OPTIMIZATION TECHNIQUES

Iterated local search and simulated annealing

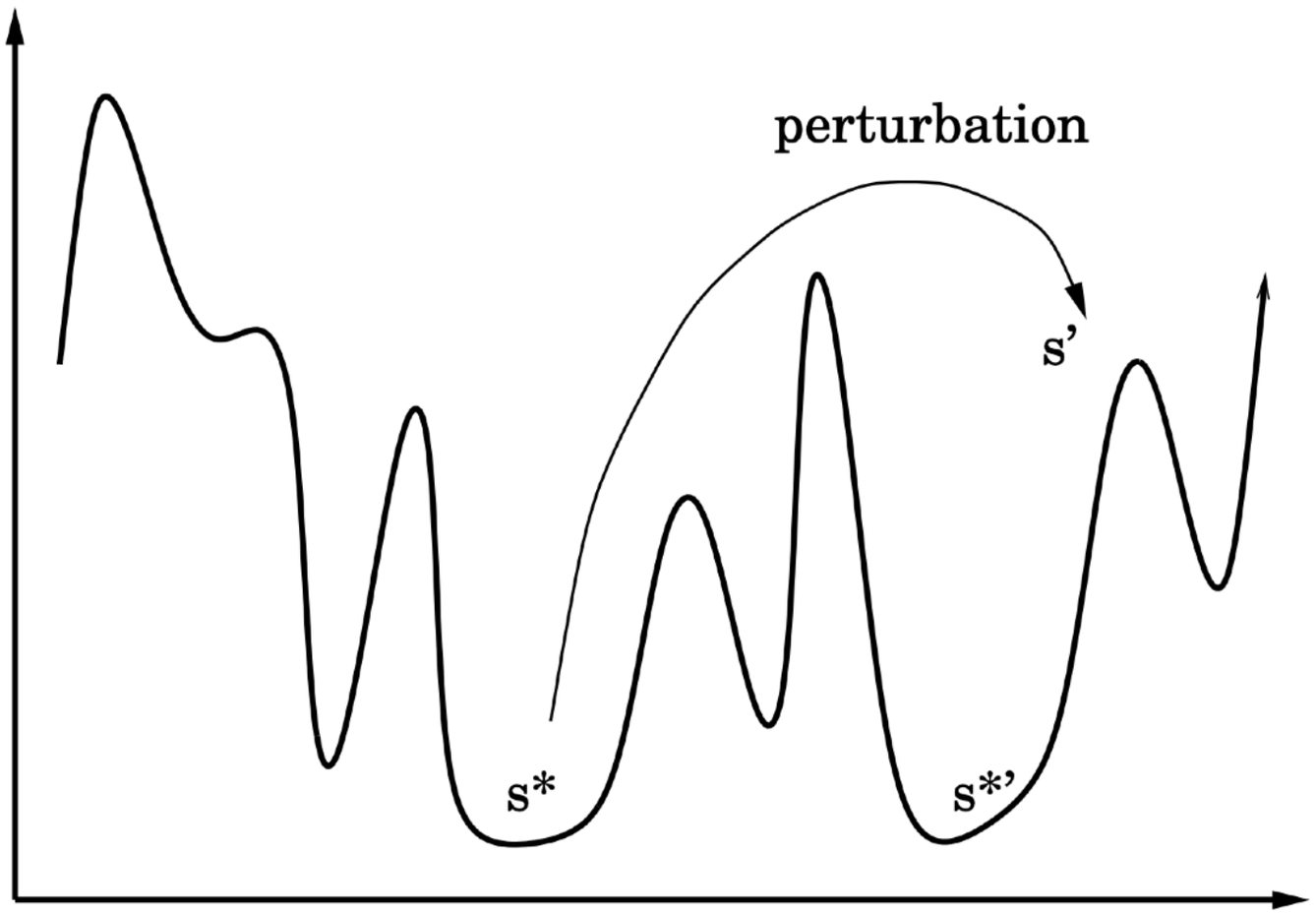
Prof. Giovanni Iacca

giovanni.iacca@unitn.it

UNIVERSITY OF TRENTO - Italy

Information Engineering
and Computer Science Department

Iterated Local Search



ITERATED LOCAL SEARCH

MAIN GIST OF THE ALGORITHM

- Iterated local search (ILS) is a stochastic local search method that generates a **sequence** of solutions generated by an **embedded heuristic**, leading to far better results than if one were to use repeated random trials of that heuristic.

procedure *Iterated Local Search*
  $s_0 \leftarrow$ GenerateInitialSolution
  $s^* \leftarrow$ LocalSearch(s_0)
  repeat
    $s' \leftarrow$ Perturbation$(s^*, history)$
    $s^{*\prime} \leftarrow$ LocalSearch(s')
    $s^* \leftarrow$ AcceptanceCriterion$(s^*, s^{*\prime}, history)$
  until termination condition met
end

ITERATED LOCAL SEARCH

RELATIONSHIP BETWEEN ILS AND VNS

- The two methods are actually very similar in many respects
- ILS appears to be in the literature more flexible in terms of interaction of the algorithmic modules
- VNS gives place to approaches like VND for obtaining more powerful local search approaches
- Many considerations hold true for both methods
  - E.g., how to initialize the first solution (greedy/construction heuristic vs random, etc.): for long runs greedy initial solutions appear to be recommendable, but for long runs dependence on should be very low
  - For long runs: effective diversification strategies result in much improved performance

ITERATED LOCAL SEARCH

PERTURBATION

- Important: strength of perturbation
  - Too strong: close to random restart → practically not very effective!
  - Too weak: **LocalSearch** may undo perturbation
- On many problems, small perturbations are sufficient
- Strength of perturbation may vary at run-time (similarly to VNS), or adapted dynamically depending on some conditions of the search process (*reactive search*)
- Perturbation should be complementary to **LocalSearch**

LOCAL SEARCH

- In principle, any LS method can be used, including short runs of Tabu Search, VNS, VND, Simulated Annealing (*see later*), etc.
- Often: the more effective the local search, the better ILS performs
- Sometimes: preferable to have fast but less effective LS (the tradeoff effectiveness vs efficiency of the LS procedure is a major aspect in ILS)

ITERATED LOCAL SEARCH

ACCEPTANCE CRITERION

- **AcceptanceCriterion** has strong influence on the effectiveness of the search process
- Controls balance between intensification and diversification
- Examples:
  - Better(*s**, *s*'*, *history*): accept *s*'* only if $f(s^{*'}) < f(s^{*})$ (extreme intensification)
  - RandomWalk(*s**, *s*'*, *history*): accept *s*'* always (extreme diversification)
- Many intermediate choices possible

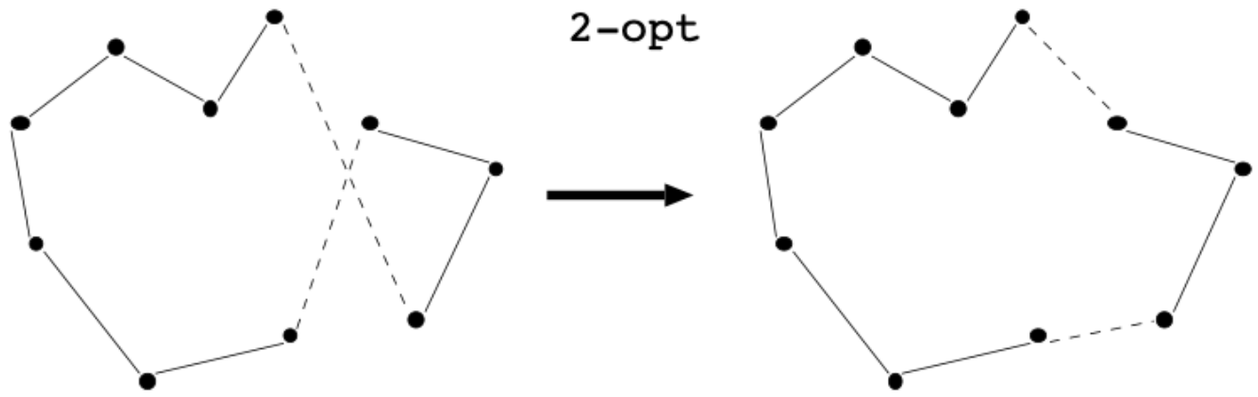
HISTORY

- Can be used in **Perturbation** and in **AcceptanceCriterion** in various ways
  E.g. Restart(*s**, *s*'*, *history*) to restart search if for a certain number of iterations no improved solution is found

ITERATED LOCAL SEARCH

EXAMPLE: TRAVELING SALESMAN PROBLEM (TSP)

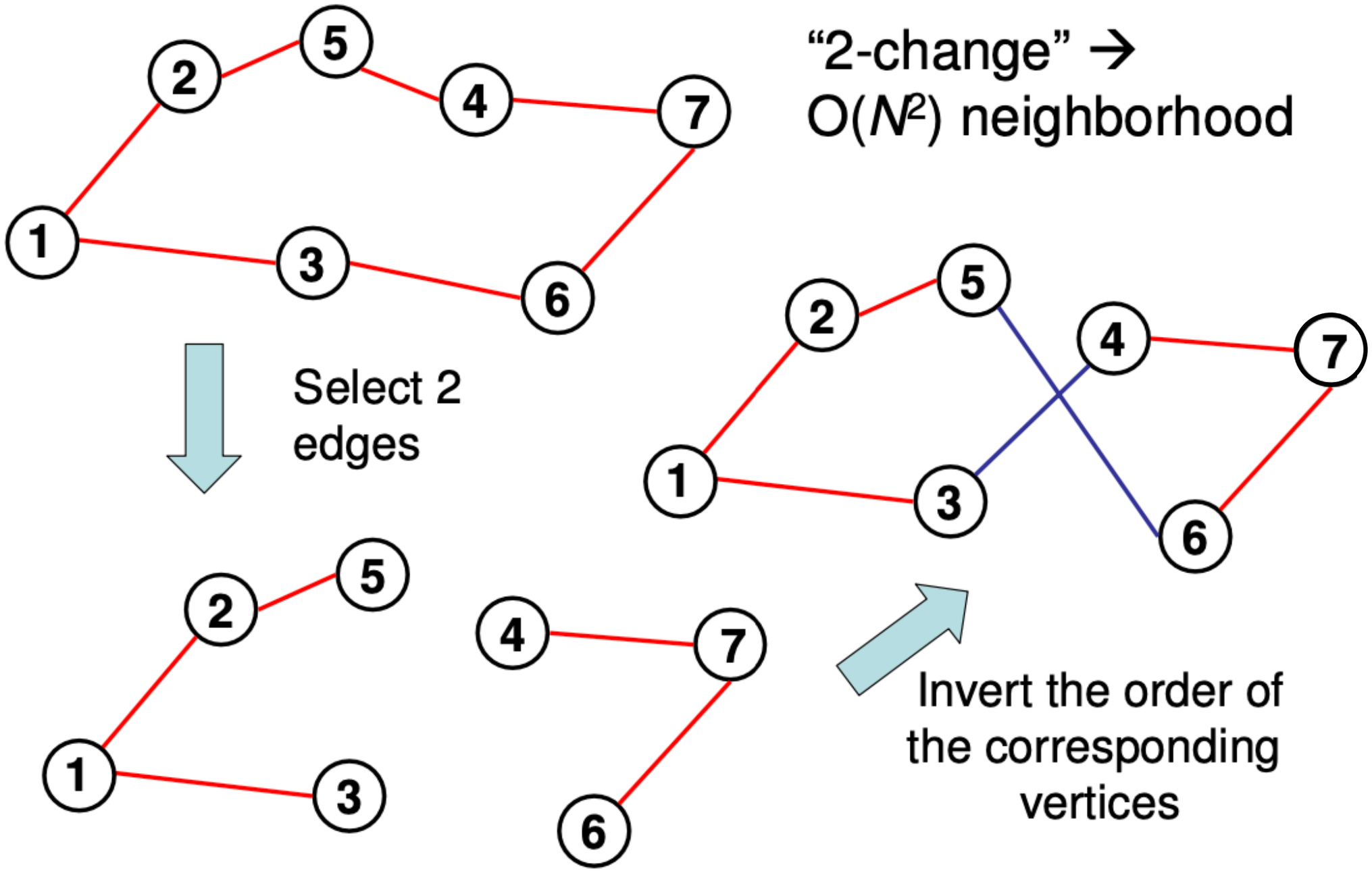
- **GenerateInitialSolution**: greedy heuristic
- **LocalSearch**: 2-opt (aka 2-change), 3-opt (aka 3-change), Lin-Kernighan (LK), or whatever available



- **Perturbation**: double-bridge move (a 4-opt move), small perturbations are known to be enough (this is because high-quality solutions are known to cluster, so good strategy needs intensification)
  → *remember the one of the observations behind VNS?*
- **AcceptanceCriterion**: accept $s^{*\prime}$ only if $f(s^{*\prime}) \leq f(s^{*})$

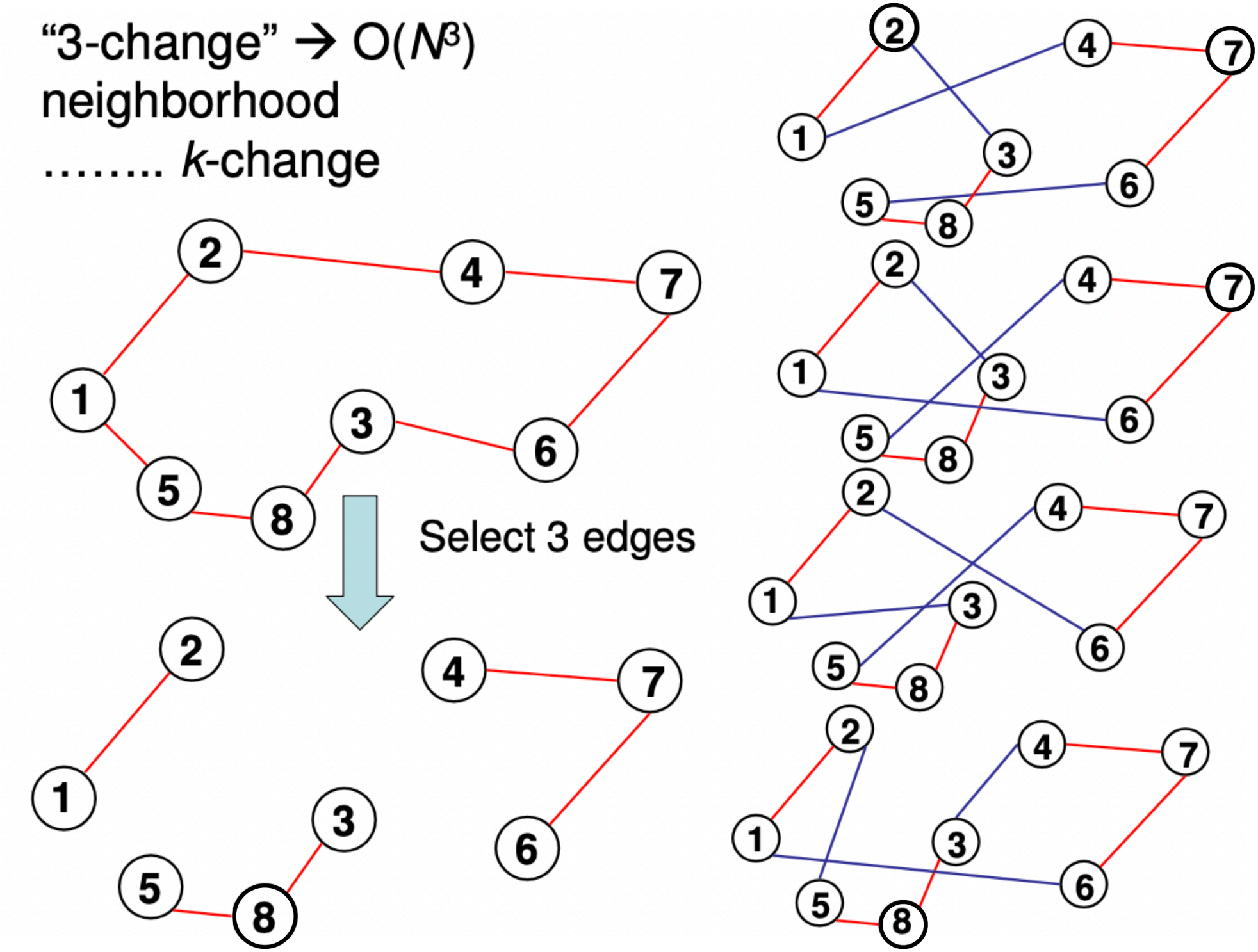
ITERATED LOCAL SEARCH

EXAMPLE: TRAVELING SALESMAN PROBLEM (TSP)



ITERATED LOCAL SEARCH

EXAMPLE: TRAVELING SALESMAN PROBLEM (TSP)



ITERATED LOCAL SEARCH

EXAMPLE: QUADRATIC ASSIGNMENT PROBLEM (QAP)

- Given *n* objects and *n* locations with:
  - a_{ij}: flow from object *i* to object *j*
  - d_{rs}: distance between location *r* and location *s*
- Goal: find an assignment (i.e., a permutation) of then *n* objects to the *n* locations that minimizes

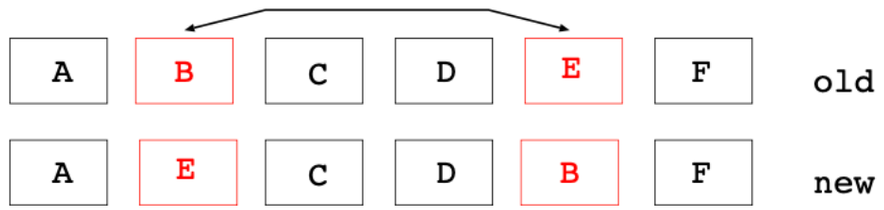
$$\min_{\pi \in \Pi(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} d_{\pi(i)\pi(j)}$$

  where π(i) gives the location of object *i*
- It is among the "hardest" combinatorial optimization problems; several applications (e.g. in distribution systems, networks, etc.)

ITERATED LOCAL SEARCH

EXAMPLE: QUADRATIC ASSIGNMENT PROBLEM (QAP)

- **GenerateInitialSolution**: random initial solution
- **LocalSearch**: 2-opt

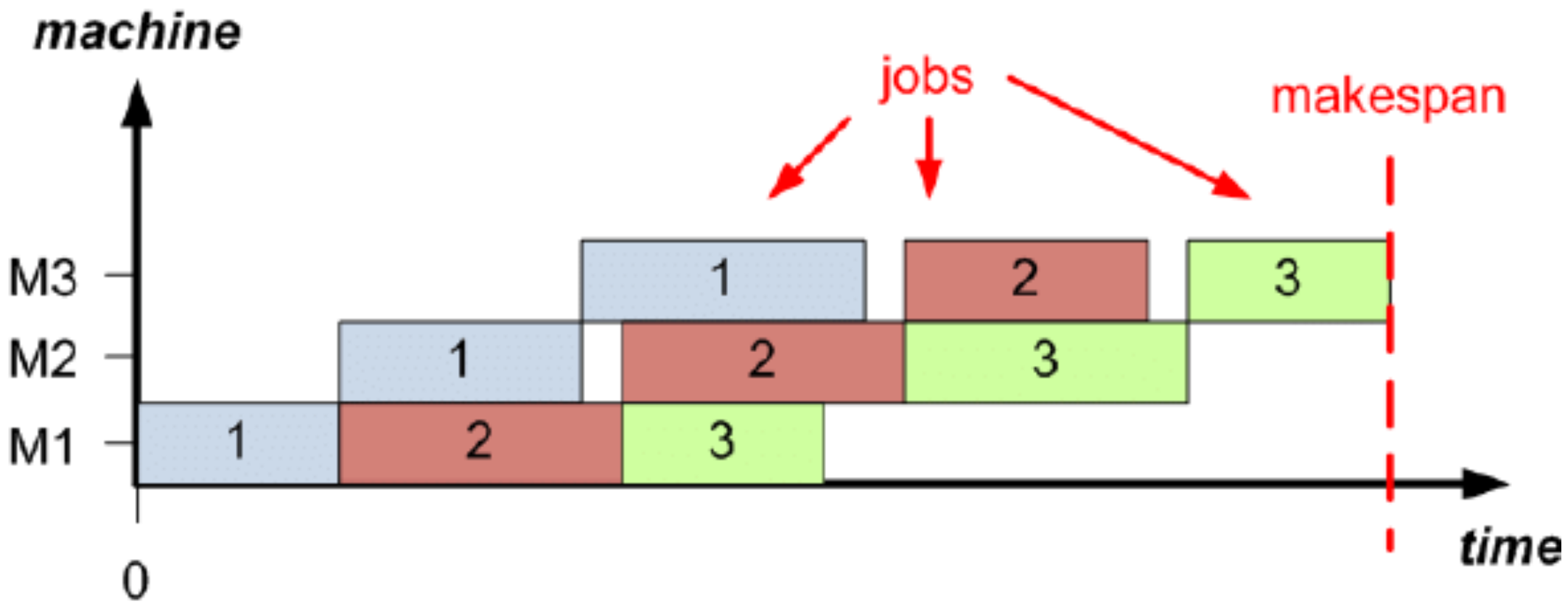


- **Perturbation**: random *k*-opt move, $k > 2$
- **AcceptanceCriterion**: accept *s*'* only if $f(s^{*'}) \leq f(s^{*})$

ITERATED LOCAL SEARCH

EXAMPLE: PERMUTATION FLOW-SHOP PROBLEM (FSP)

- Given:
  - n jobs to be processed on m machines
  - processing times t_{ij} of job i on machine j
  - machine order for all jobs is identical
  - permutation FSP: same job order on all machines (differently from JSP)
- Goal: minimize the completion time C_{max} of last job (makespan)
- Prototypical scheduling problem, NP-hard

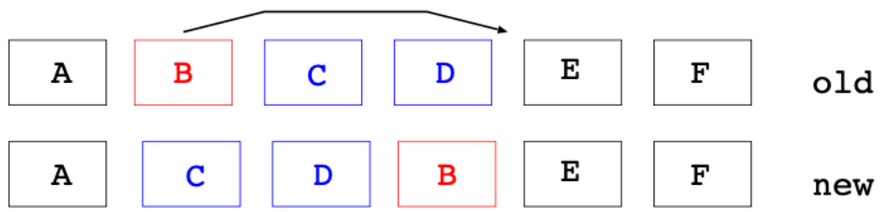


ITERATED LOCAL SEARCH

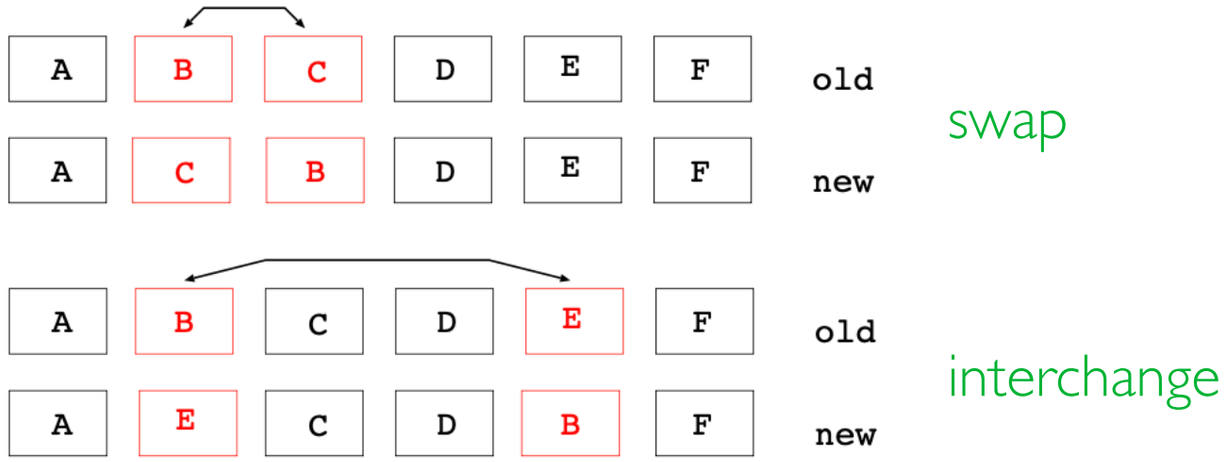
EXAMPLE: PERMUTATION FLOW-SHOP PROBLEM (FSP)

- **GenerateInitialSolution**: Nawaz, Enscore and Ham (NEH) heuristic (jobs with greater total processing time should be given a greater priority than jobs with a smaller total processing time)

- **LocalSearch**: insertion neighborhood



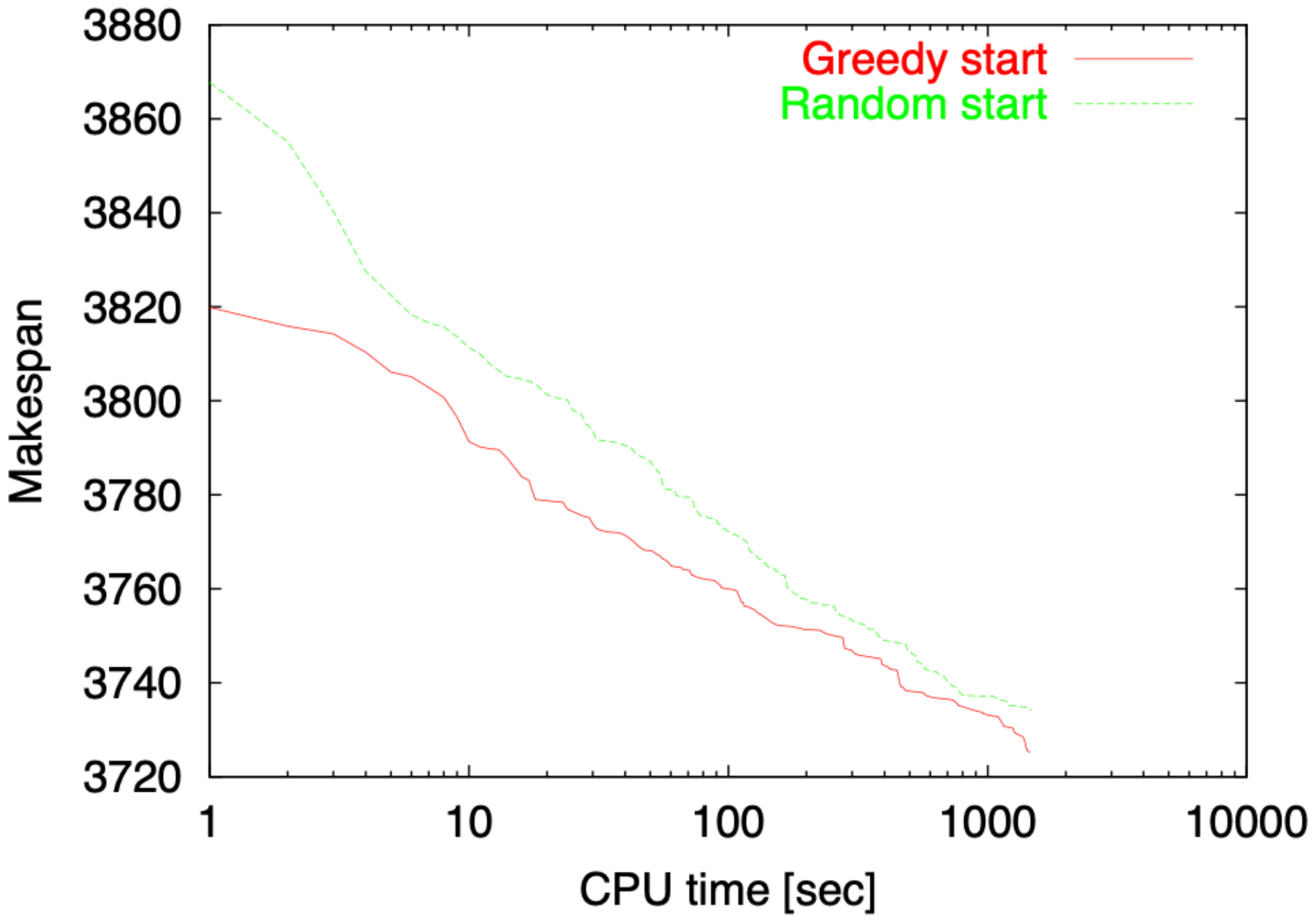
- **Perturbation**: a number of swap or interchange moves



- **AcceptanceCriterion**: accept $s^{*\prime}$ only if $f(s^{*\prime}) \leq f(s^{*})$

ITERATED LOCAL SEARCH

EXAMPLE: PERMUTATION FLOW-SHOP PROBLEM (FSP)



ITERATED LOCAL SEARCH

POSSIBLE IMPROVEMENTS/OPTIMIZATION OF ILS SCHEMES

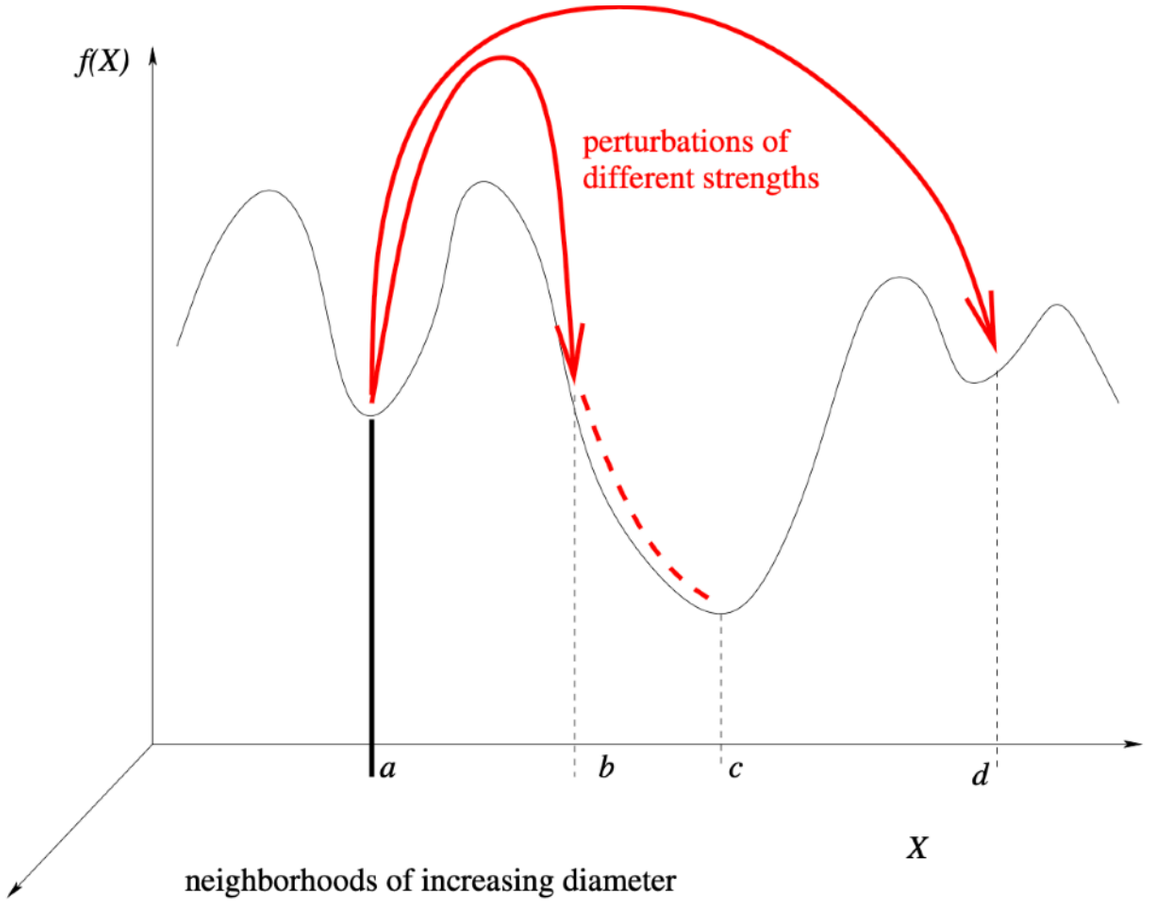
- The various ILS components (**GenerateInitialSolution**, **LocalSearch**, **Perturbation** and **AcceptanceCriterion**) can be optimized:
  - Either independently
  - Or, taking into account the complex interactions among them
  - Main dependencies:
    - **Perturbation** should not be easily undone by the **LocalSearch**; if **LocalSearch** has obvious shortcomings, a good perturbation should compensate for them.
    - Combination **Perturbation**-**AcceptanceCriterion** determines the relative balance of intensification and diversification; large perturbations are only useful if they can be accepted; in general, accepting several small perturbations may be better than accepting few large ones (remember: the balance intensification-diversification is very important!)
  - In both cases: optimization can be done through an iterative process (meta-learning)

- Global optimization of ILS components is complex, therefore often heuristic approach
- Global optimization of ILS components is important to reach peak performance
- Robustness (w.r.t. problems/problem instances) is another important issue

ITERATED LOCAL SEARCH

GUIDELINES FOR SETTING ILS UP

- **GenerateInitialSolution** should be to a large extent irrelevant for longer runs
- **LocalSearch** should be as effective and as fast as possible
- Best choice of **Perturbation** may depend strongly on **LocalSearch**
- Best choice of **AcceptanceCriterion** depends strongly on **Perturbation** and **LocalSearch**
- Particularly important can be interactions among perturbation strength and **AcceptanceCriterion**



ITERATED LOCAL SEARCH

APPLICATIONS

- First approaches by Baxter, 1981 to a location problem

- Most developed applications are those to TSP
  - Iterated descent (Baum, 1986)
    - First approach, relatively poor results
  - Large step Markov chains (Martin, Otto, Felten, 1991, 1992, 1996)
    - First effective ILS algorithm for TSP
  - Iterated Lin-Kernighan (Johnson, 1990, 1997)
    - Efficient ILS implementation
- Perturbation (Codenotti et.al, 1996)
  - Complex perturbation based on changing problem data
- Improved LargeStepMarkovChain (LSMC), similar to SA (Hong, Kahng, Moon, 1997)
  - Study of different perturbation sizes, acceptance criteria
- Implemented in Concorde TSP solver (Applegate, Bixby, Chvatal, Cook, Rohe, 1992-today)
  - Very fast LK implementation, publicly available, very large instances (25 million cities!)
- ILS with fitness-distance based diversification (Stützle, Hoos 1999–today)
  - Diversification mechanism in ILS for long run times
- ILS with genetic transformation (Katayama, Narisha, 1999)
  - Perturbation guided by a second solution

ITERATED LOCAL SEARCH

APPLICATIONS

- Several applications to scheduling problems
  - Single machine total weighted tardiness problem
    - Iterated search (Congram, Potts, Van de Velde, 1998)
    - ILS with VND local search (den Besten, Stützle, Dorigo, 2000)
  - Single and parallel machine scheduling
    - Several problems attacked (Brucker, Hurink, Werner 1996, 1997)
  - Flow-shop scheduling
    - Permutation flow-shop problem (Stützle, 1998)
    - Flow-shop problem with stages in series (Yang, Kreipl, Pinedo, 2000)
- Job-shop scheduling (JSP)
  - JSP with makespan criterion (Lourenço 1995; Lourenço, Zwijnenburg, 1996)
  - Guided LS extensions to JSP with makespan criterion (Balas, Vazacopoulos 1998)
  - Total weighted tardiness JSP (Kreipl, 2000)

ITERATED LOCAL SEARCH

APPLICATIONS

- Other applications
  - Graph partitioning (Martin, Otto, 1995)
    - Problem specific perturbation
  - Unweighted MAX-SAT (Battiti, Protasi, 1997)
    - A reactive search algorithm which fits into ILS framework; tabu search was used in perturbation phase; good performance also due to good tie-breaking criterion
  - Weighted MAX-SAT (Smith, Hoos, Stützle, 2002)
    - Tabu-type search in perturbation; at the time of publication, state-of-the-art results for MAX-SAT; a preliminary version won the "competition" in the Metaheuristics Network
  - Graph coloring (Chiarandini, Paquete, Stützle, 2001, 2002)
    - Very good performance over a wide range of instances
  - … and many more!

ITERATED LOCAL SEARCH

FURTHER READING

- Lourenço, Helena R., Olivier C. Martin, and Thomas Stützle. "Iterated local search." Handbook of metaheuristics. Springer, Boston, MA, 2003. 320-353.

- Lourenço, Helena Ramalhinho, Olivier C. Martin, and Thomas Stützle. "Iterated local search: Framework and applications." Handbook of metaheuristics. Springer, Cham, 2019. 129-168.

Simulated Annealing

SIMULATED ANNEALING

FROM GREEDY SEARCH TO SIMULATED ANNEALING

Most basic algorithm: Hill-Climbing/Descent (Greedy Local Search)

- $X \leftarrow$ Initial configuration
- Iterate:
  1. $E \leftarrow Eval(X)$
  2. $\mathcal{N} \leftarrow Neighbors(X)$
  3. For each X_i in $\mathcal{N}$
     $E_i \leftarrow Eval(X_i)$
  4. If all E_i's are higher than E
     Return X
     Else
     $i^* = \text{argmin}_i (E_i)$ $X \leftarrow X_{i^*}$ $E \leftarrow E_{i^*}$

SIMULATED ANNEALING

FROM GREEDY SEARCH TO SIMULATED ANNEALING

Stochastic Search: Randomized Hill-Climbing/Descent

- *X* ← Initial configuration
- Iterate:

1. *E* ← *Eval*(*X*)
2. *X'* ← one configuration randomly selected in *Neighbors* (*X*)
3. *E'* ← *Eval*(*X'*)
4. If *E'* < *E*

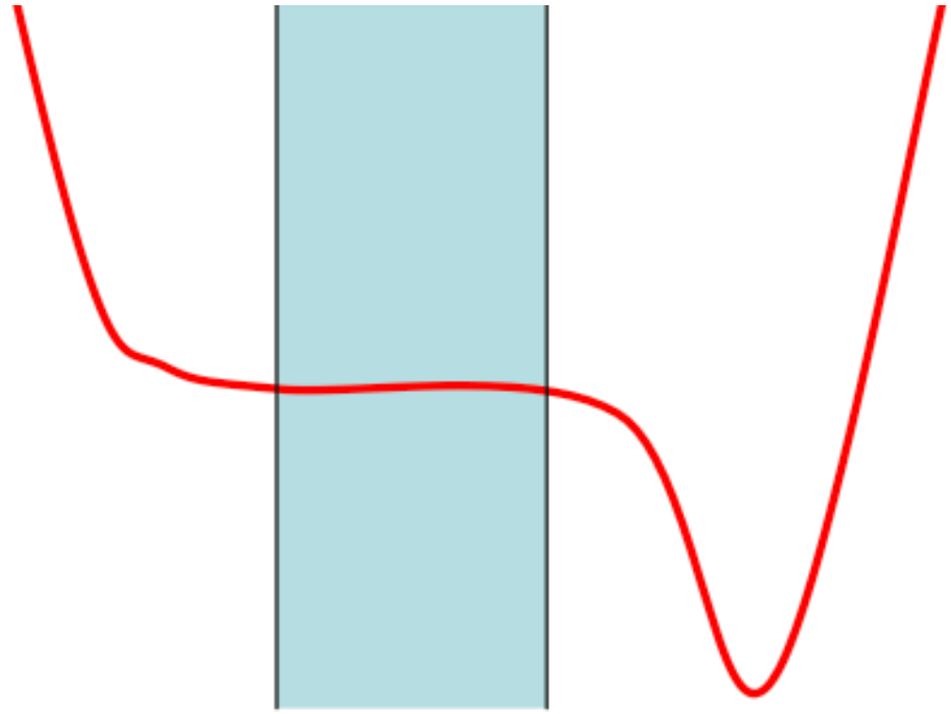
   X ← *X'*

   E ← *E'*

SIMULATED ANNEALING

ISSUES WITH GREEDY/RANDOMIZED HILL-CLIMBING

Highly multimodal landscape



Plateaus

SIMULATED ANNEALING

FROM GREEDY SEARCH TO SIMULATED ANNEALING

Probabilistic uphill (ascent) acceptance criterion

1. $E \leftarrow Eval(X)$
2. $X' \leftarrow$ one configuration randomly selected in *Neighbors* (X)
3. $E' \leftarrow Eval(X')$
4. If $E' \leq E$

   $X \leftarrow X'$

   $E \leftarrow E'$

Else accept the move to X' with some probability p:

   $X \leftarrow X'$

   $E \leftarrow E'$

How to set p?

If p constant: We don't know how to set p
→ Depends on the problem

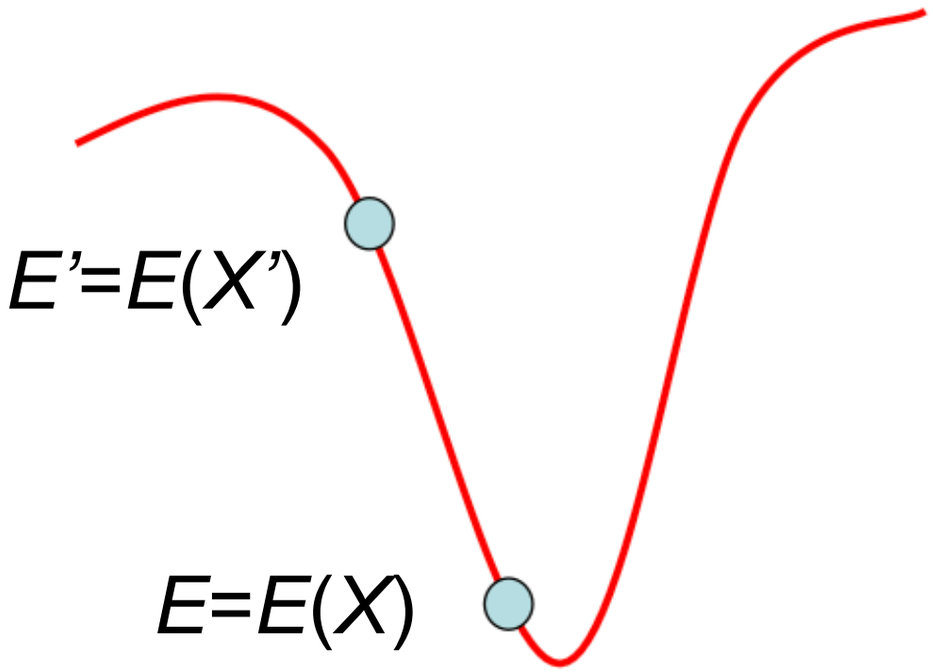
Decrease p as the iterations progress
→ We accept fewer uphill moves as we approach the global minimum

Decrease p as E'-E increases
→ Lower probability to move uphill if the difference on the evaluation function is high

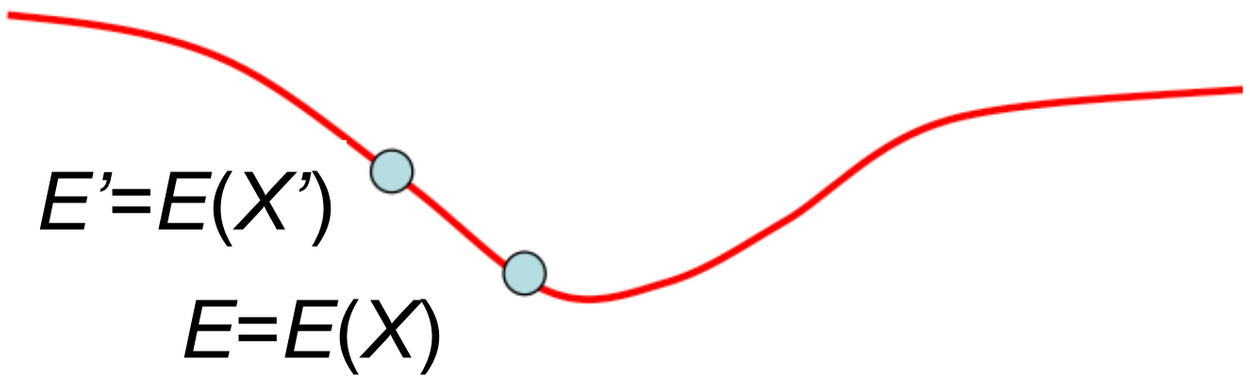
SIMULATED ANNEALING

FROM GREEDY SEARCH TO SIMULATED ANNEALING

Probabilistic uphill (ascent) acceptance criterion - Intuition



E'-E is **large**: It is more likely that we are around a (promising) "deep" minimum so we don't want to move uphill too much.



E'-E is **small**: It is likely that we are around a "shallow" minimum that is likely to be a (uninteresting) local minimum, so we would like to move uphill to explore other parts of the landscape.

SIMULATED ANNEALING

SIMULATED ANNEALING - GENERAL SCHEME

- $X \leftarrow$ Initial configuration
- $T \leftarrow$ Initial high temperature
- Iterate:

1. Do K times:
   1.1 $E \leftarrow Eval(X)$
   1.2 $X' \leftarrow$ one configuration randomly selected in *Neighbors* (X)
   1.3 $E' \leftarrow Eval(X')$
   1.4 If $E' \leq E$
   $X \leftarrow X'; E \leftarrow E';$
   Else accept the move with probability $p = e^{-(E'-E)/T}$
   $X \leftarrow X'; E \leftarrow E';$
2. $T \leftarrow \alpha T$

Many parameters need to be tweaked!!

SIMULATED ANNEALING

SIMULATED ANNEALING - PARAMETERS

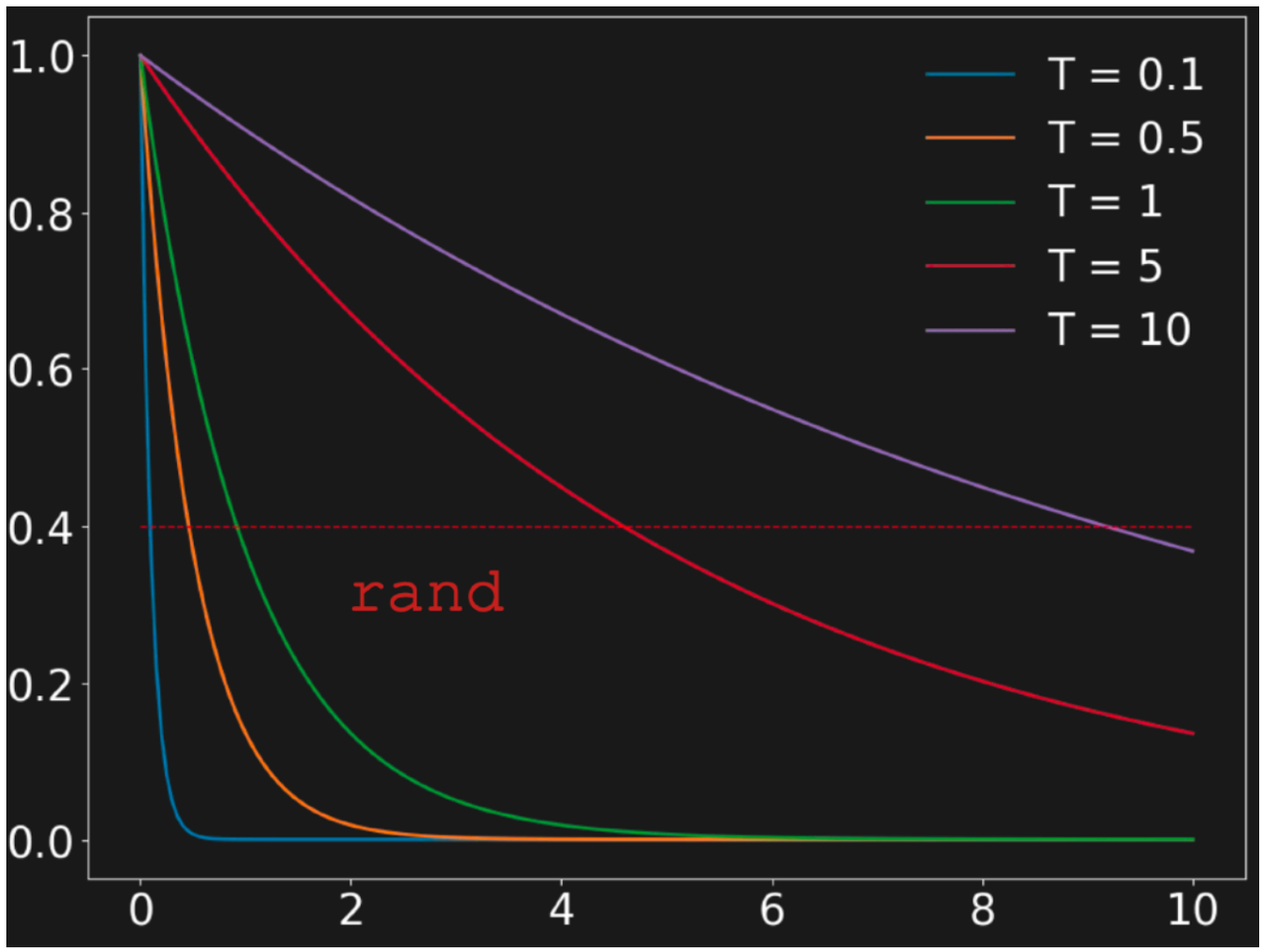
- *K*: no. of iterations at fixed "temperature"
- α: cooling scheduling parameter (a.k.a. Boltzmann-factor)

- $T = 0 \rightarrow$ Greedy hill climbing
- $T = \infty \rightarrow$ Random walk

SIMULATED ANNEALING

RECALL: METROPOLIS CRITERION (USED IN BASIN HOPPING)

A stochastic decision function with a "temperature" variable *T*.
Temperature is adjusted as a function of the number of iterations of the algorithm and/or the quality of the solution. This facilitates arbitrary solutions to be accepted early in the run, when the temperature is "high", while it enforces a stricter policy of just accepting solutions of improved quality later on in the search, when the temperature is "low".



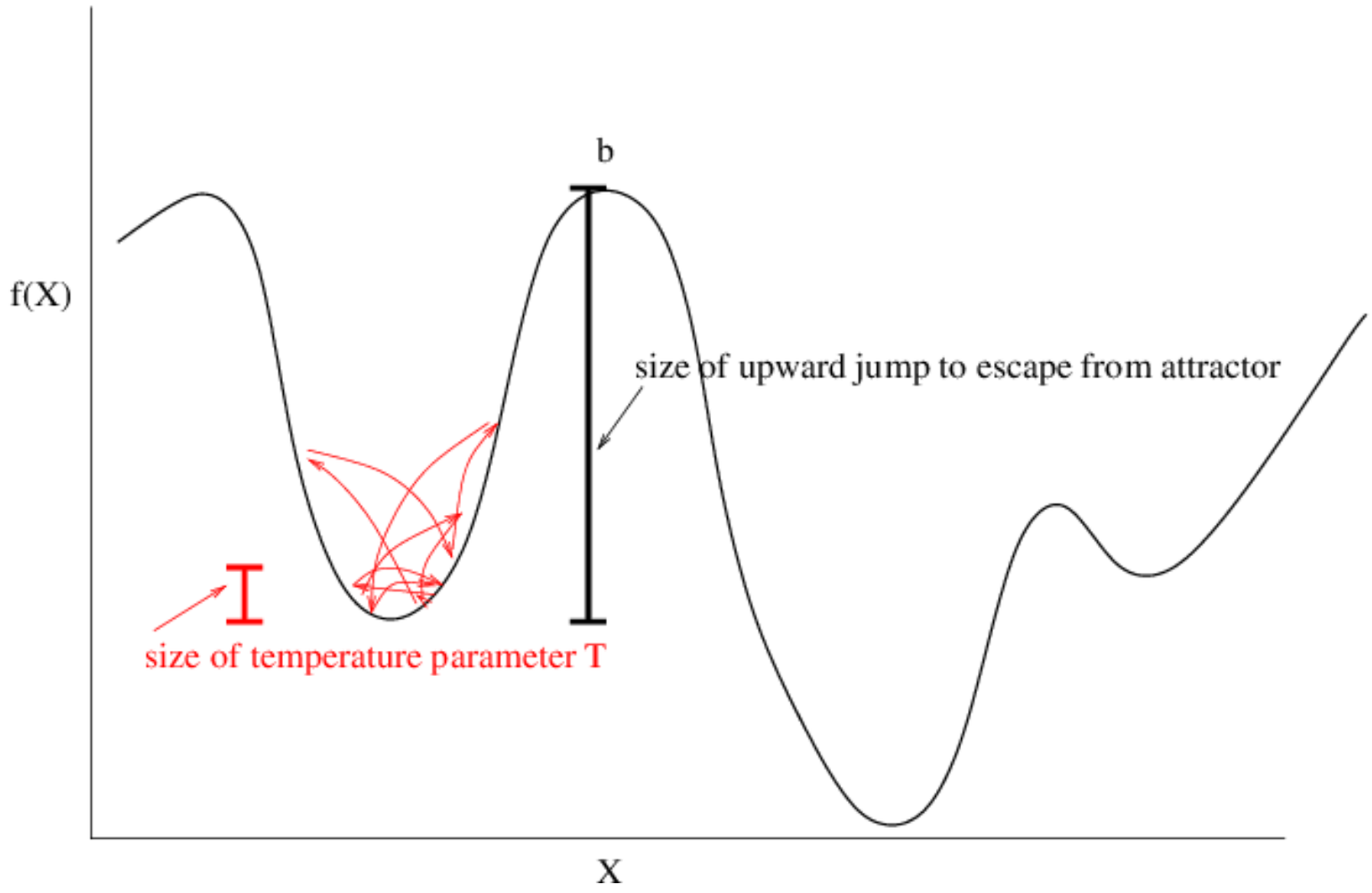
if $f(x_n) < f(x_{n-1})$
  accept x_n
else
  if $e^{-(f(x_n) - f(x_{n-1}))/T} \geq$ rand
    accept x_n
  else:
    reject x_n

NOTE: in BH, α=1,
T is fixed and predefined

SIMULATED ANNEALING

INTUITION

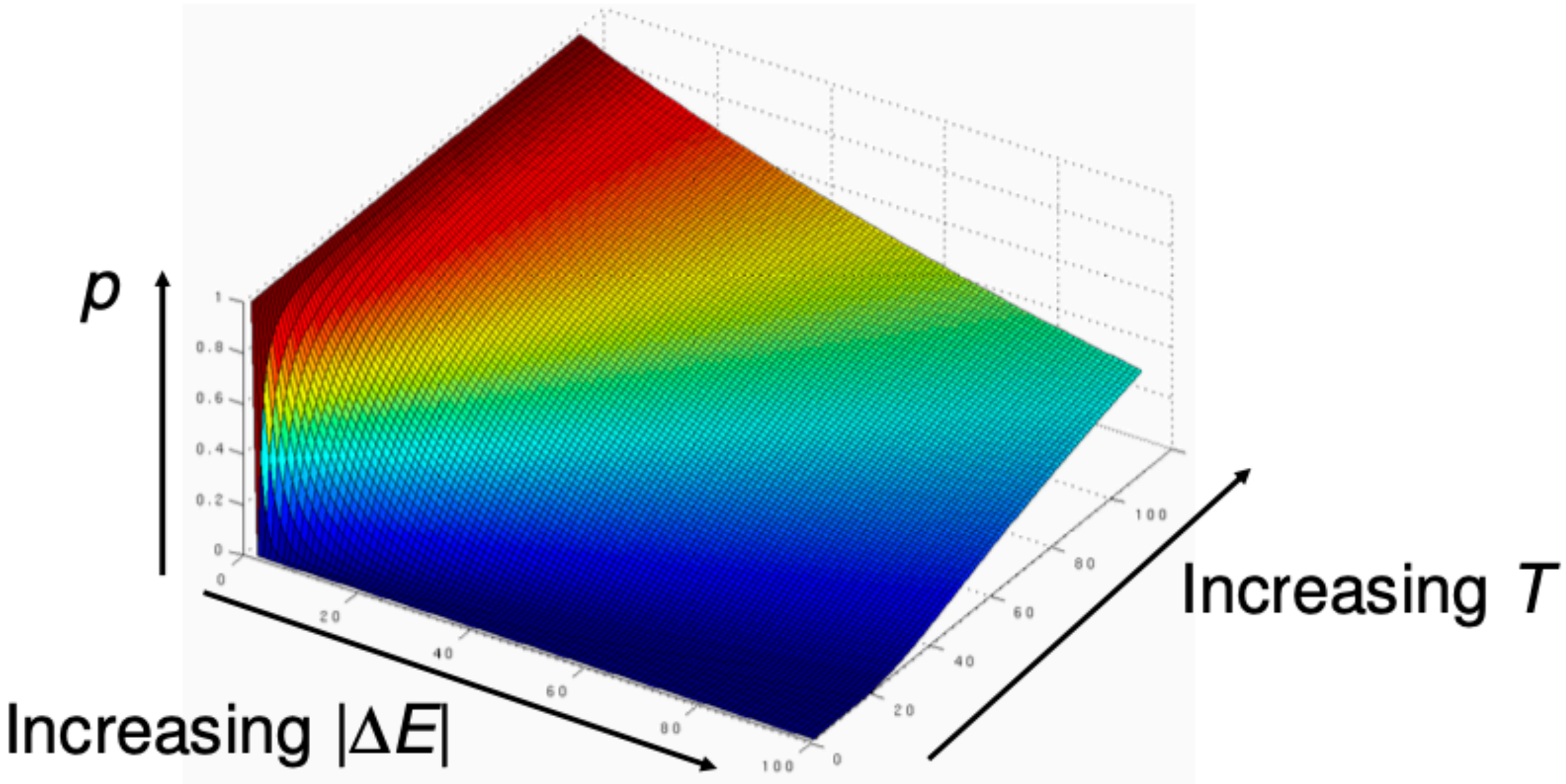
A finite "temperature" allows for controlled uphill steps thus enabling the search to get out of local minima. Starting from a high temperature, the search is then successively "cooled down" according to an annealing schedule. With decreasing temperature, the step width is therewith reduced thus freezing the system to the ground state. The cooling rate needs to be inverse logarithmic in time, to assure convergence.



SIMULATED ANNEALING

COOLING SCHEDULE

SA starts with high temperature *T* and decreases *T* gradually as iterations increase.



SIMULATED ANNEALING

MAIN ASPECTS OF THE ALGORITHM

- Introduced by Kirkpatrick et al. in 1983:
  S. Kirkpatrick, C. D. Gelatt Jr., and M. P.Vecchi. Optimization by simulated annealing. Science, 220:671–680, 1983.

- Often better than hill-climbing.

- Successful algorithm in many applications, many SA variants are possible.

- Essentially, a Markov process (i.e., a memory-less process) in which the trajectory is built in a randomized manner: the successor of the current point is chosen stochastically, with a probability that depends only on the current point and not on the previous history.

- For vanilla version of SA, convergence if:
  - Perform a large enough number of iterations (*K* "large enough")..
  - Decrease temperature slowly enough (α "close enough" to 1; otherwise there is a high risk of the method of being trapped into the region of attraction of a local minimum).
  - But, if not careful, we may have to perform an enormous number of evaluations.

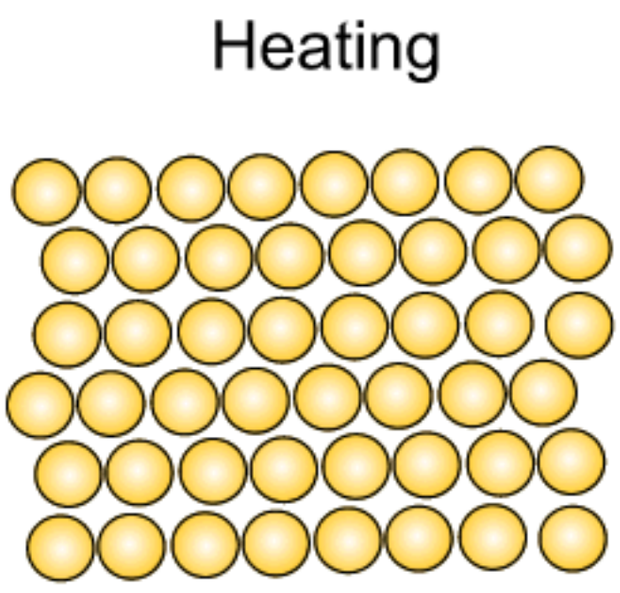
- Several parameters to tweak. If not careful, may require very large number of evaluations.

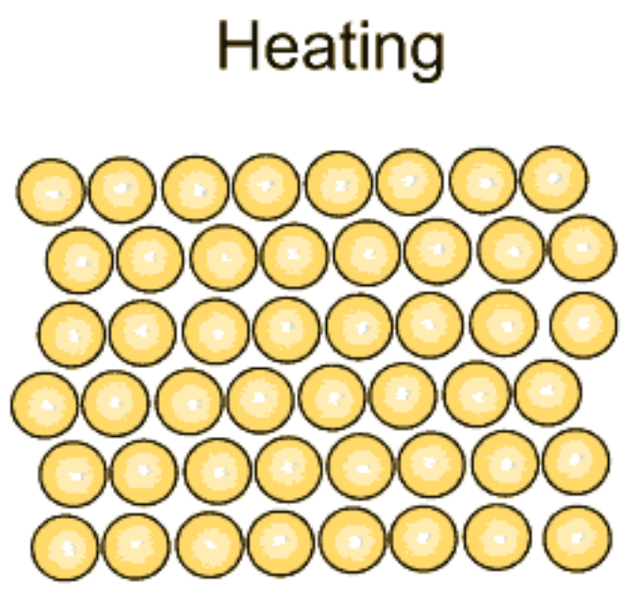
SIMULATED ANNEALING

THE ANNEALING ANALOGY - WHY DO WE TALK ABOUT TEMPERATURE?

The atoms in a metal form a crystal lattice structure. If the metal is heated, the atoms move around freely. As the metal cools down, the atoms slow down, and if the cooling is slow enough, they reconfigure into a minimum-energy state. Alternatively, if the metal is quenched (i.e., cooled rapidly), the metal recrystallizes with a different higher-energy state (called an amorphous metal).

Annealing: Heating a material above its recrystallization temperature, maintaining a suitable temperature for an appropriate amount of time, and then cooling. During annealing, atoms migrate in the crystal lattice and the number of dislocations decreases, leading to an increase in ductility and reduction of hardness (which makes metal material workable). As the material cools, it recrystallizes.





SIMULATED ANNEALING

THE ANNEALING ANALOGY - THE PHYSICS BEHIND IT

If the temperature of a solid is *T*, the probability of moving between two states of energy is:

$$e^{-\Delta Energy/\alpha T}$$

If the temperature *T* of a solid is decreased slowly, it will reach an equilibrium at which the probability of the solid being in a particular state is proportional to:

$$e^{-Energy(State)/\alpha T}$$

Boltzmann distribution → States of low energy (relative to *T)* are more likely

Analogy:
– State of solid ⇔ Configurations (i.e., problem solutions) *X*

– Energy ⇔ Evaluation function *Eval*(*X*)

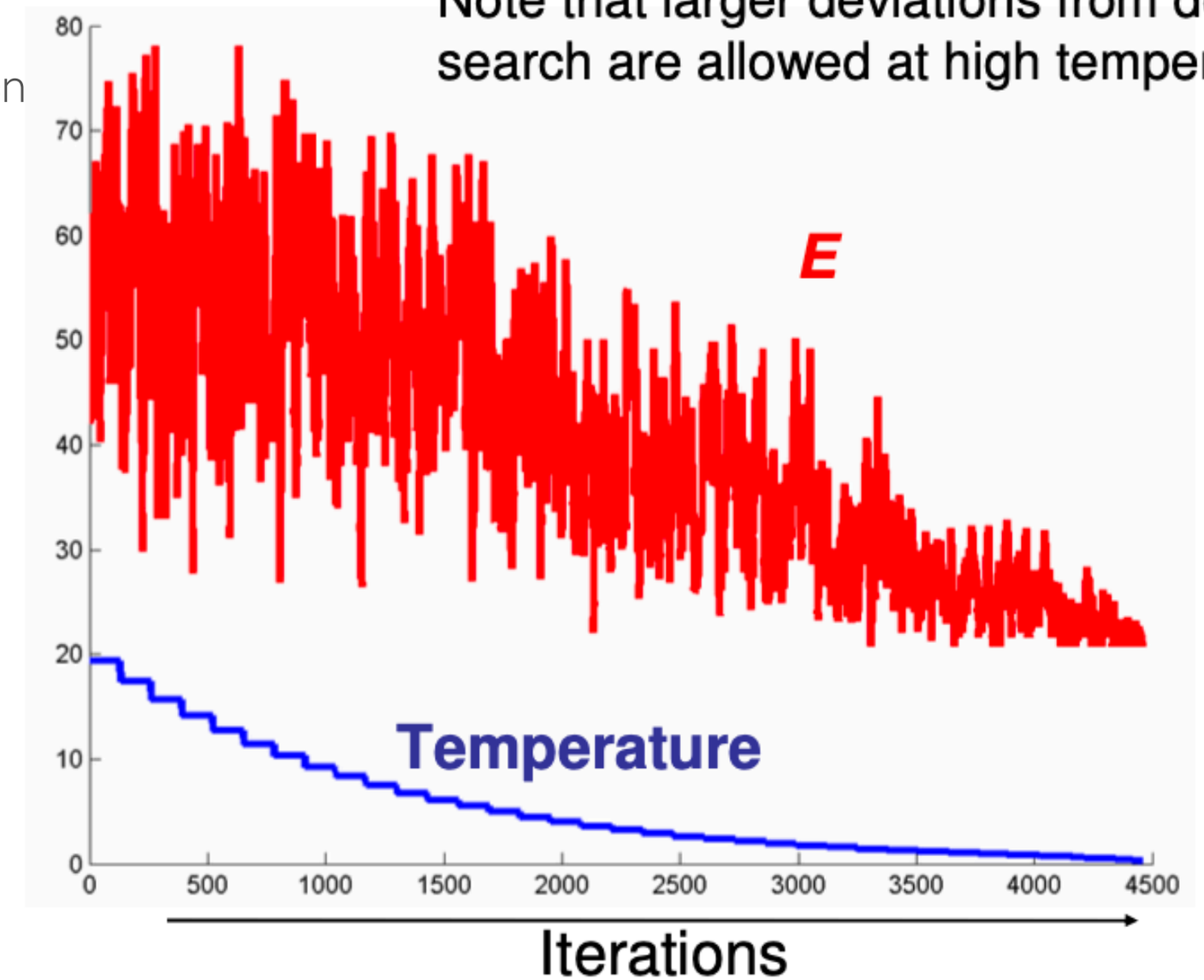
Reference:
N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth. A.H. Teller and E. Teller, Journal Chem. Phys. 21 (1953) 1087-1092

SIMULATED ANNEALING

EXAMPLE: TRAVELING SALESMAN PROBLEM (TSP)

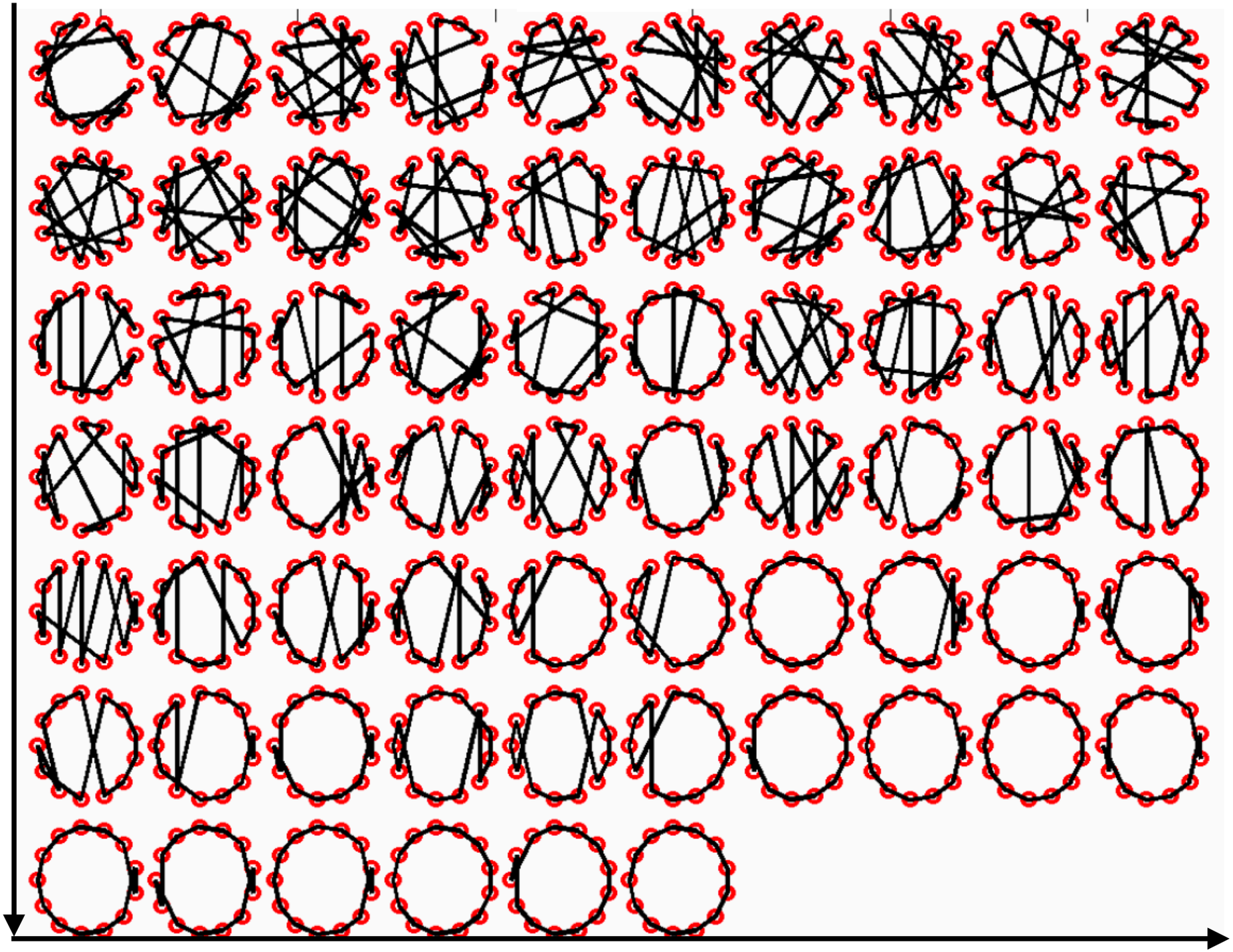
13 cities
Initial configuration
E(x) = 55

E(x) = 25



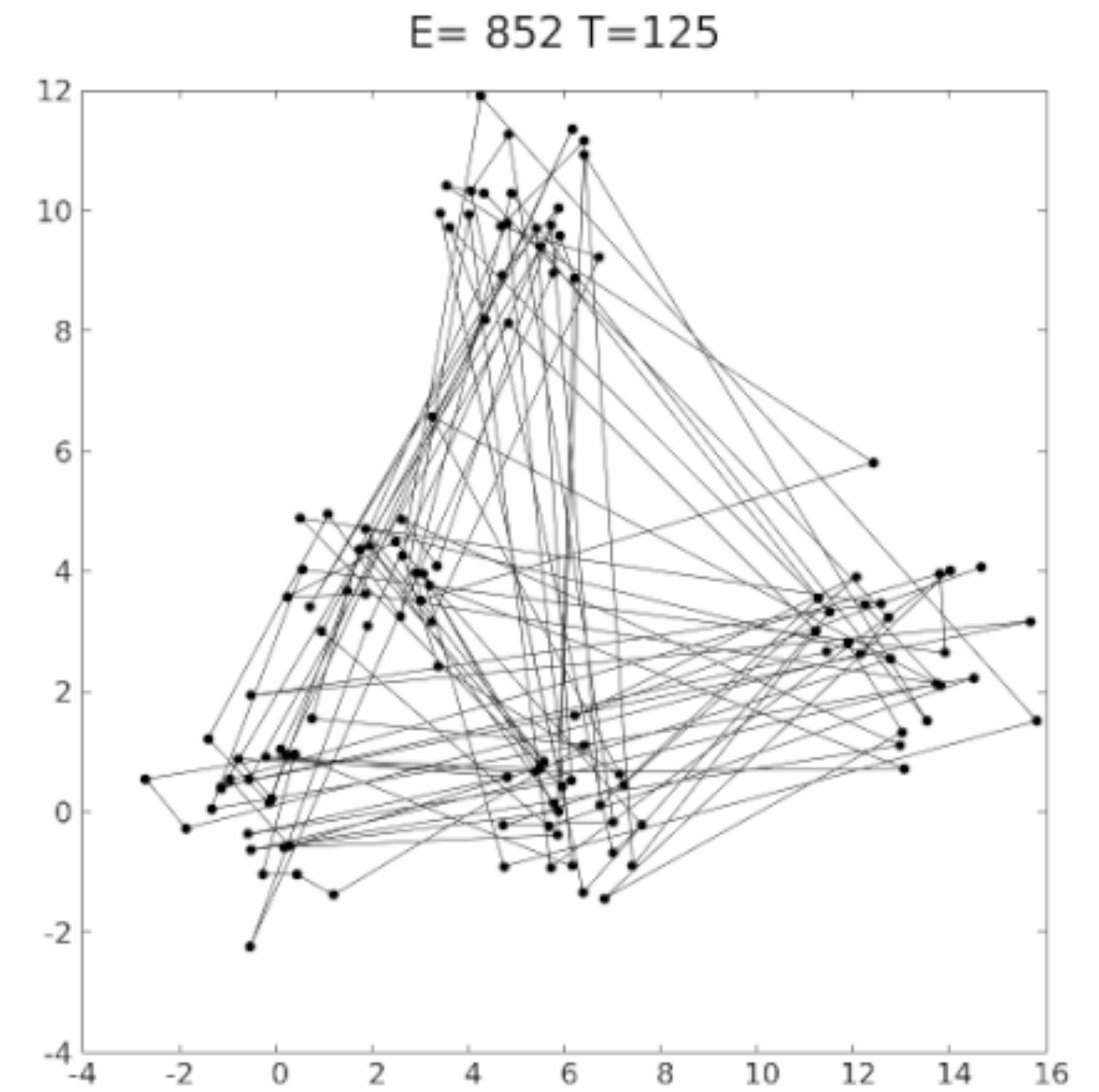
SIMULATED ANNEALING

EXAMPLE: TRAVELING SALESMAN PROBLEM (TSP)



SIMULATED ANNEALING

EXAMPLE: TRAVELING SALESMAN PROBLEM (TSP)



SIMULATED ANNEALING

ALGORITHMIC CHOICES (DIFFERENT CHOICES LEAD TO MANY SA VARIANTS)

- Design of neighborhood is critical.

- How to choose *K*? Typically related to size of neighborhood.

- How to choose α? Critical to avoid large number of useless evaluations. Especially a problem close to convergence (empirically, most of the time spent close to the optimum).

- How to choose starting temperature?
  Typically related to the distribution of anticipated values of ΔE.
  E.g., T_{start} = max{ΔE over a large sample of pairs of neighbors}.

- What if we choose a really bad starting *X*? Multiple random restart.

- How to avoid repeated evaluation? Use a bit more memory by remembering the previous moves that were tried (as in Tabu Search).

- Use (faster) approximate evaluation if possible.
  How? E.g. by using neural networks, RBF, K-nearest, etc.

SIMULATED ANNEALING

CONCLUDING REMARKS

- From a continuous optimization point of view, SA is nothing more than an acceptance rule to be applied within a general global optimization method based on local perturbations. In other words, SA can be seen as a sequential global optimization heuristic with a probabilistic acceptance rule.

- The use of Simulated Annealing for optimization has been motivated with asymptotic convergence results. When the number of iterations goes to infinity and the temperature is decreased slowly, the probability of observing a global optimum goes to one. However, this is not practical.

- SA can be successfully applied when the barriers separating the local minima are low with respect to the "depth" of the valleys where such local minima lie. However, in general, the time necessary to move uphill in order to escape from the region of attraction of a local minimum may too high, at least for hard global optimization problems.

SIMULATED ANNEALING

CONCLUDING REMARKS

- In modern global optimization methods, the importance of SA is less relevant than it used to be in the '80. The most successful applications are those in which the criterion of SA is used within, e.g. as we saw in Basin Hopping.

- One has to abandon vanilla "Markov-process" SA in favor of more pragmatic versions which estimate progress, determine a possible entrapment, and activate direct ways of escaping, e.g., by raising the temperature (re-heating), or just by restarting the search from scratch. In these cases the convergence conditions are not valid anymore, as theoretical analysis becomes very complex if not impossible, but better local optima can be determined in most practical cases.

Questions?