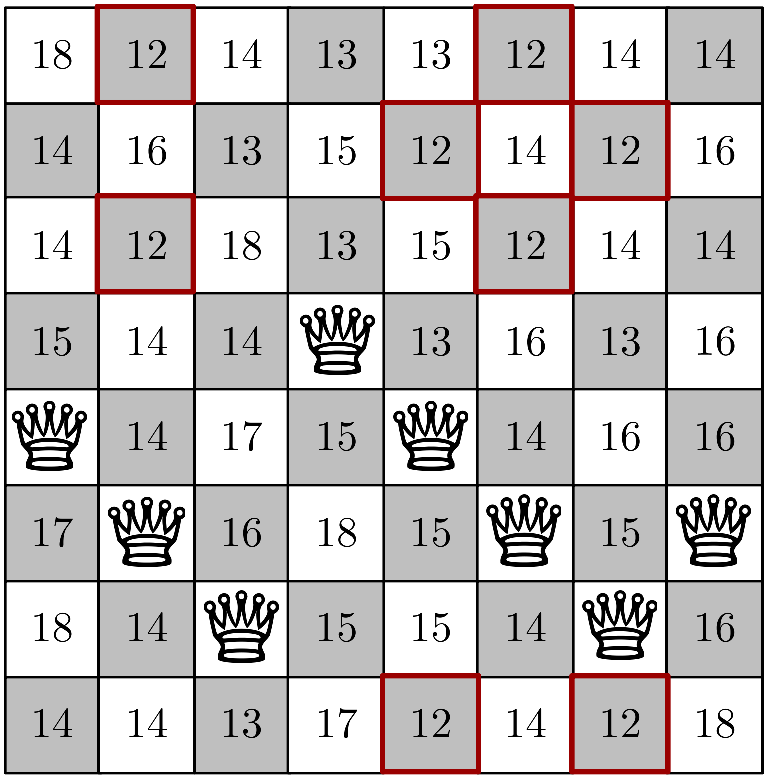
State: Position of the n queens, one per column (or row)

Successor states: generated by moving a single queen to another square in its column *(n(n-1))*

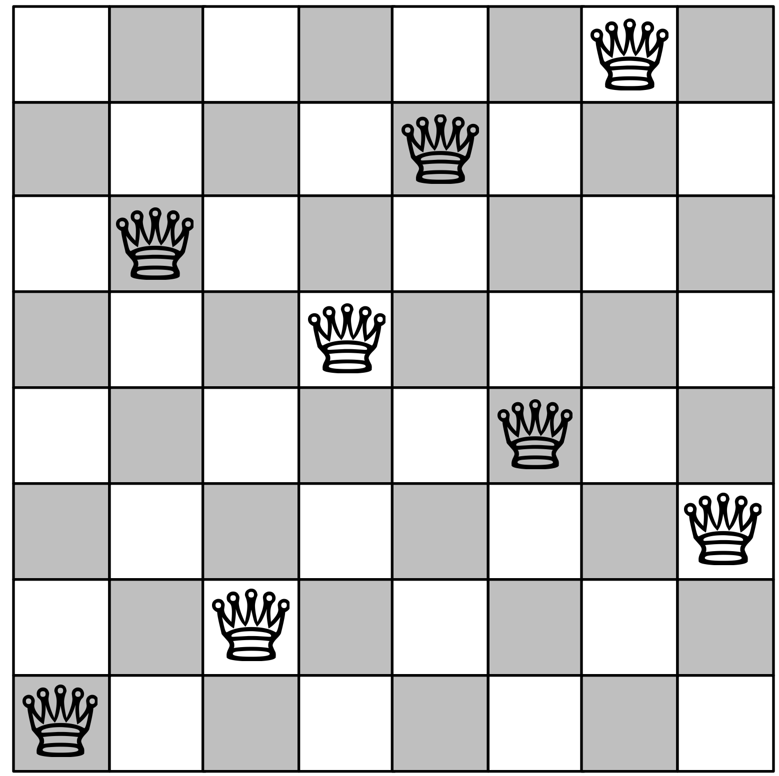
Cost of a state: the number of constraint violations



N-Queens



State with 17 conflicts, showing the #conflicts by moving a queen within its column, with best moves in red

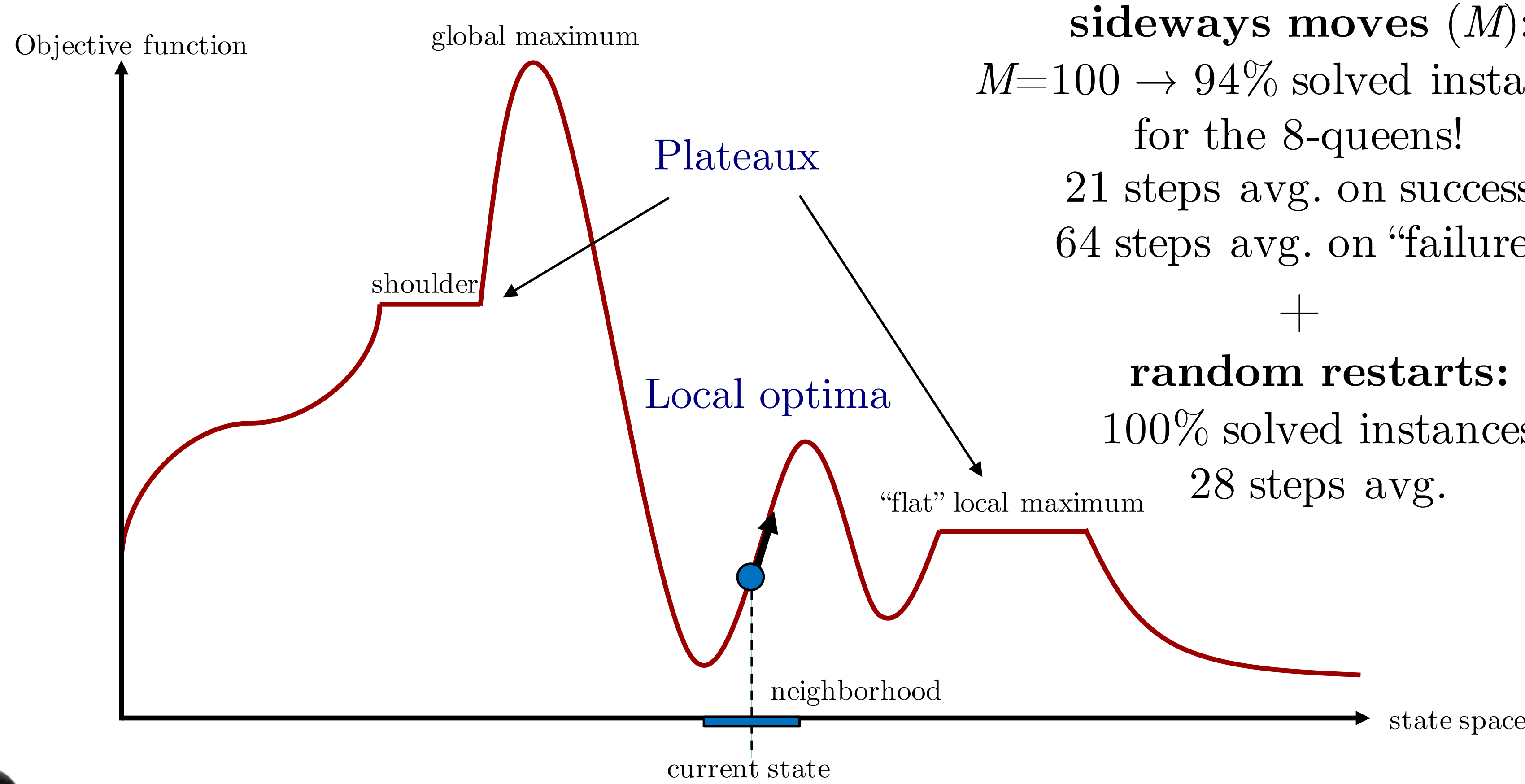


Local optimum: state that has only one conflict, but every move leads to larger #conflicts

Hill-Climbing performance on N-Queens

- Hill-climbing can solve large instances of n-queens ($n = 10^6$) in a few (ms)seconds
- 8 queens statistics:
  - State space of size ≈17 million
  - Starting from random state, steepest-ascent hill climbing solves 14% of problem instances
  - It takes 4 steps on average when it succeeds, 3 when it gets stuck
  - When "sideways" moves are allowed, performance improve ...
  - When multiple restarts are allowed, performance improves even more

HILL-CLIMBING CAN GET STUCK!

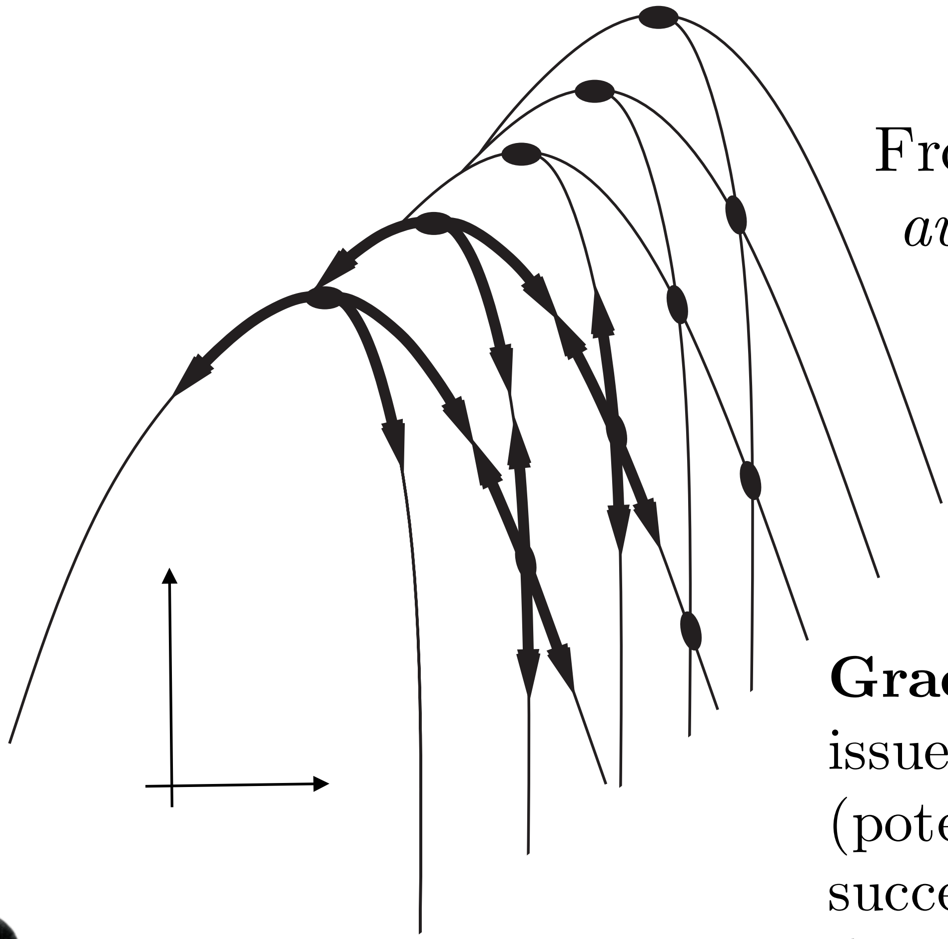


sideways moves (M):
M=100 → 94% solved instances
for the 8-queens!
21 steps avg. on success
64 steps avg. on "failure"

+

random restarts:
100% solved instances
28 steps avg.

HILL-CLIMBING CAN GET STUCK!



Diagonal ridges:

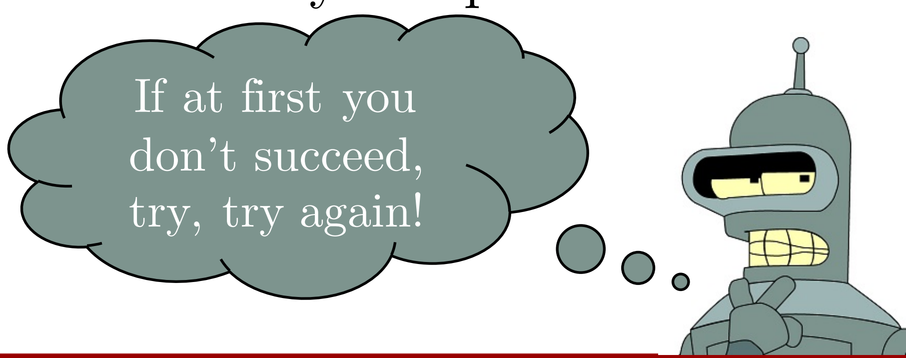
From each local maximum all the *available* actions point downhill, but there is an uphill path!

Zig-zag motion, very long ascent time!

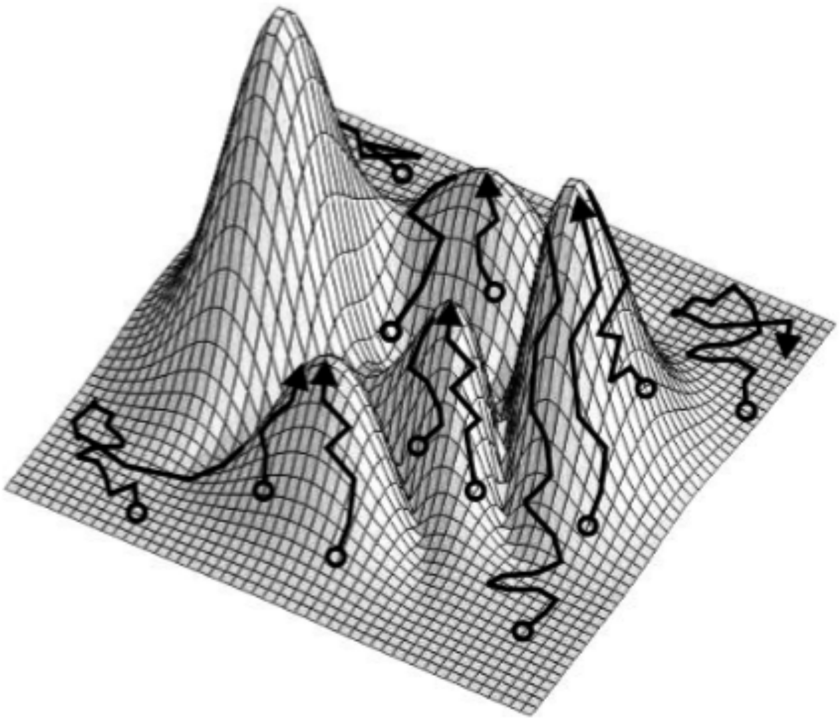
Gradient ascent doesn't have this issue: *all* state vector components are (potentially) changed when moving to a successor state, climbing can follow the direction of the ridge

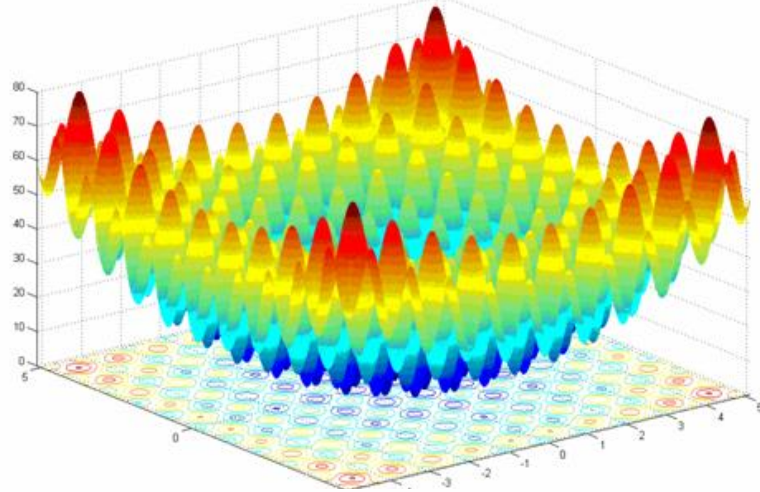
Variants of hill-climbing

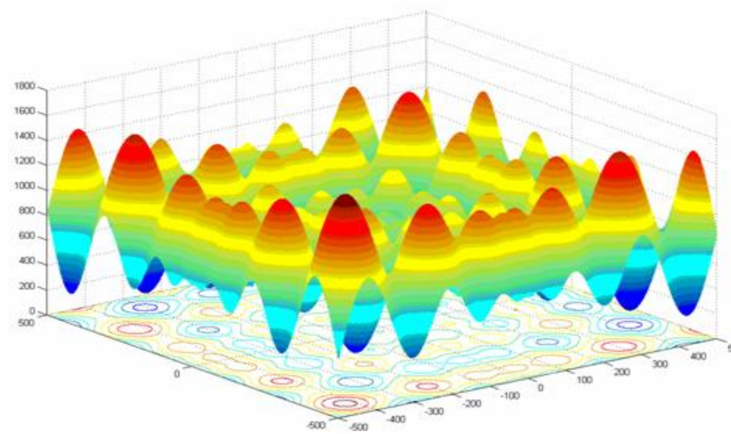
- Sideways moves: if no uphill moves, allow moving to a state with the *same value* as the current one (escape shoulders)
- Stochastic hill-climbing: *selection* among the available uphill moves is done *randomly* (uniform, proportional, soft-max, ε-greedy, …) to be "less" greedy
- First-choice hill-climbing: successors are generated *randomly*, one at a time, until one that is better than the current state is found (deal with large neighborhoods)
- Random-restart hill climbing: probabilistically complete



Trajectories, difficulties

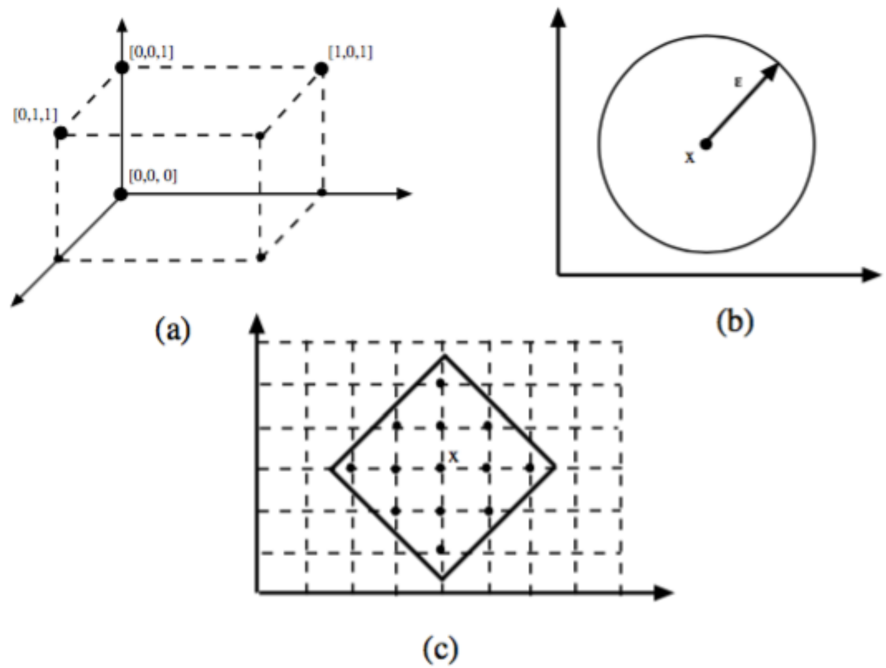




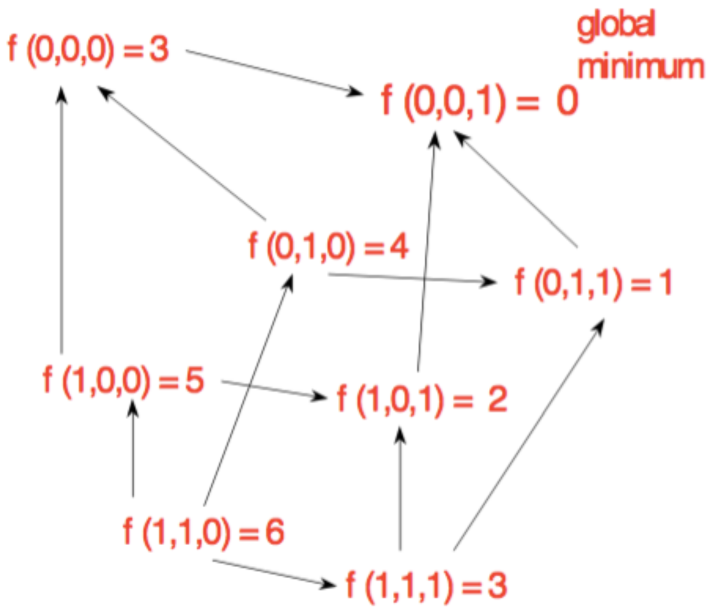


NEIGHBORHOOD

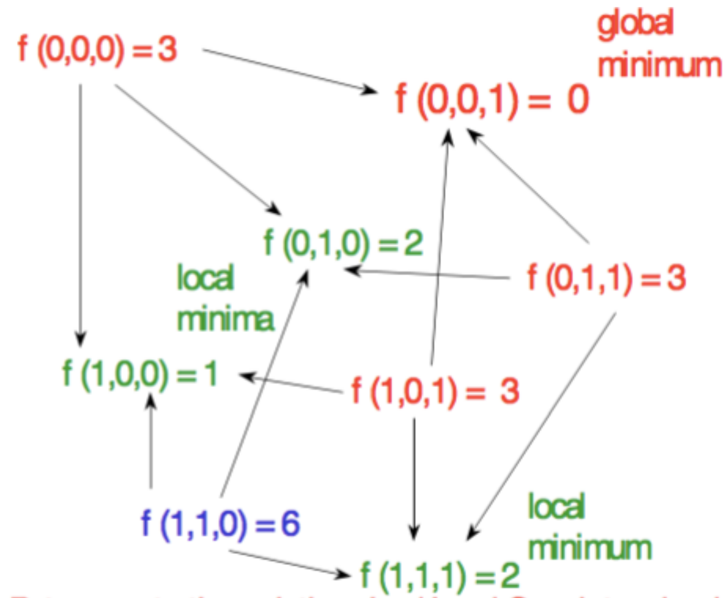
- A *mapping (rule)* that associate two states *(s,s')*
- It should preserve a certain degree of *correlation* between the value of *s* and that of *s'*
- It should balance *size and search*



GOOD VS. REALISTIC SCENARIOS

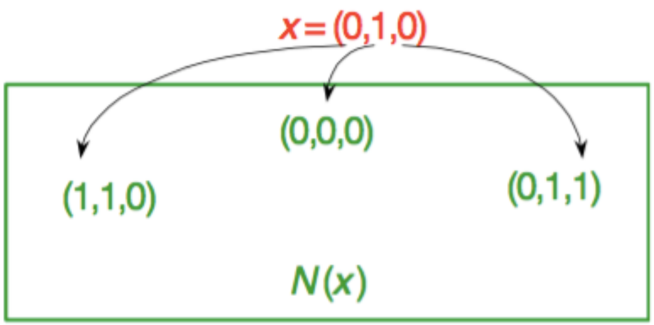


With any starting solution Local Search finds the global optimum.

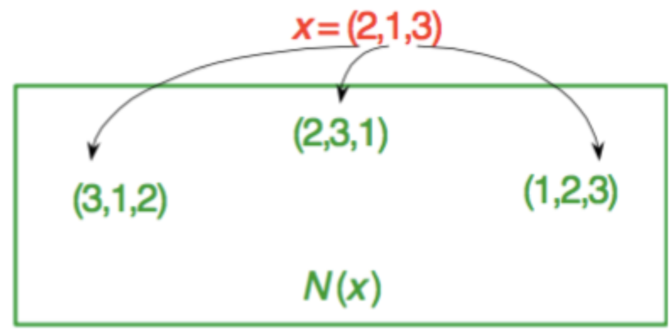


But some starting solutions lead Local Search to a local minimum.

Example neighborhoods



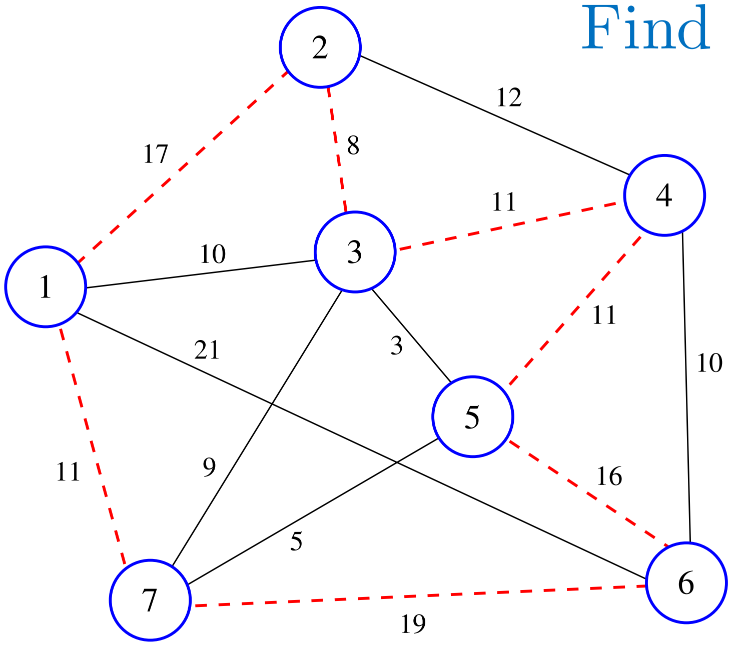
1-flip neighborhood,
for 0-1 vectors



2-swap neighborhood,
for permutation vectors

k-exchange neighborhood (for TSP and similar problems): The neighborhood $N(s)$ of a solution s is the set of solutions s' that differ from s up to k solution components

Optimization example: TSP



Find the Hamiltonian tour of minimal cost

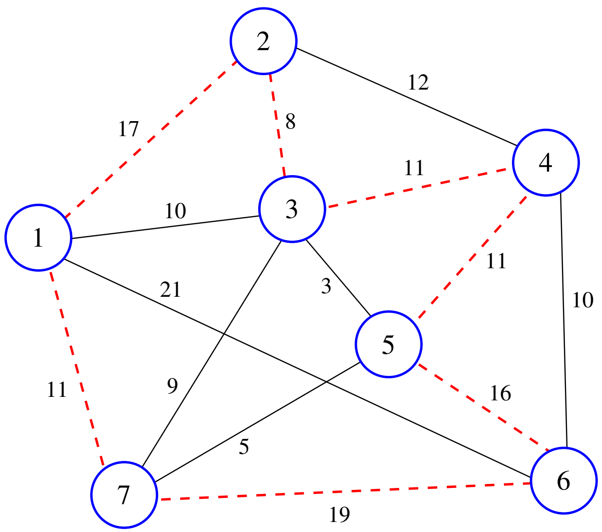
Every *cyclic permutation* of n integers is a feasible solution

If two nodes are not connected, they can be seen as connected by an arc of ∞ length!

$$\pi_1 = (1,3,4,2,6,5,7,1), \;\; \pi_2 = (2,3,4,5,6,7,1,2)$$
$$c(\pi_2) = d_{23} + d_{34} + d_{45} + d_{56} + d_{67} + d_{71} + d_{12} = 93$$

Read also as set of edges: {(2,3), (3,4), (4,5), (5,6), (6,7), (7,1), (1,2)}

Optimization example: TSP



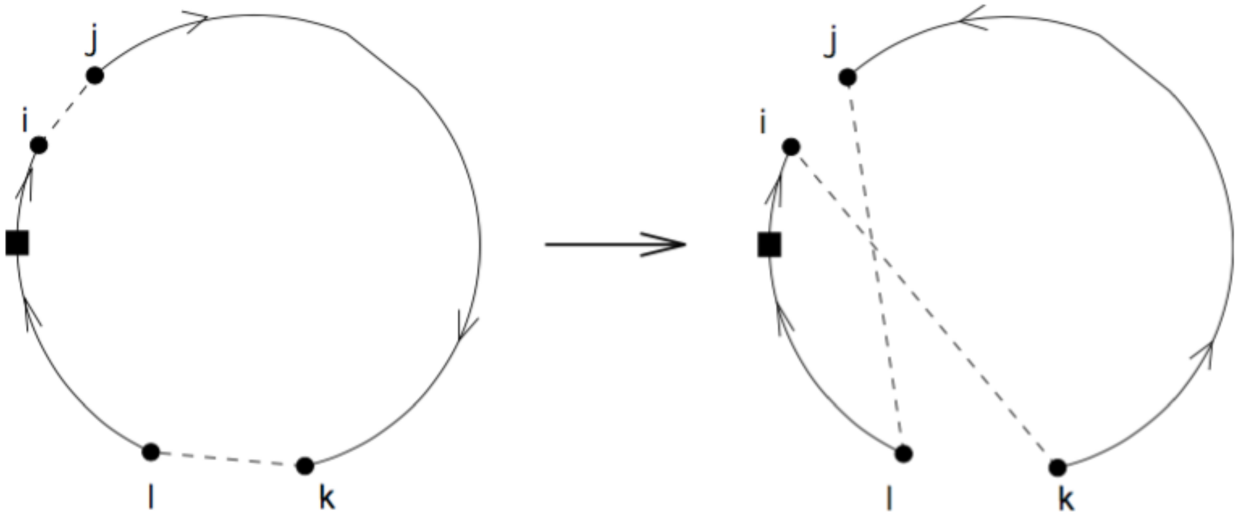
K-exchange neighborhood:

$N(s)$ is the set of tours s' can be obtained from s by exchanging k *edges* in s with k edges in $E \setminus \{s\}$ (E is the graph's edge set)

Each s' is obtained deleting a selected set of k edges in s and *rewiring* the resulting fragments into a complete tour by *inserting a different set of k edges*

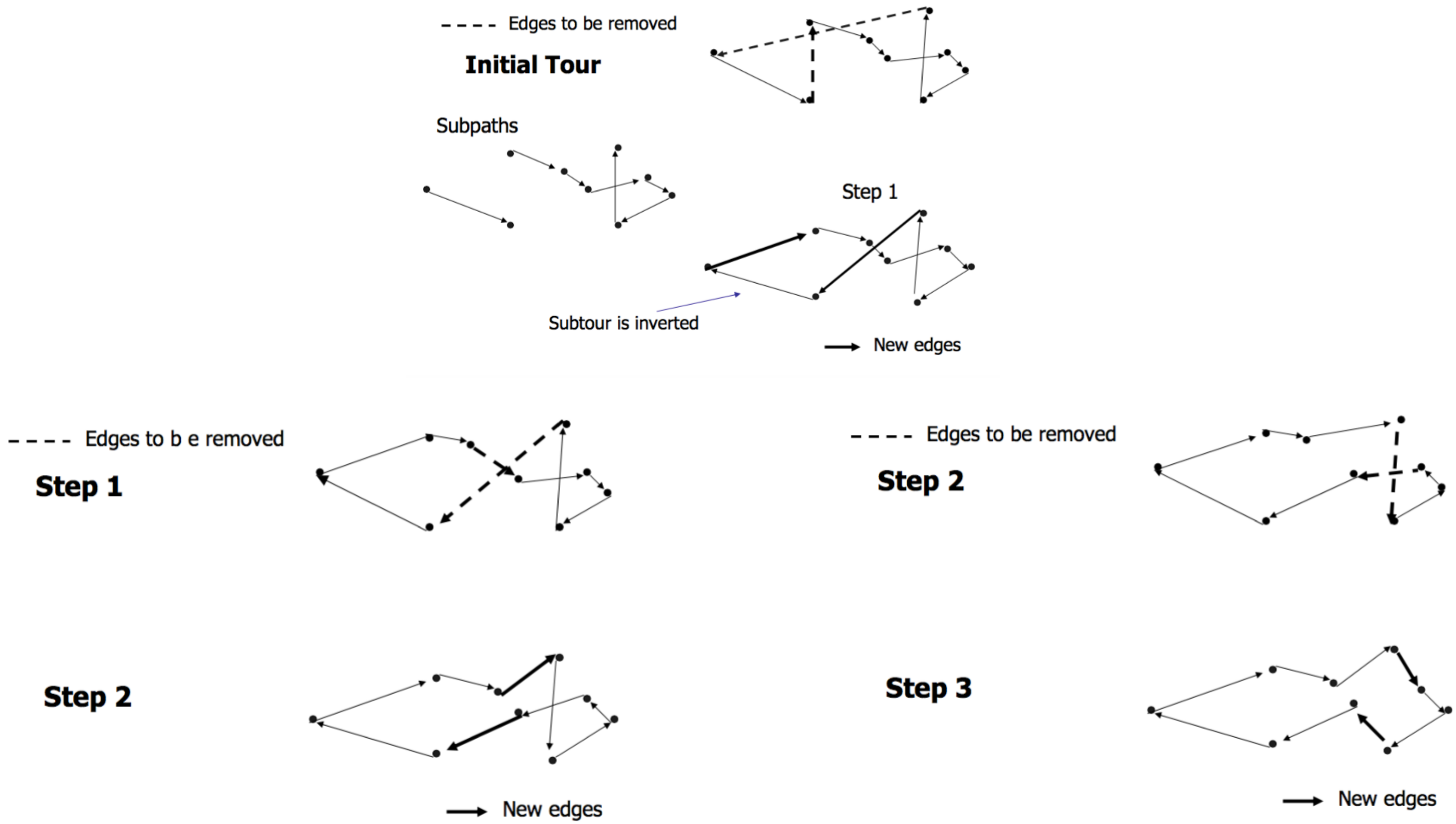
$\binom{n}{k}$ possible ways to drop k edges in a tour
$(k-1)!2^{k-1}$ ways to relink the disconnected paths

2-Opt local search

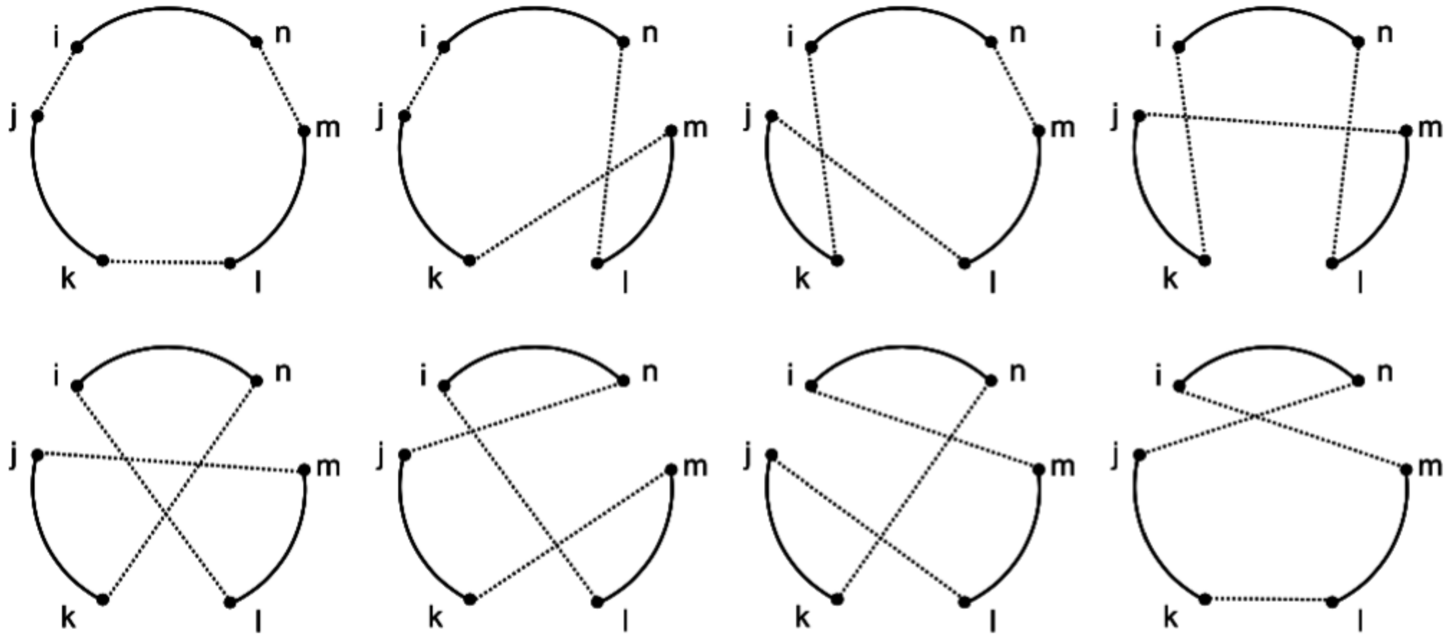


- Two edges, (i,j) and (l,k), are **selected, removed,** and **replaced** by two other edges (i,k) and (j,l) (or, (k,i), (l,j))
- One of the two paths needs to get *reverted*!
- Gain: $(i,k) + (j,l) - (i,j) - (k,l)$
- $n(n-1)=O(n^2)$ possible successors in the 2-exchange neighborhood ➔ quadratic search complexity for each single 2-opt step move

2-Opt local search

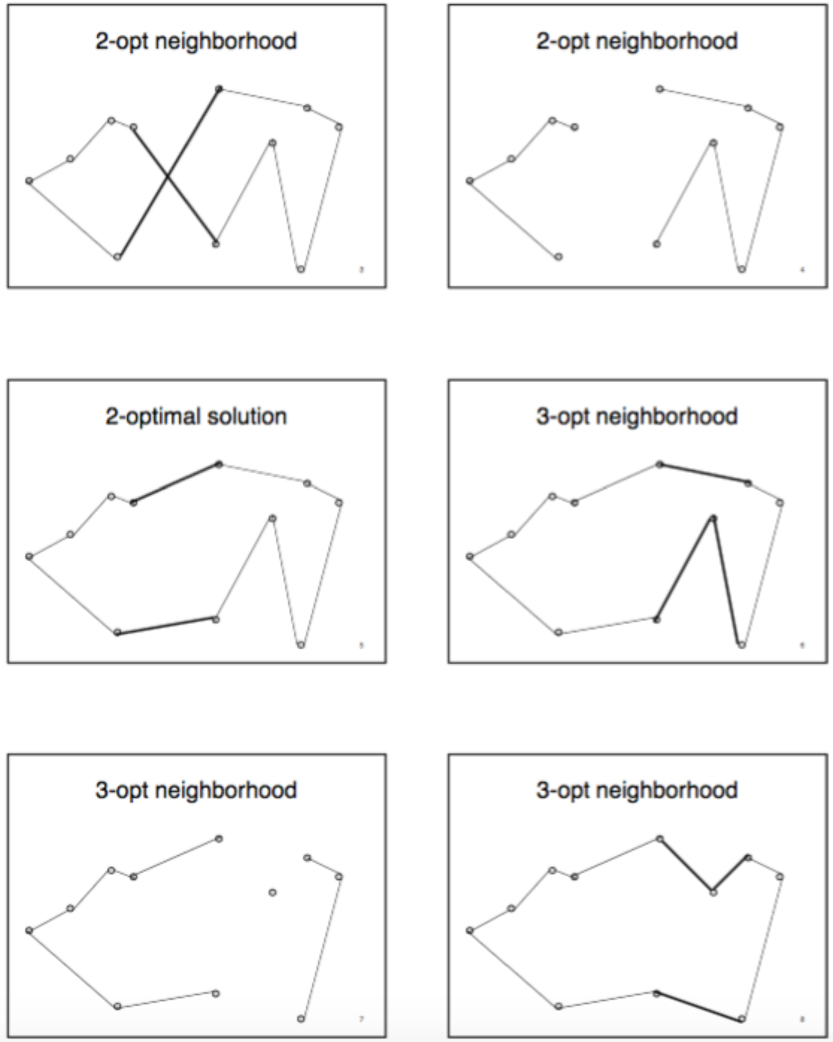


3-Opt local search

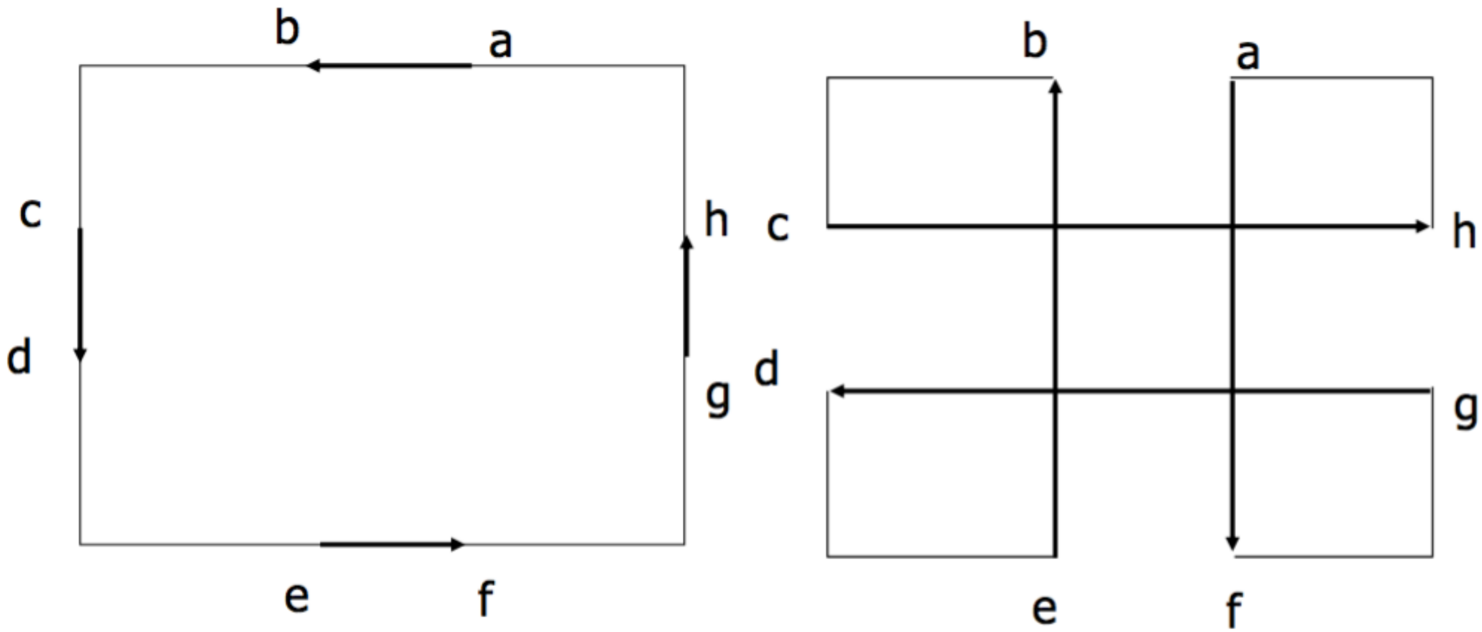


- Including the initial solution, as well as 2-opt moves, there is a total of 2^3 feasible rewirings for each selected triple of edges
- $n(n-1)(n-2) = O(n^3)$ successors
- One move does *not revert the path* $\rightarrow$ appropriate for *asymmetric* TSP

2-OPT VS. 3-OPT


2-opt neighborhood
2-opt neighborhood
2-optimal solution
3-opt neighborhood
3-opt neighborhood
3-opt neighborhood

4-OPT DOUBLE BRIDGE



- *Does not revert the tours*
- Computational complexity of a single step: $O(n^2)$
- Often used in conjunction with 2-opt and 3-opt

(SOME) PERFORMANCE COMPARISON

Average Percent Excess over the Held-Karp Lower Bound									
N =	10^2	$10^{2.5}$	10^3	$10^{3.5}$	10^4	$10^{4.5}$	10^5	$10^{5.5}$	10^6
	Random Euclidean Instances								
GR	19.5	18.8	17.0	16.8	16.6	14.7	14.9	14.5	14.2
CW	9.2	10.7	11.3	11.8	11.9	12.0	12.1	12.1	12.2
CHR	9.5	9.9	9.7	9.8	9.9	9.8	9.9	–	–
2-Opt	4.5	4.8	4.9	4.9	5.0	4.8	4.9	4.8	4.9
3-Opt	2.5	2.5	3.1	3.0	3.0	2.9	3.0	2.9	3.0
	Random Distance Matrices								
GR	100	160	170	200	250	280	–	–	–
2-Opt	34	51	70	87	125	150	–	–	–
3-Opt	10	20	33	46	63	80	–	–	–

D. Johnson and L. McGeoch, *The Traveling Salesman Problem: a case study in local optimization*, in Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra (editors), John Wiley and Sons, Ltd., 1997

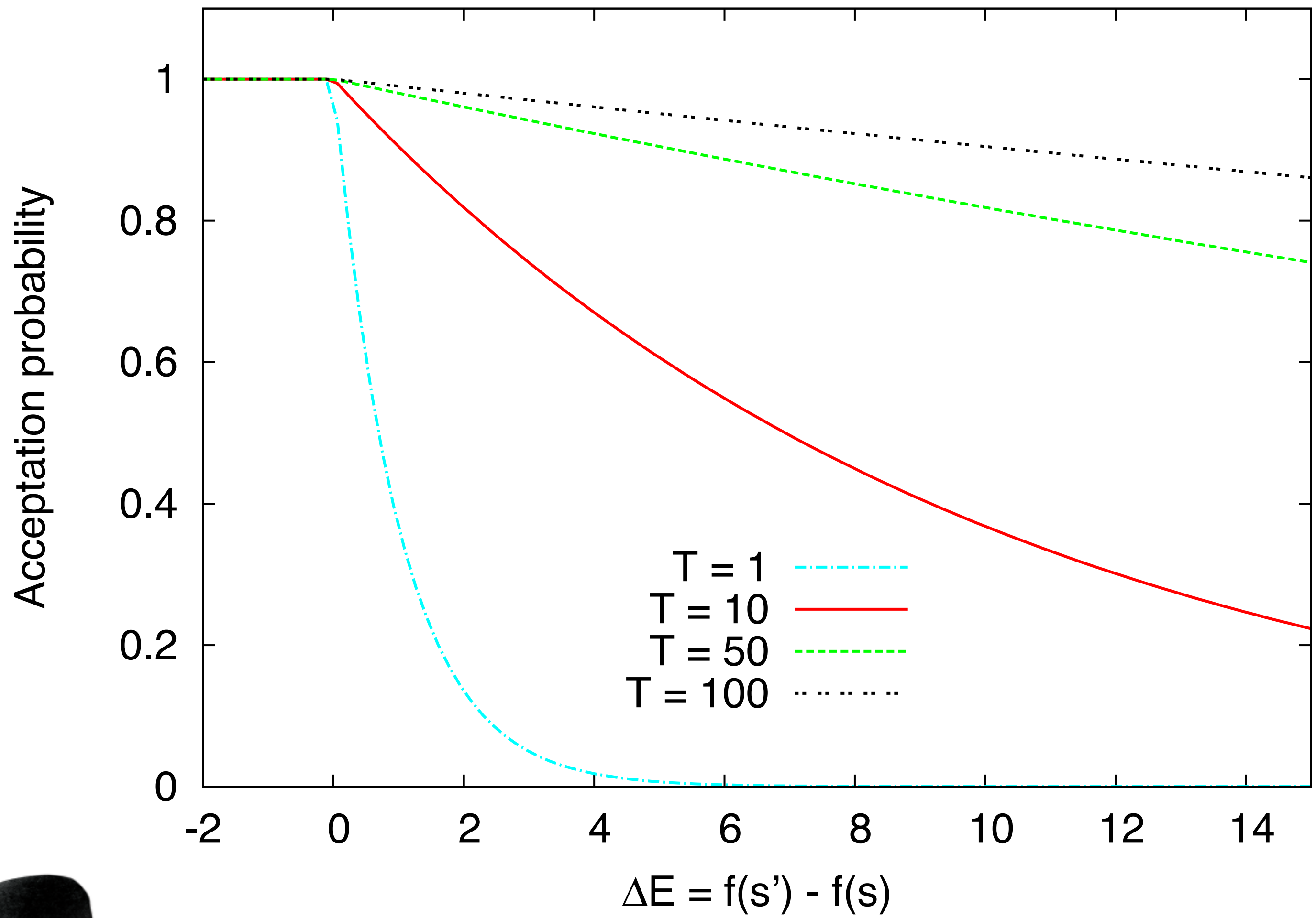
Simulated annealing

- *Escape from local optima* by accepting, with a probability that decreases during the search, also moves that are worse than the current solution (going downhill!)
- Stochastic, solution-improvement *metaheuristic for* global *optimization*
- Inspired by the process of *annealing* of solids in metallurgy:
  - The temperature of the solid is increased until it melts
  - The temperature is *slowly decreased through a quasi-static process* until the solid reaches a minimal energy state in which a regular crystal structure appears

SIMULATED ANNEALING

procedure Simulated_Annealing()
  $S = \{$set of all feasible solutions$\}$;
  $\mathcal{N} =$ neighborhood structure defined over S;
  $s \leftarrow$ Generate a starting feasible solution; // e.g., with a construction heuristic
  $s^{best} \leftarrow s$;
  $T \leftarrow$Determine a starting value for temperature;
  while (NOT YET *frozen*) // termination criterion
    while (NOT YET AT *equilibrium* FOR THIS TEMPERATURE)
      $s' \leftarrow$Choose a random solution from neighborhood $\mathcal{N}(s)$; // e.g., select a random 2-opt move
      $\Delta E \leftarrow f(s') - f(s)$;
      if $(\Delta E \leq 0)$ // downhill, locally improving move
        $s \leftarrow s'$;
        if $(f(s) < f(s^{best}))$
          $s^{best} \leftarrow s$;
      else // uphill move
        $r \leftarrow$Choose a random number uniformly from [0,1];
        if $(r < e^{-\Delta E/T})$ // accept the uphill, not improving, move
          $s \leftarrow s'$
      end if
  end while
  $T \leftarrow$Lower the temperature according to the selected cooling schedule;
end while
return s^{best};

Effect of temperature



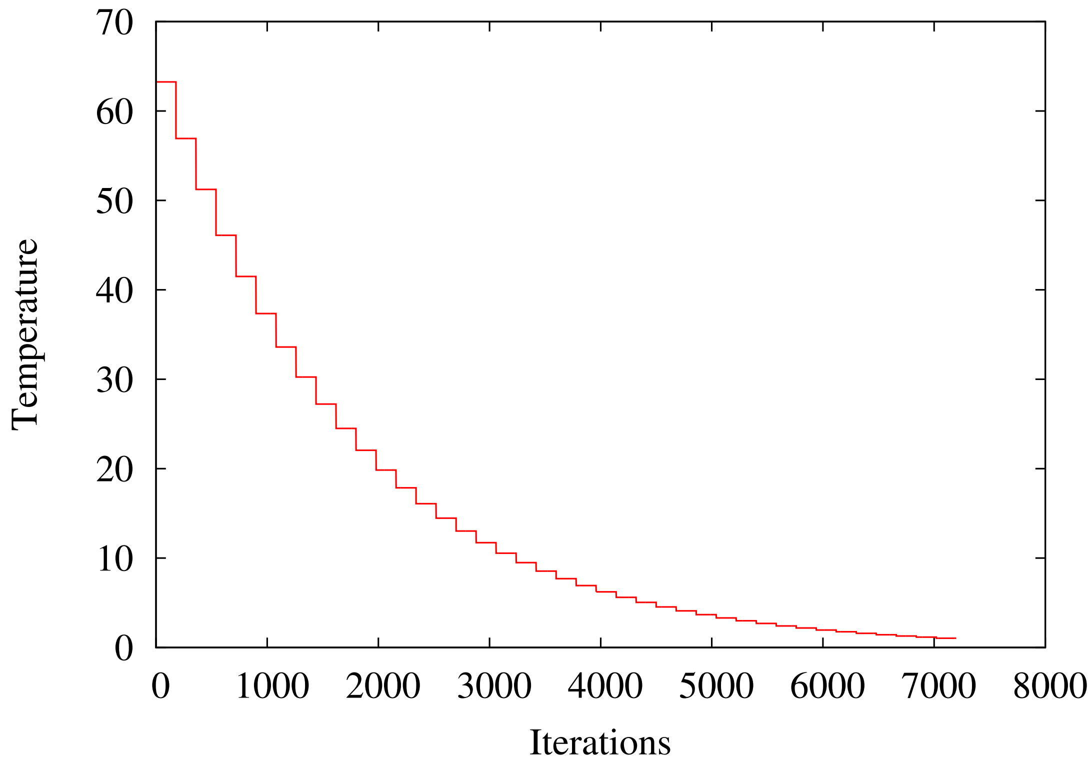
PROPERTIES

- Acceptation probability depends on the current candidate solution and on the previous one → The solution sequence can be seen as a *Markov chain*
- If T_k decreases "slowly enough" the algorithm will *asymptotically converge in probability to the global optimum* → Asymptotically complete and optimal
- *Convergence can be guaranteed* if at each step T drops no more quickly than $C/log\ n$, C=constant, n # of steps so far
- Cooling schedules that work in practice often lack of convergence properties :-(
- For TSP, n! solutions, the required # of iterations $k = O\left(n^{n^{2n-1}}\right)$

A popular temperature schedule: exponential cooling

- Temperature drops roughly as C^n, $C \in (0, 1)$
- A fixed number of moves is performed at each temperature, after which one arbitrarily declares *"equilibrium"* and reduces the temperature by a standard factor, $T_{k+1} = \gamma T_k$, $\gamma \in [0,1]$ is a constant ($\gamma = 0.95$ is a common choice)
- Under an exponential cooling regime, the temperature reaches values sufficiently close to zero after a *polynomially-bounded* amount of time and the *"frozen"* state can be declared

Exponential cooling



Start T?

Temperature length?

(some) Performance comparison

Variant		Random Euclidean Instances					
		Average Percent Excess			Running Time in Seconds		
Variant		10^2	$10^{2.5}$	10^3	10^2	$10^{2.5}$	10^3
SA_1 (Baseline Annealing)	$\alpha = 1$	3.4	3.7	4.0	12.40	188.00	3170.00
SA_1 + Pruning	$\alpha = 1$	2.7	3.2	3.8	3.20	18.00	81.00
SA_1 + Pruning	$\alpha = 10$	1.7	1.9	2.2	32.00	155.00	758.00
SA_2 (Pruning + Low Temp)	$\alpha = 10$	1.6	1.8	2.0	14.30	50.30	229.00
SA_2	$\alpha = 40$	1.3	1.5	1.7	58.00	204.00	805.00
SA_2	$\alpha = 100$	1.1	1.3	1.6	141.00	655.00	1910.00
2-Opt		4.5	4.8	4.9	0.03	0.09	0.34
Best of 1000 2-Opts		1.9	2.8	3.6	6.60	16.20	52.00
Best of 10000 2-Opts		1.7	2.6	3.4	66.00	161.00	517.00
3-Opt		2.5	2.5	3.1	0.04	0.11	0.41
Best of 1000 3-Opts		1.0	1.3	2.1	11.30	33.00	104.00
Best of 10000 3-Opts		0.9	1.2	1.9	113.00	326.00	1040.00
Lin-Kernighan		1.5	1.7	2.0	0.06	0.20	0.77
Best of 100 LK's		0.9	1.0	1.4	4.10	14.50	48.00

Suggestions for Further readings

M. Gendreau and J.-Y. Potvin (Editors), Handbook of Metaheuristics, Springer 2010

H. Hoos, T. Stueztle, Stochastic Local Search: Foundations & Applications, Morgan Kauffmann, 2004

E. Aarts and J. Lenstra (Editors), Local Search in Combinatorial Optimization, Princeton University Press, 2003