Zürcher Hochschule
für Angewandte Wissenschaften
School of
Engineering
IAMP Institut für Angewandte
Mathematik und Physik

MSE Algorithms

L03: Local Search and Randomized Methods

Samuel Beer

Course Overview

Zürcher Hochschule
für Angewandte Wissenschaften
School of
Engineering
IAMP Institut für Angewandte
Mathematik und Physik

1. Introduction: Basic problems and algorithms
2. Constructive methods: Random building, Greedy
3. **Local searches**
4. **Randomized methods**
5. **Threshold accepting, Simulated Annealing**
6. Decomposition methods: Large neighborhood search
7. Learning methods for solution building: Artificial ant systems
8. Learning methods for solution improvement: Tabu search
9. Methods with a population of solutions: Genetic algorithms
10. Application: Santa' Challenge

Lecture 3:
Local Search, Randomized methods

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

Goal:

Know various methods to improve an existing solution.

- Local Search (Local Improvements)
- Randomized Local Search
- Simulated Annealing
- Other Randomized Search Methods

Recap: Previous Lecture

Meta-Heuristics for Optimization Problems:

1. **Random Sampling** are easy to implement and can generate solutions very fast, but usually with bad quality

2. **Greedy Methods** can create an initial solution with reasonable good quality (e.g. Nearest Neighbor or Best Insertion for TSP)

3. **Look-Ahead Methods** (e.g. Pilot Method, Beam Search) can improve greedy methods

- UNTIL NOW: **Create** a first solution

- TODAY: **Improve** an existing solution

Local Search

Meta-Heuristic: Local Search

Start with a **given solution** (obtained e.g. with a constructive method).

Repeat

```
Try to find a modification that improves the solution

If such a modification is found, then apply it
```

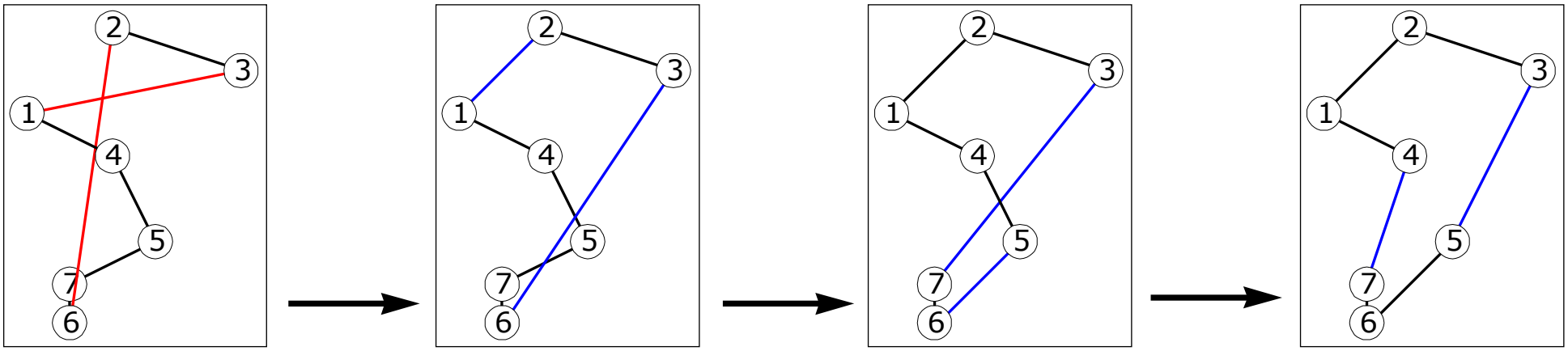
While An improvement is found

The set of all potential next solutions which can be achieved by a specific type of modifications (sometimes called moves) is called a **Neighbourhood** of the current solution.

Example of Local Search for TSP

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

- Idea: replace 2 edges of the tour by 2 others
- New solution is very similar to old one (is "in the neighbourhood")
- This is called "*2-opt Neighbourhood*"



Remarks

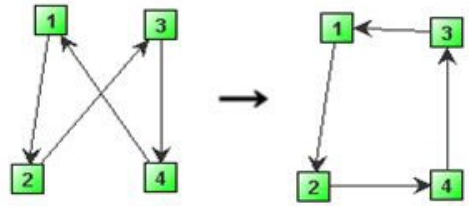
- 2-opt Neighbourhood for TSP is not only for "crossing lines". Instead, one tries for ALL pairs of two edges to check whether replacing them by two other edges improves the solution.

- 2-opt Neighbourhood for TSP is equivalent to the set of moves reversing a subtrail of the tour.

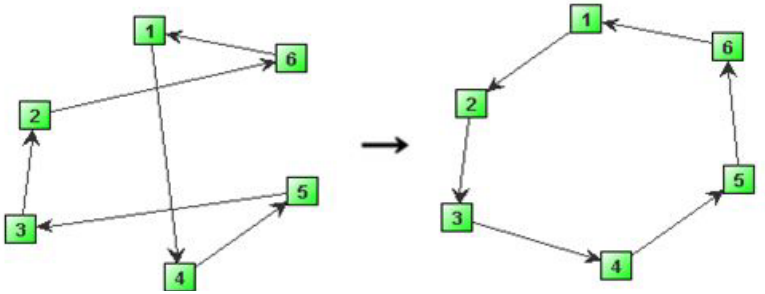
More Neighbourhoods for TSP

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

2-opt: Replace 2 edges by 2 other edges
→ Reverses the visiting order in a sub-chain.



3-opt: Replace 3 edges by 3 other edges
→ Moves a sub-chain somewhere else in the tour.



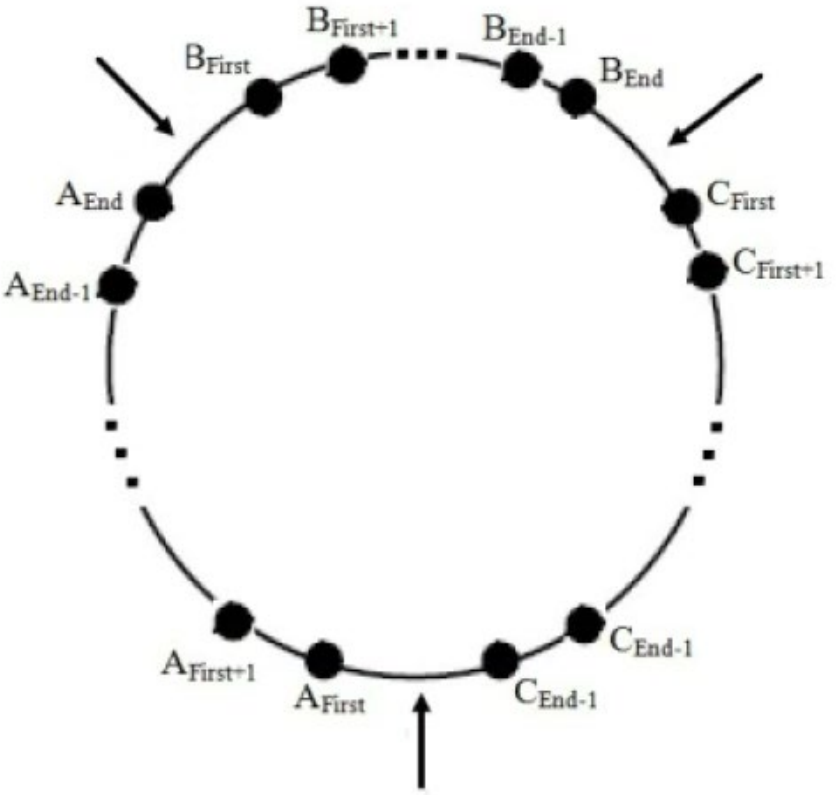
4-opt: Replace 4 edges by 4 other edges
→ Exchanges two sub-chains in the tour.

k-opt: Same as above but with *k* edges.

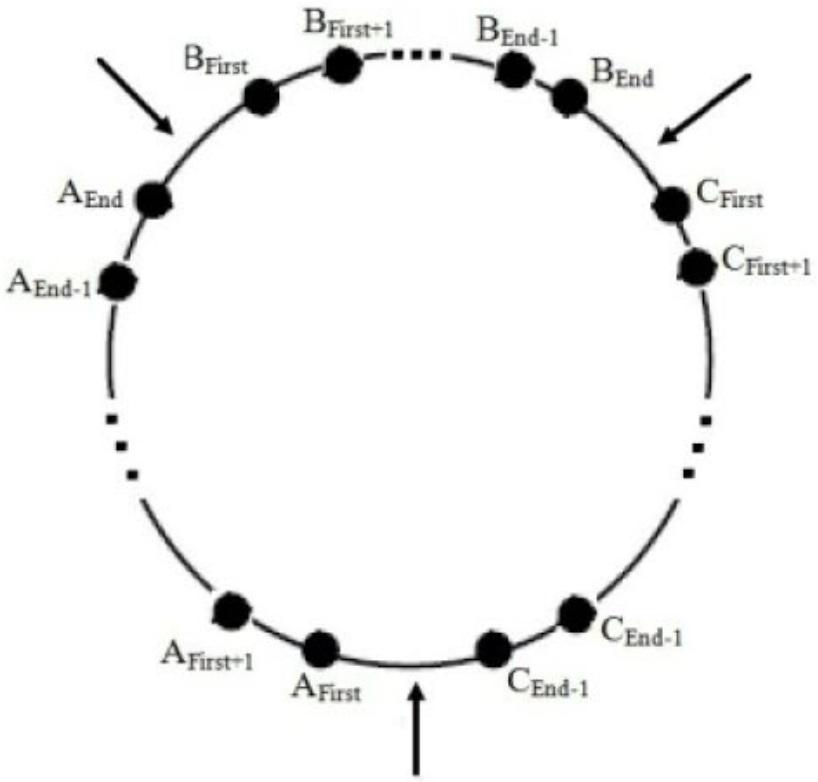
Or-opt: Move a sub-chain of *k* vertices somewhere else in the tour (3-opt with restricted length)

Excercise: 3-Opt

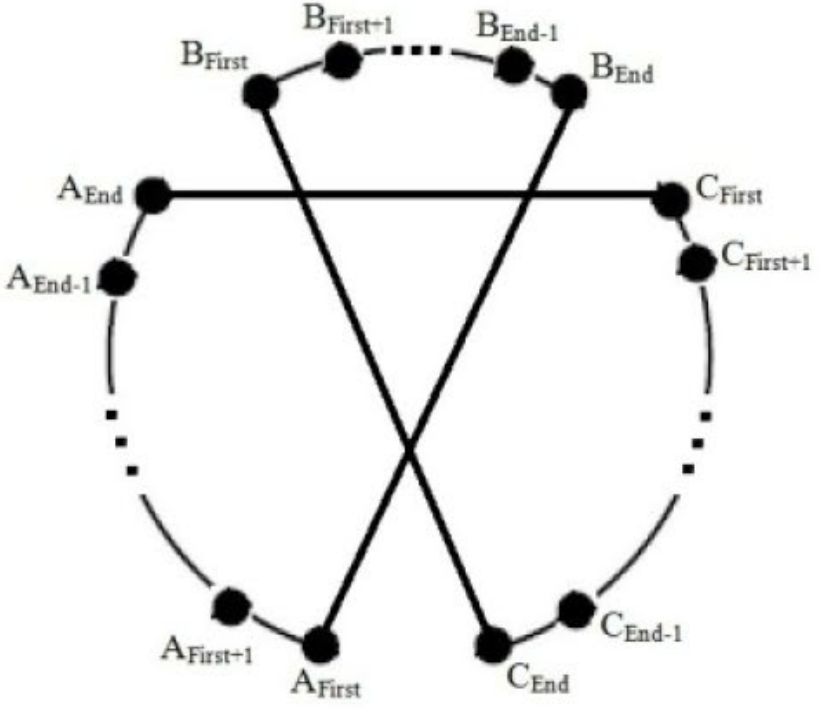
For the tour to the right, apply 3-Opt for the three edges that are marked with arrows.



SOLUTION: One way to apply 3-Opt

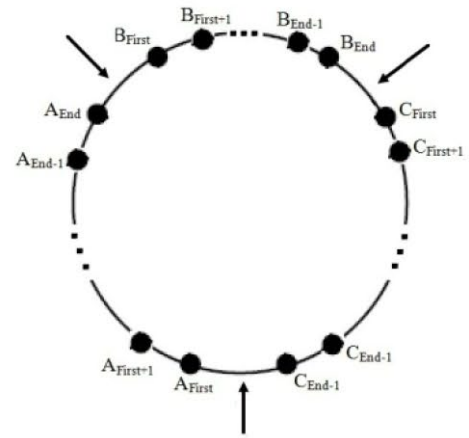


Original tour : ABC

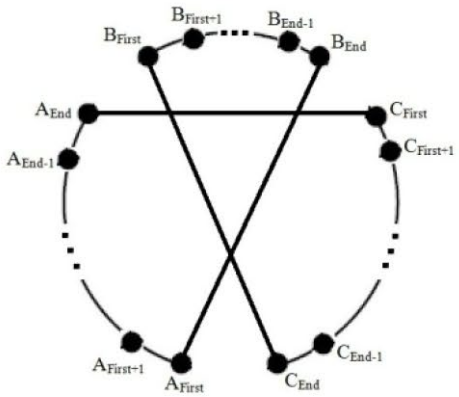


Method 1. ACB

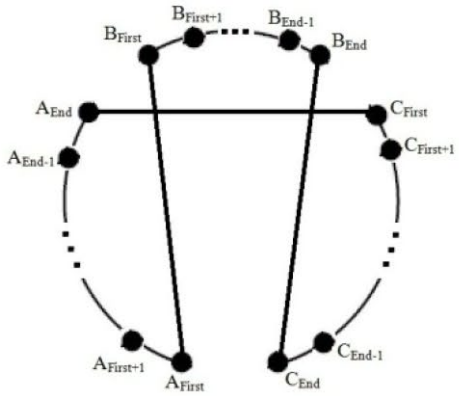
SOLUTION: 3-Opt



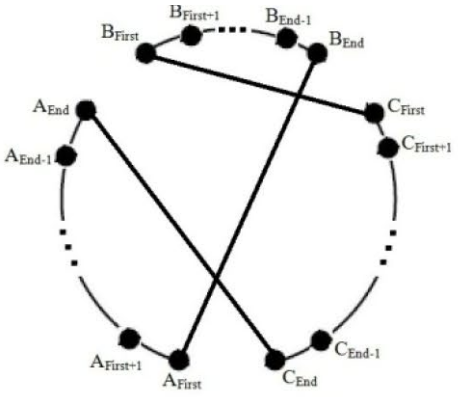
Original tour : ABC



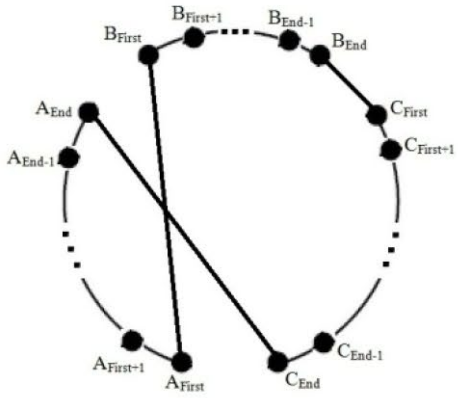
Method 1. ACB



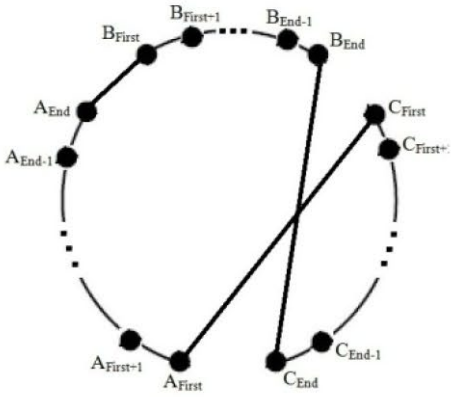
Method 2. ACB^{-1}



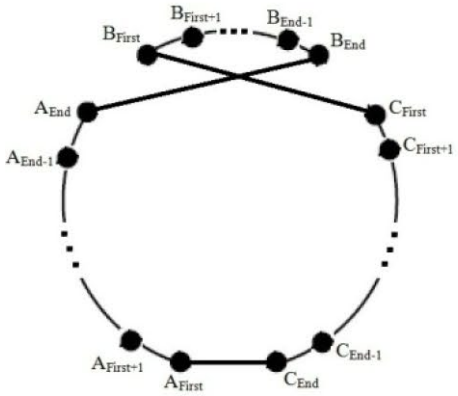
Method 3. $AC^{-1}B$



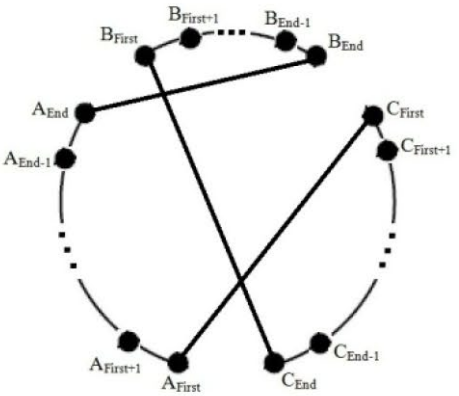
Method 4. $AC^{-1}B^{-1}$



Method 5. ABC^{-1}



Method 6. $AB^{-1}C$



Method 7. $AB^{-1}C^{-1}$

Local Search

If there are many moves that improve the current solution (i.e. the neighbourhood is large):

How do we decide **WHICH** move to take?

Selection with First Improving Move

Given a current solution s with cost c(s). Let M(s) be the neighbourhood of s, i.e. the set of all potential local improvements that can be applied to the solution s by a specific type of moves.

First Improving Move

Idea: Take the **first** move that improves the current solution

```
Repeat
    Choose a move m from M(s) and apply it to s.
    Let s' be the new solution
    if c(s') < c(s)
        s = s'
Until no move can improve the current solutioln s
```

Selection with Best Improving Move

Best Improving Move

Idea: Take the ***best*** move that improves the current solution

```
costOfBestNewSolution = ∝
for each move m in M(s)
    s' = new solution by applying m to s
    if c(s') < costOfBestNewSolution
       costOfBestNewSolution = c(s')
       bestMove = m
Apply bestMove to s
```

Notes:

- bestMove is the local optimum for moves in M(s)
- Running time can be large if M(s) contains many potential moves

Neighbourhood: Desired Properties

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

Connectivity: A global optimum can be reached from any feasible solution

Low diameter: Number of moves for linking any pair of solutions is small

Low ruggedness: Limited number of local optima; high correlation between utility function values of neighbour solutions

Limited size: Number of neighbour solutions is small (typically polynomial in problem size)

Easy to evaluate: A good neighbour solution must be found without prohibitive computation

→ Usually, a single neighbourhood does not combine all these properties!

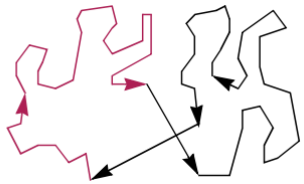
Example: 2-opt for TSP

Yes

Diameter: O(n)

No: major portion of a tour maybe inverted by a 2-opt move. Many local optima.

$O(n^2)$



Modification of a solution in O(n) (or less with good data structure)

Ruggedness of Neighbourhoods

Idea: Rugged neighbourhoods, i.e. neighbourhoods with high variability of utility function values between neighboring solutions are hard to search. Neighbourhood ruggedness is closely related to local minima density: Rugged neighbourhoods tend to have many local minima.

Characterization of ruggedness: The ruggedness of a neighbourhood N can be measured by means of the empirical autocorrelation function $\tau(i)$:

$$\tau(i) := \frac{m \cdot \sum_{k=1}^{m-i} (g_k - \bar{g}) \cdot (g_{k+i} - \bar{g})}{(m-i) \cdot \sum_{k=1}^{m} (g_k - \bar{g})^2} \qquad (i = 1, 2, \ldots, m-1)$$

where $g_1, \ldots, g_m$ are utility function values sampled from a random walk in N and with mean value $\bar{g}$.

High AC (close to 1 or -1): Smooth neighbourhood with low local optima density. Utility function values for neighboring candidate solutions are close on average. Problem is typically relatively easy for local search.
Low AC (close to zero): Rugged neighbourhood with high density of local optima in which local search typically gets trapped.

Limitation of Neighbourhood Size

Example: TSP with 2-opt move

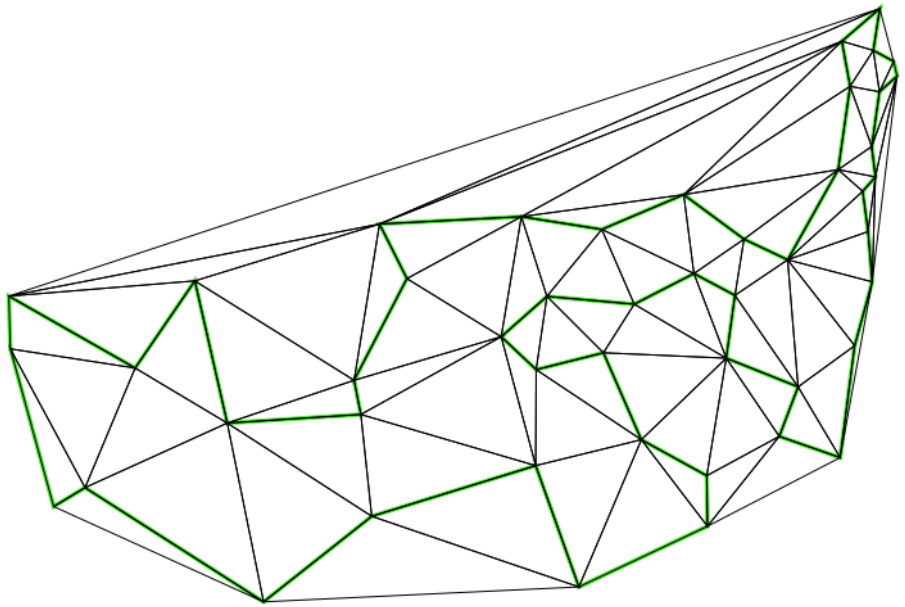
A size in $O(n^2)$ is prohibitive for large instances

Idea: Static limitation

The set M of moves is reduced to those that are assumed to be interesting

Example TSP:

Restrict tour to edges of Delauney-Triangulation



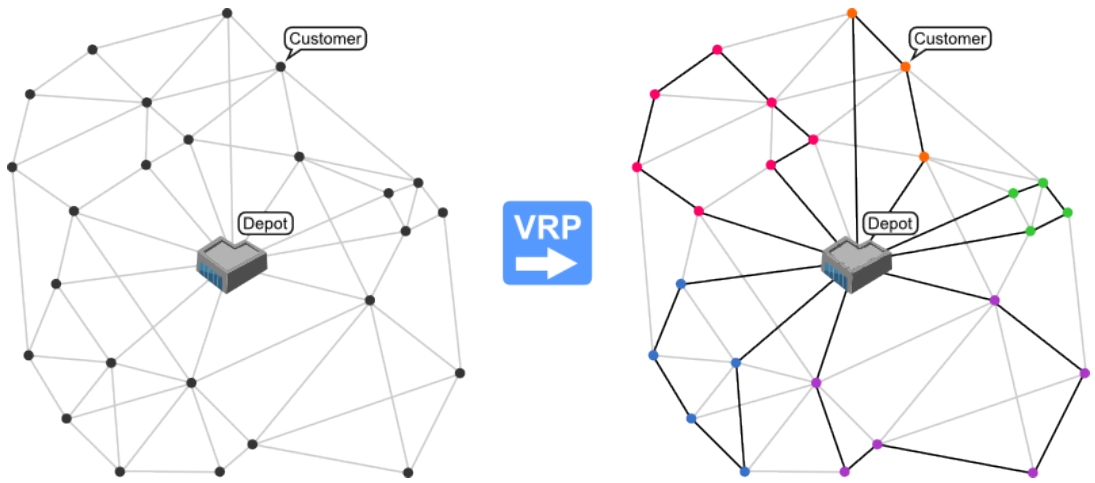
Capacitated Vehicle Routing Problem
CVRP

Definition: Capacitated Vehicle Routing Problem (CVRP)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

Input:

- n customers and 1 depot
- q_i : quantity ordered by customer i
- Distances d_{ij} between each pair of customers and distances between depot and each customer
- Q : vehicle capacity



Source: https://neo.lcc.uma.es/vrp/vehicle-routing-problem/

Feasible Solution:

A set of tours such that

- Each tour starts from and comes back to the depot
- Each customer appears exactly once in the set of tours
- Each customer receives her ordered quantity
- The sum of quantities on any tour ≤ Q

+ eventually other constraints on the tour length, time windows, etc.

Objective: Minimize the total travel length performed by the vehicle

LINK: https://neo.lcc.uma.es/vrp/

Exercise

Neighbourhoods for CVRP

a) Propose different neighbourhoods for the Capacitated Vehicle Routing Problem.

b) Give their size complexity as a function of the number *n* of customers and of the number *m* of tours.

c) Do these neighbourhoods have the connectivity property?

d) Give the runtime complexity of one move in one of the chosen neighbourhoods

SOLUTION: Neighborhood for CVRP

Neighborhood: select a customer c randomly, select destination tour randomly, select insertion position randomly (move c to new position if possible)

Analysis:

- *Size of Neighborhood*: O(n * number of tours * average number of customers per tour) = $O(n^2)$
- *Connectivity*: No. Since we move customers from one tour to another, the number of tours can never increase -> if initial solutions has "to few tours", one cannot achieve an optimal solution
- *Diameter*: Infinite, since not connected
- *Evaluation Time (for evaluating one move)*: O(1), since only constant edges in the original and destination tour are changed.

SOLUTION: Neighborhood for CVRP

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

General Idea: "Select a customer and move him to another tour«: Runtime-Complexities for one move

Select a customer X:

- Randomly: O(1)
- One that reduces cost of remaining solution most: O(n)
- One that is furthest away from point of gravity of his current tour: O(n)

Select destination tour T (only out of those where customer fits due to capacity constraints)

- Randomly: O(m)
- Tour with mininmum distance between X and a next customer on the same tour: O(n)
- Tour with minimum/maximum remaining capacity: O(m)
- Tour with minimum distance between X and point of gravity of the tour: O(m)

Select insertion point within tour T:

- Randomly: O(1)
- First/last position: O(1)
- Position that increases cost of tour the least: O(length of T)
- Solve TSP for tour T with the customer; O(time for TSP)

Randomized Local Searches

Motivation for Randomized Local Searches

Weakness of greedy methods

The best choice at a given step can be globally very bad

Observation

No available policy for selecting the globally best choice

It is always possible to find problem instances for which a greedy method degenerates (for NP-hard problems)

Tentative idea

Incorporate a random component in the selection of the next move within a given neighbourhood for altering the current solution

Idea of Randomized Local Searches

Zürcher Hochschule
für Angewandte Wissenschaften
School of
Engineering
IAMP Institut für Angewandte
Mathematik und Physik

Idea

- Avoid returning to the same local optimum again and again by introducing a random component in the choice of the moves
- Same template as for Local Search but with the possibility to "sometimes accept bad moves" in order to avoid being trapped in local optima
- Bias the choice of moves in favour of those leading to better solutions, i.e. moves which yield improvements or at most “small” deteriorations to the current solution

Randomized Approaches

- **Simulated Annealing:** Always accept improving moves. Furthermore, accept deteriorating moves with a probability corresponding to a “temperature”. Over time, temperature is lowered more and more
- **Threshold Accepting**: Accept all moves deteriorating the solution less than a given threshold
- **Great Deluge:** Accept deteriorating moves with a given minimal quality (i.e. lying above a “water level”). Water level is increased more and more (like a "flood")
- **Noising Methods:** Add a random noise when evaluating the moves

Simulated Annealing

Motivation from Metallurgy

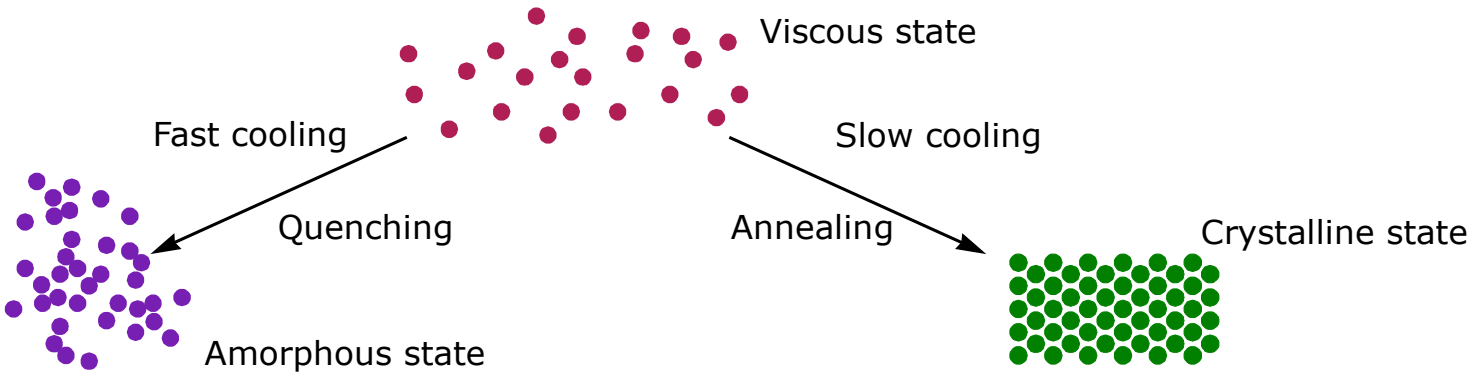
Analogy with the physical annealing process used in metallurgy

- By fast cooling of molten metal, an amorphous structure is obtained
- By cooling slowly, a crystalline structure is obtained

- Crystalline structures minimize the internal energy of the metal
- Amorphous structure corresponds to local minimum of energy:
  The particles are trapped in stable states
  They don't have enough energy to go to other stable states of lower energy

Idea

- Internal energy of a metal = utility function to minimize
- Particles are moving randomly = Accept degrading moves



Meta-Heuristic: Simulated Annealing

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

Input:

Initial solution s
Objective function to minimize f
Set M of moves that can be applied to any solution
Initial temperature T_0, final temperature T_f
Decreasing temperature factor $0 < \alpha < 1$
Best solution found $s^* = s$, current temperature $T = T_0$

$s \bullet m$ means: apply move m to solution s

Repeat
  Choose move m randomly, uniformly in M
  $D = f(s \bullet m) - f(s)$
  If $D < 0$ **Then**
    $s = s \bullet m$ // Improving moves are always accepted
  Else
    Generate random number $u \in]0, 1[$
    If $e^{-D/T} > u$ **Then**
      $s = s \bullet m$
  If $f(s) < f(s^*)$ **Then**
    $s^* = s$
  $T = \alpha T$ // or $T = T/(1 + \alpha T)$
While $T > T_f$
Return s^*

Practical Tipps

Value of initial temperature T_0:

Perform a given number of random moves, compute their average improvement value δ
Choose an initial acceptance rate $\mathbf{t}_0$ for degrading moves (e.g. $\mathbf{t}_0 = 30\%$)

Set $T_0 := -\delta/\ln(\mathbf{t}_0)$ (solution of the equation $\mathbf{t}_0 := e^{-\delta/T_0}$)

Do not diminish current temperature T at each iteration. Examples:

Set $T := \alpha T$ after:

- a fixed number of iterations (depending on problem size, e.g. : $100n$); or
- a fixed number of improvement steps (e.g. $12n$)

Exiting the main loop:

Do not choose a stopping criterion depending on T. Rather exit if e.g. s^* was not improved while temperature was decreased by 3 steps.

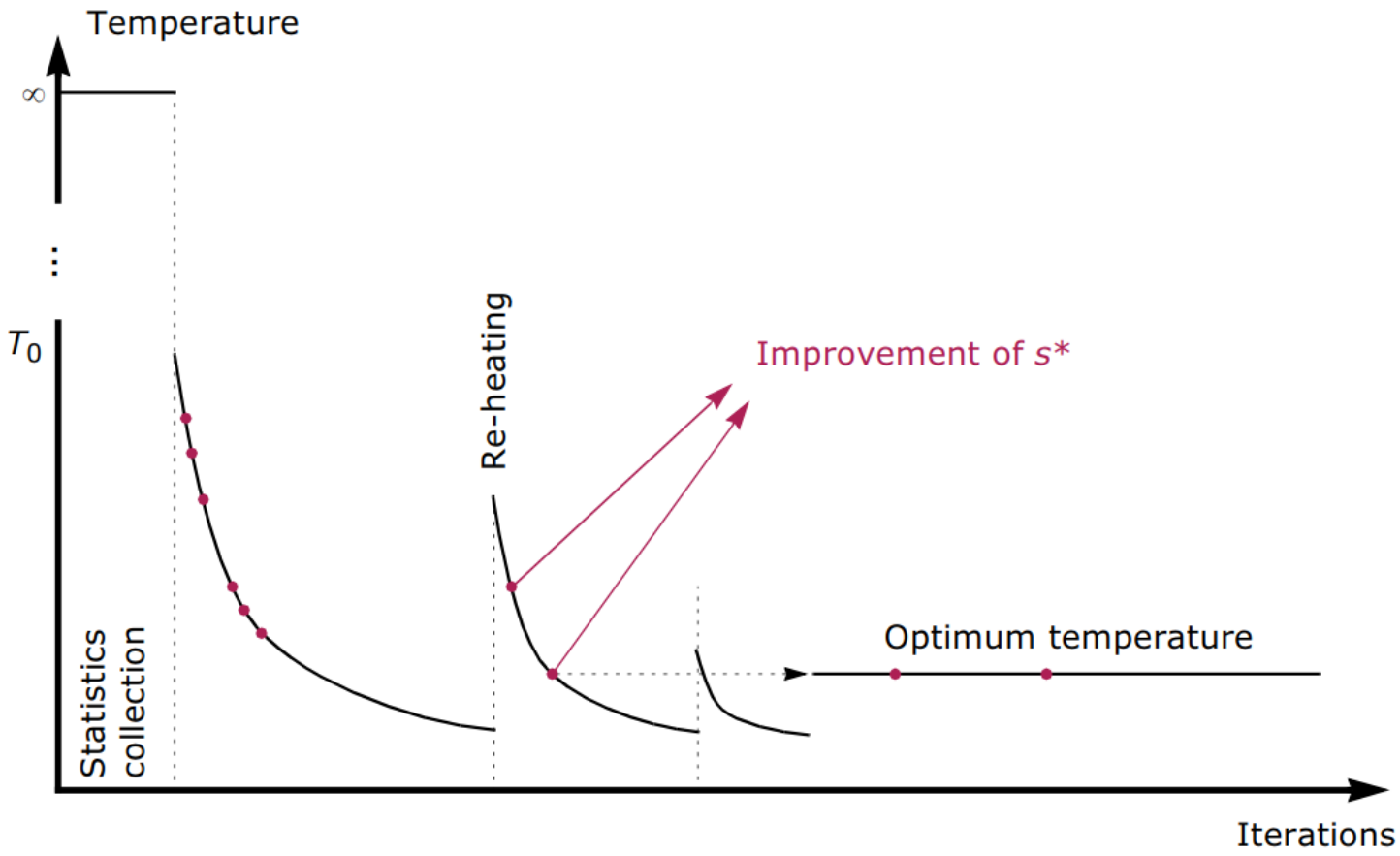
Temperature decreasing factor:

Value near to 1, e.g. $\alpha = 0.9$

Periodically re-heat the system

In general: Rather choose the sequence of temperatures freely

Example of Annealing Scheme



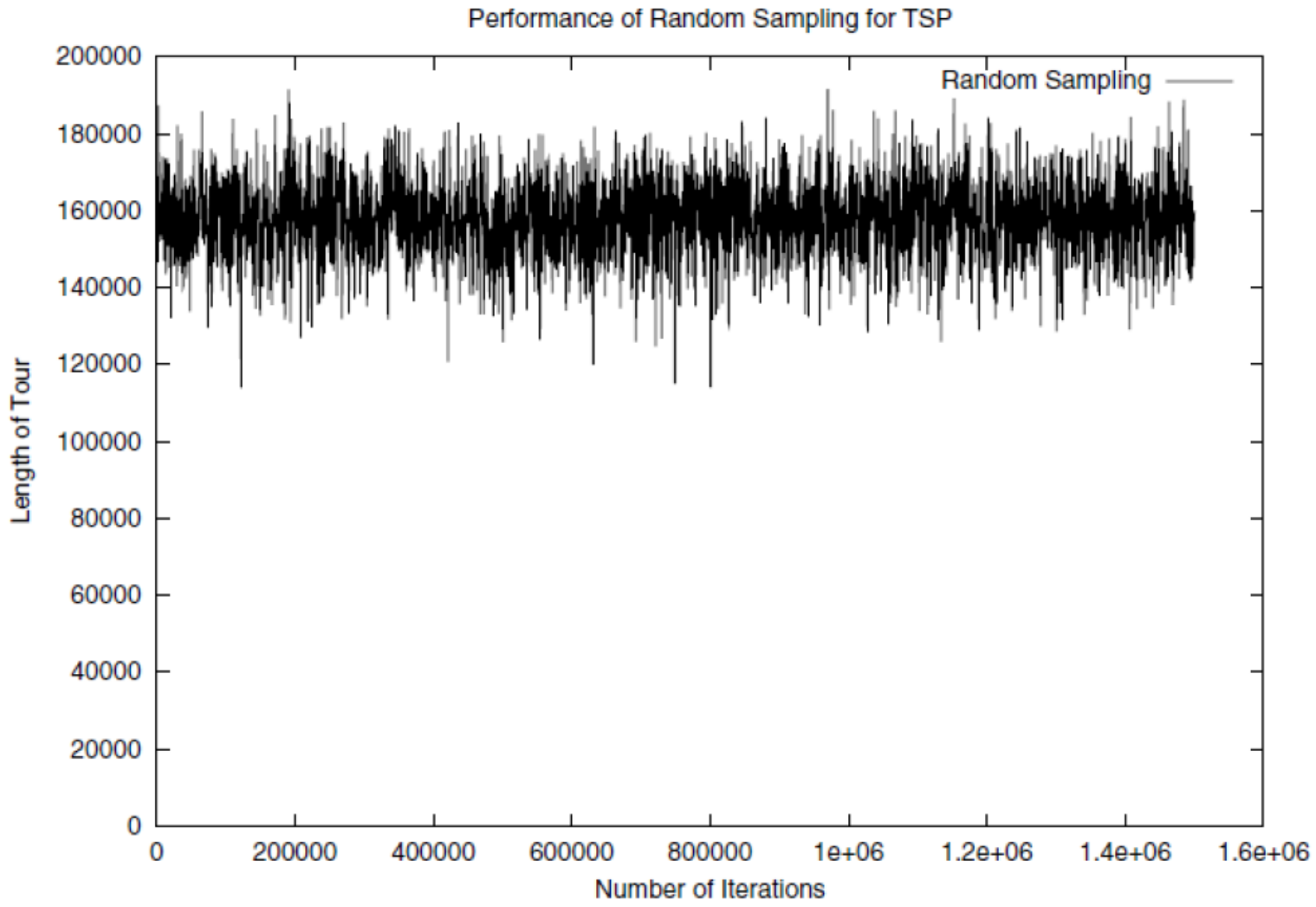
Important Note on Simulated Annealing

"Forget about this molten metal business. Simulated annealing is effective because it spends much more of its time working on good elements of the solution space than on bad ones, and because it avoids getting trapped repeatedly in the same local optima."

Source: Skiena, Algorithm Design Manual

Performance of Random Building

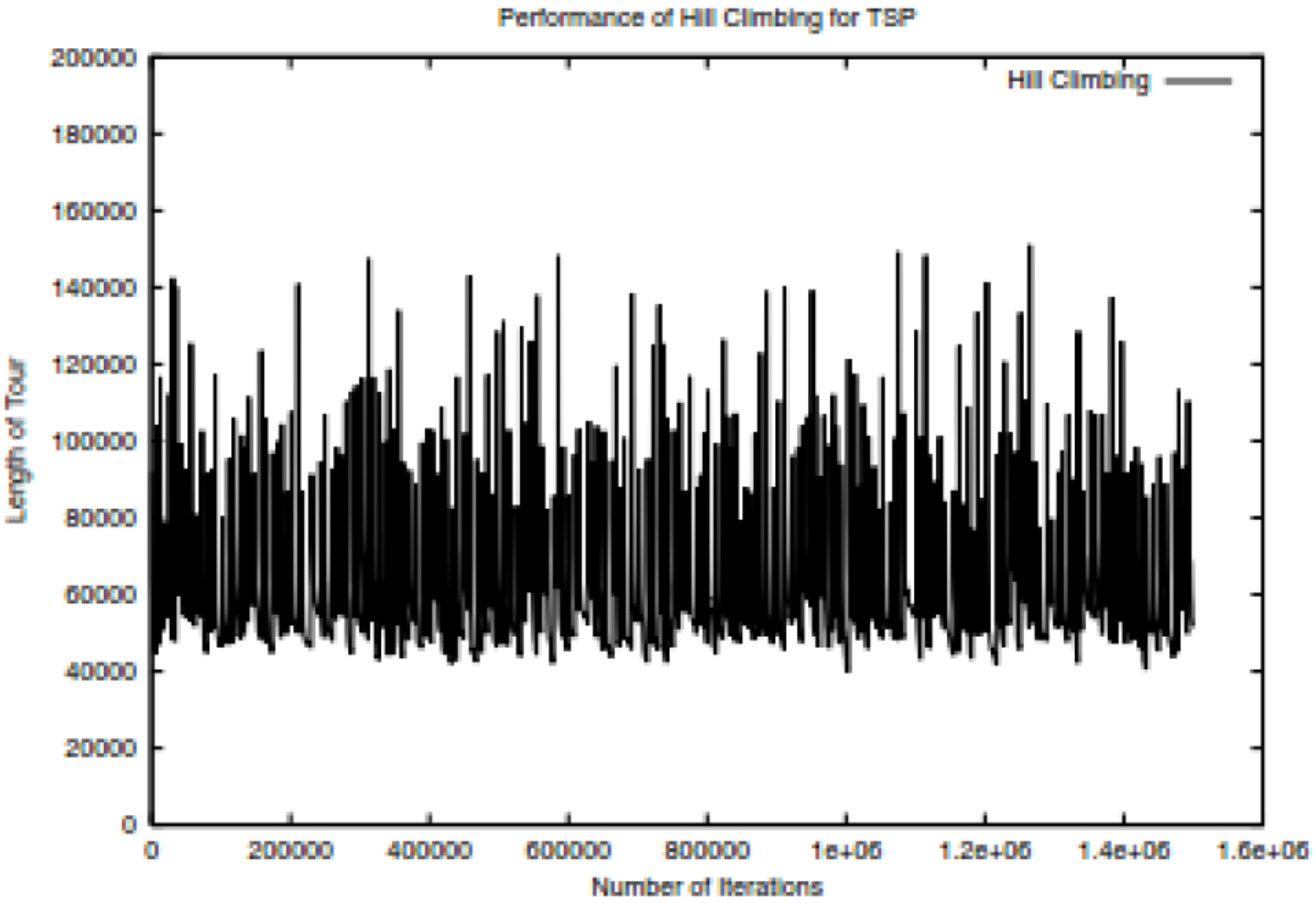
Zürcher Hochschule
für Angewandte Wissenschaften
School of
Engineering
IAMP Institut für Angewandte
Mathematik und Physik



Source: Skiena, Algorithm Design Manual

Performance of Local Search

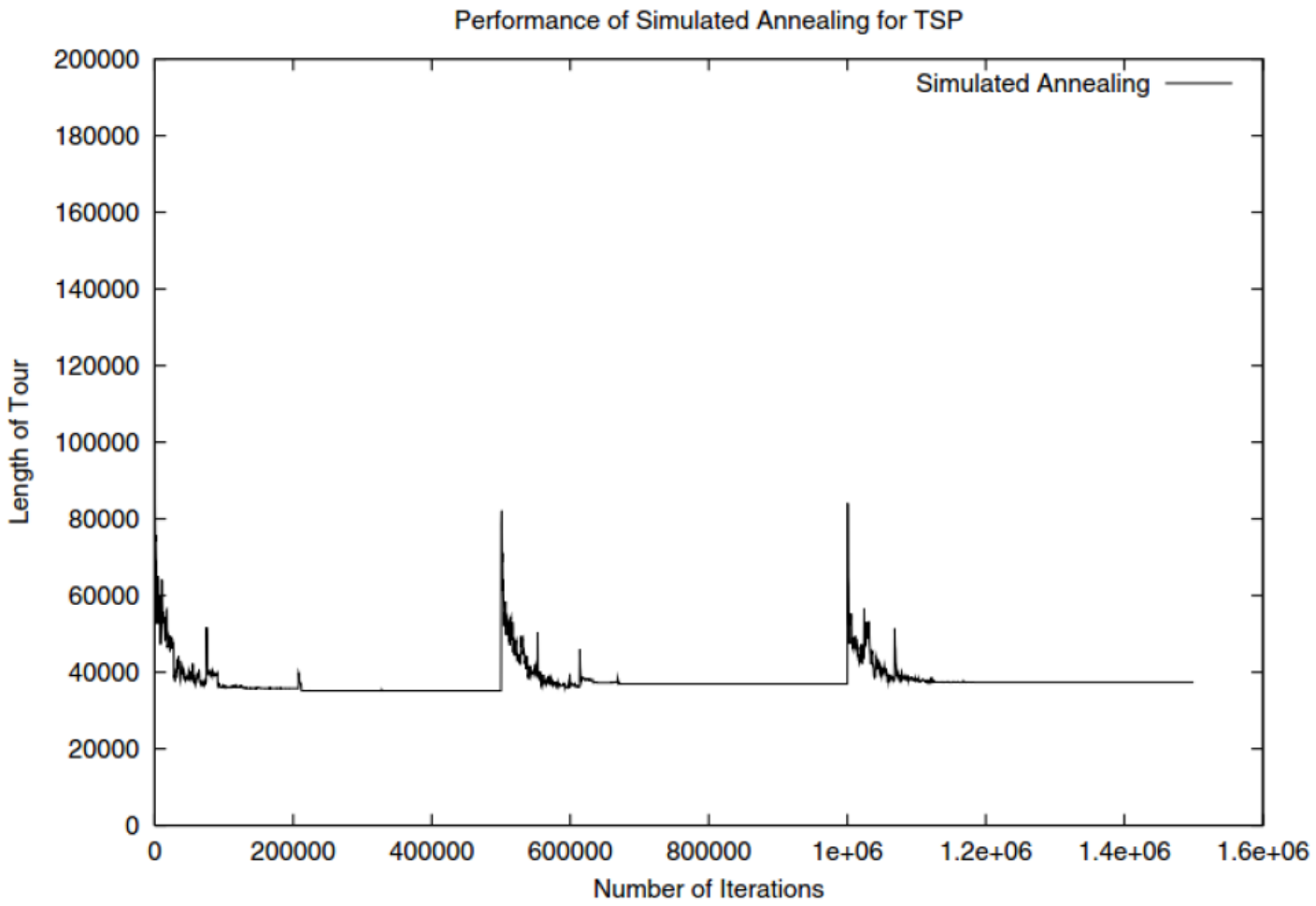
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik



Setting: start with random tour, improve tour by swapping 2 arbitrary cities until no imrpovements are possible any more. Then restart.

Source: Skiena, Algorithm Design Manual

Performance of Simulated Annealing (3 runs)



Source: Skiena: Algorithm Design Manual

Summary

Meta-Heuristics for Optimization Problem:

1. Use a **Constructive Method** to create an initial solution (e.g. Nearest Neighbour for TSP)

2. Use a **Local Improvement Method** to improve the solution (e.g. 2-opt for TSP)

3. Apply a **Selection Rule** to decide which improvement move is taken (e.g. Best Improving Move)

4. Use **Randomized Methods** (e.g. Simulated Annealing) to explore larger search spaces than purely greedy methods