Heuristic Optimization Methods

Lecture 4 – SA, TA, ILS

Agenda

- A bit more about SA
- Threshold Accepting
  - A deterministic variation of SA
- Generalized Hill-Climbing Algorithm
  - Generalization of SA
- Some additional Local Search based Metaheuristics
  - Iterated Neighborhood Search
  - Variable Neighborhood Search
  - Guided Local Search
- Leading to our next main metaheuristc: Tabu Search

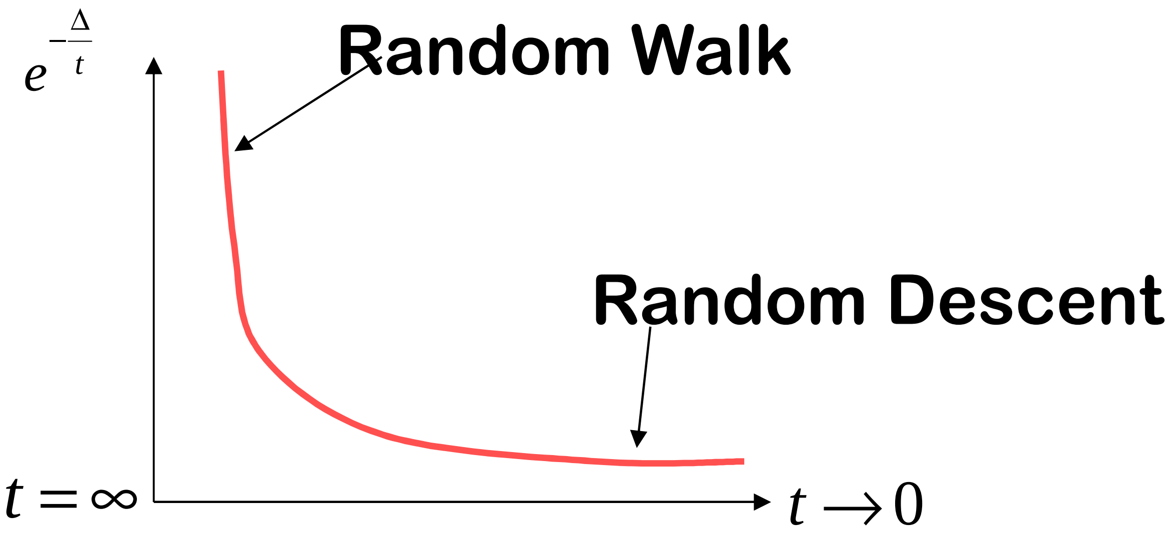
SA - Overview

- A modified random descent
  - Random exploration of neighborhood
  - All improving moves are accepted
  - Also accepts worsening moves (with a given probability)
- Control parameter: temperature
  - Start off with a high temperature (high probability of accepting worsening moves)
  - Cooling schedule (let the search space ”harden”)

Simulated Annealing

```
input: starting solution, s_0
input: neighborhood operator, N
input: evaluation function, f
input: the cooling schedule, t_k
input: the number of iterations for each temperature, M_k
current ⇐ s_0
k ⇐ 0
while stopping criterion not met do
    m ⇐ 0
    while m < M_k do
        s ⇐ randomly selected solution from N(current)
        if f(s) ≤ f(current) then
            current ⇐ s
        else
            Δ ⇐ f(s) − f(current)
            ξ ⇐ a random number, uniformly drawn from [0, 1]
            if ξ ≤ e^{−Δ/t_k} then
                current ⇐ s
            end if
        end if
        m ⇐ m + 1
    end while
    k ⇐ k + 1
end while
```

SA – Cooling Schedule



- Requires:
  - Good choice of cooling schedule
  - Good stopping criterion
  - Faster cooling at the beginning and end
  - Testing is important

SA – Choice of Move

- Standard: Random selection of moves in the neighborhood
  - Problematic around local optima
  - Remedy: Cyclic choice of neighbor
- Standard: Low acceptence rate at low temperatures
  - A lot of unneccesary calculations
  - Possible remedies
    - Acceptance probability
    - Choice of neighbor based on weighted selection
    - Deterministic acceptance

SA – Modifications and Extensions

- Probabilistic
  - Altered acceptance probabilities
  - Simplified cost functions
  - Approximation of exponential function
    - Can use a look-up table
  - Use few temperatures
  - Restart
- Deterministic
  - Threshold Accepting, TA
  - Cooling schedule
  - Restart

SA – Combination with Other Methods

- Preprocessing – find a good starting solution
- Standard local search during the SA
  - Every accepted move
  - Every improving move
- SA in construction heuristics

Threshold Accepting

- Extensions/generalizations
  - Deterministic annealing
  - Threshold acceptance methods
  - Why do we need randomization?
- Local search methods in which deterioration of the objective up to a ***threshold*** is accepted
  - Accept if and only if $\Delta \leq \Theta_k$
- Does not have proof of convergence, but in practice results have been good compared to SA

Threshold Accepting

input: starting solution, s_0
input: neighborhood operator, N
input: evaluation function, f
input: threshold, Θ
$current \Leftarrow s_0$
while stopping criterion not met **do**
    $s \Leftarrow$ randomly selected solution from $N(current)$
    $\Delta \Leftarrow f(s) - f(current)$
    if $\Delta < \Theta$ **then**
        $current \Leftarrow s$
    end if
end while

Generalized Hill-Climbing Algorithms

- Generalization of SA
- General framework for modeling Local Search Algorithms
  - Can describe Simulated Annealing, Threshold Accepting, and some simple forms of Tabu Search
  - Can also describe simple Local Search variations, such as the ”First Improvement”, ”Best Improvement”, ”Random Walk” and ”Random Descent”-strategies

Generalized Hill-Climbing Algorithm

input: starting solution, s_0
input: neighborhood operator, N
input: evaluation function, f
input: outer loop bound, K, inner loop bounds M_k, $k = 1, 2, \ldots, K$
input: hill-climbing (random) functions $R_k : S \times S \to \mathbb{R} \cup \{-\infty, +\infty\}$
$current \Leftarrow s_0$
$k \Leftarrow 1$
$m \Leftarrow 1$
while $k \leq K$ **do**
    while $m \leq M_k$ **do**
        $s \Leftarrow$ solution generated from $N(current)$
        $\Delta \Leftarrow f(s) - f(current)$
        if $R_k(current, s) \geq \Delta$ **then**
            $current \Leftarrow s$
        end if
        $m \Leftarrow m + 1$
    end while
    $k \Leftarrow k + 1$
end while

Generalized Hill-Climbing Algorithms (2)

- The flexibility comes from
  - Different ways of generating the neighbors
    - Randomly
    - Deterministically
    - Sequentially, sorted by objective function value?
  - Different acceptance criteria, R_k
    - Based on a threshold (e.g., Threshold Accepting)
    - Based on a temperature and difference in evaluation (e.g., SA)
    - Other choices?

Some Other LS-based Metaheuristics

- Our first main metaheuristic:
  - Simulated Annealing
- Our second main metaheuristic:
  - Tabu Search
- But first, some other LS-based methods:
  - Threshold Accepting (variation of SA)
  - Generalized Hill-Climbing Algorithm (generalization of SA)
  - Iterated Local Search (better than random restarts)
  - Variable Neighborhood Search (using a set of neighborhoods)
  - Guided Local Search (closer to the idea of Tabu Search)

Restarts (1)

- Given a Local Search procedure (either a standard LS or a metaheuristic such as SA)
  - After a while the algorithm stops
    - A Local Search stops in a local optimum
    - SA stops when the temperature has reached some lowest possible value (according to a cooling schedule)
  - What to do then?
- Restarts
  - Repeat (iterate) the same procedure over and over again, possibly with different starting solutions

Restarts (2)

- If everything in the search is deterministic (no randomization), it does no good to restart
- If something can be changed…
  - The starting solution
  - The random neighbor selection
  - Some controlling parameter (e.g., the temperature)
- … then maybe restarting can lead us to a different (and thus possibly better) solution

Iterated Local Search (1)

- We can look at a Local Search (using ”Best Improvement”-strategy) as a function
  - Input: a solution
  - Output: a solution
  - LS: S → S
  - The set of local optima (with respect to the neighborhood used) equals the range of the function
- Applying the function to a solution returns a locally optimal solution (possibly the same as the input)

Iterated Local Search (2)

- A simple algorithm (Multi-start Local Search):
  - Pick a random starting solution
  - Perform Local Search
  - Repeat (record the best local optimum encountered)
- Generates multiple independent local optima
- Theoretical guarantee: will encounter the global optimum at some point (due to random starting solution)
- Not very efficient: wasted iterations

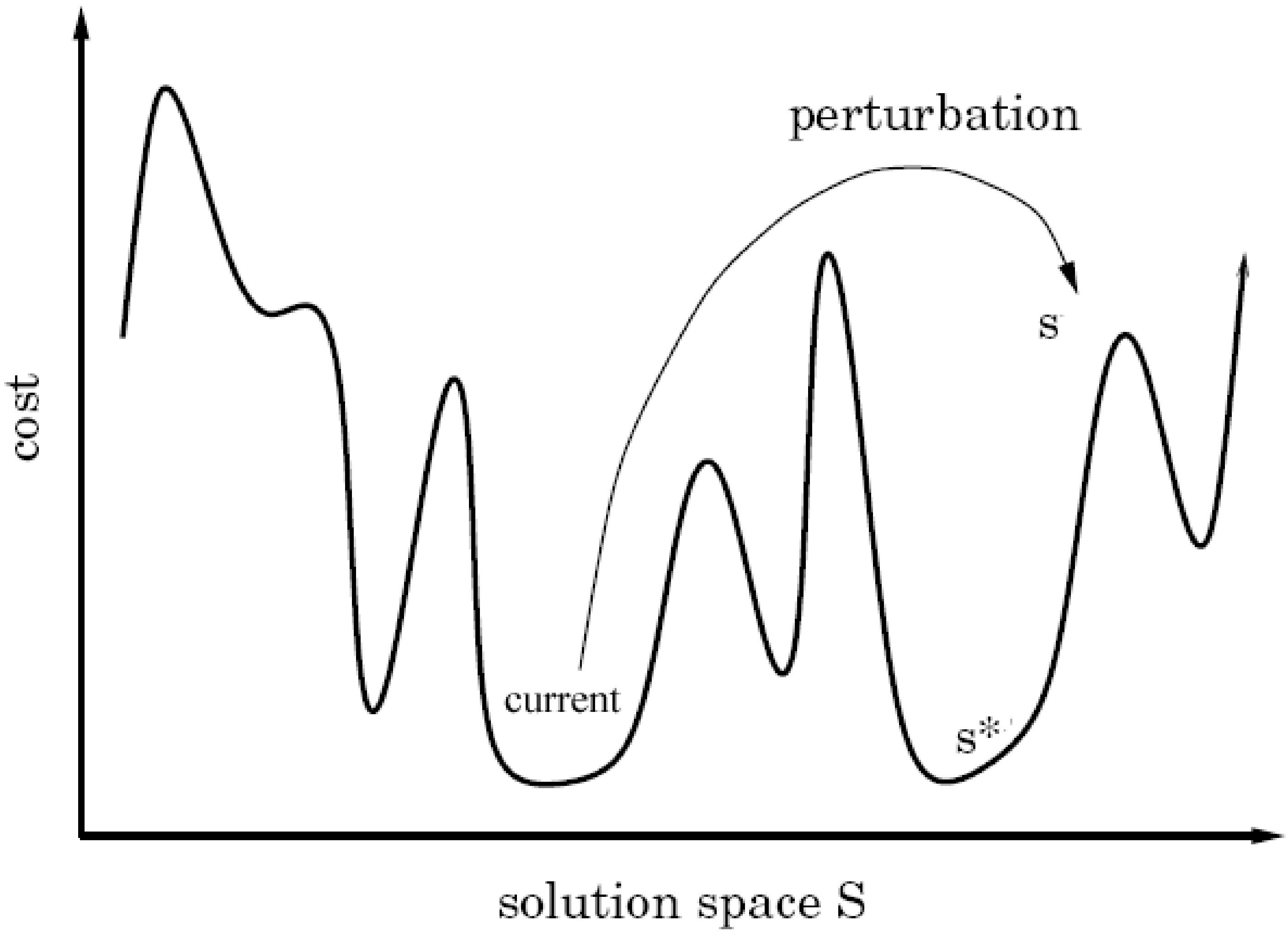
Iterated Local Search (3)

- Iterated Local Search tries to benefit by restarting close to a currently selected local optimum
  - Possibly quicker convergence to the next local optimum (already quite close to a good solution)
  - Has potential to avoid unnecessary iterations in the Local Search loop, or even unnecessary complete restarts
    - Uses information from current solution when starting another Local Search

Iterated Local Search

```
input: starting solution, s_0
input: Local Search procedure, LS
current ⇐ LS(s_0)
while stopping criterion not met do
    s ⇐ perturbation of current based on search history
    s* ⇐ LS(s)
    if s* is accepted as the new current solution then
        current ⇐ s*
    end if
end while
```

Pictorial Illustration of ILS



Principle of Iterated Local Search

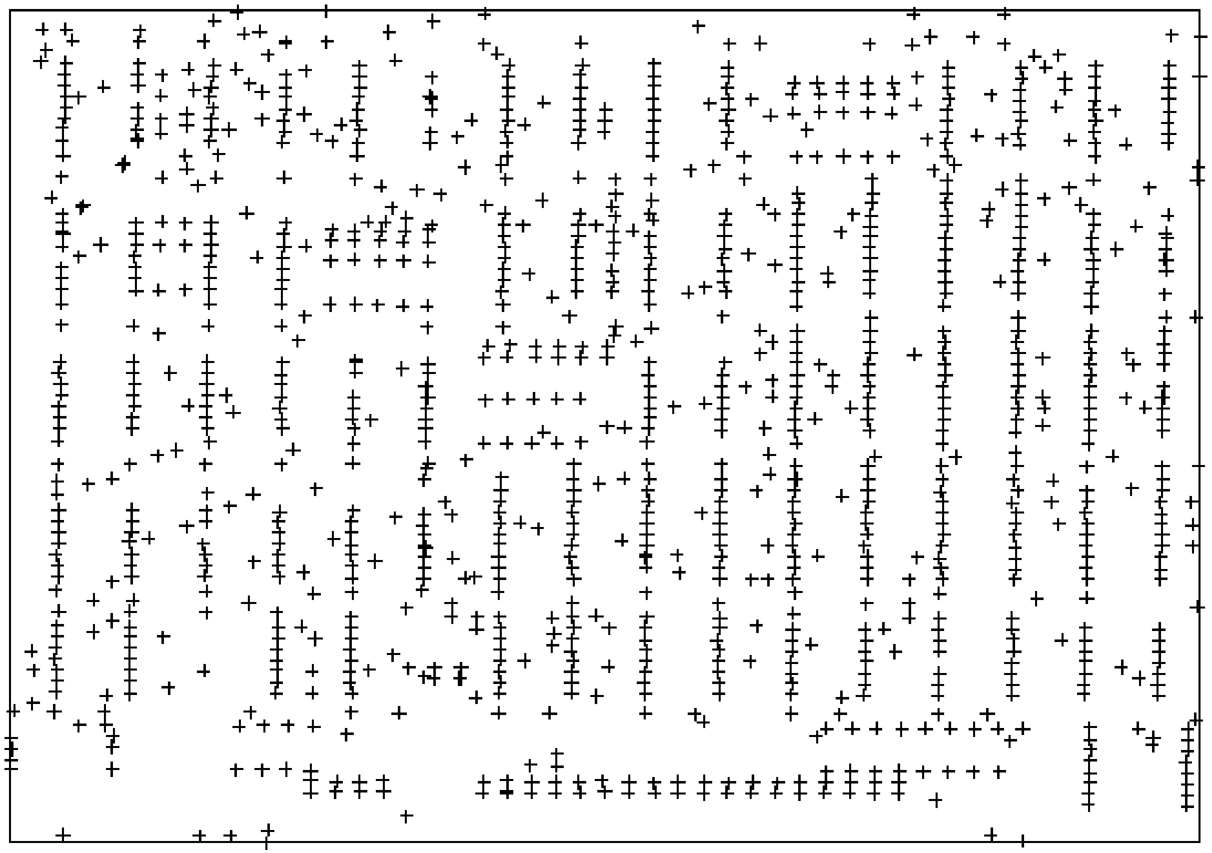
- The Local Search algorithm defines a set of locally optimal solutions
- The Iterated Local Search metaheuristic searches among these solutions, rather than in the complete solution space
  - The search space of the ILS is the set of local optima
  - The search space of the LS is the solution space (or a suitable subspace thereof)

A Basic Iterated Local Search

- Initial solution:
  - Random solution
  - Construction heuristic
- Local Search:
  - Usually readily available (given some problem, someone has already designed a local search, or it is not too difficult to do so)
- Perturbation:
  - A random move in a ”higher order neighborhood”
  - If returning to the same solution (*s*=current*), then increase the strength of the perturbation?
- Acceptance:
  - Move only to a better local optimum

ILS Example: TSP (1)

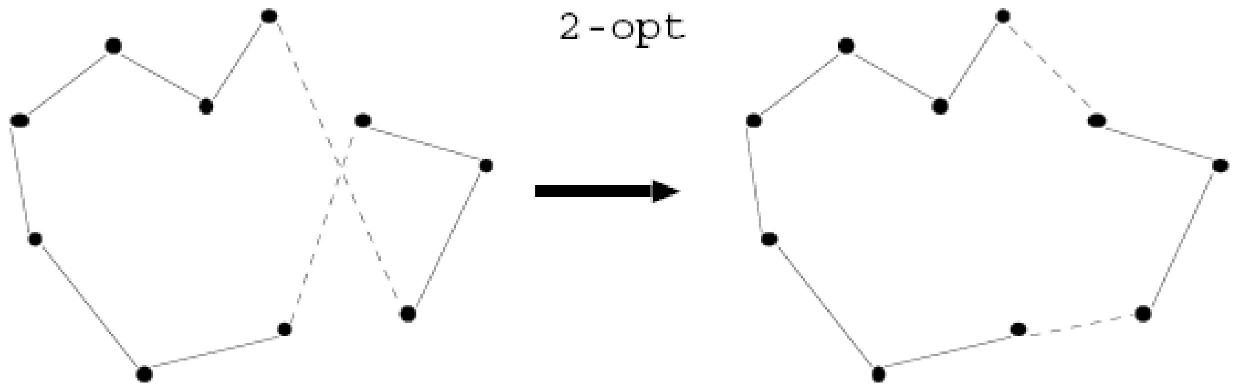
- Given:
  - Fully connected, weighted graph
- Find:
  - Shorted cycle through all nodes
- Difficulty:
  - NP-hard
- Interest:
  - Standard benchmark problem



(Example stolen from slides by Thomas Stützle)

ILS Example: TSP (2)

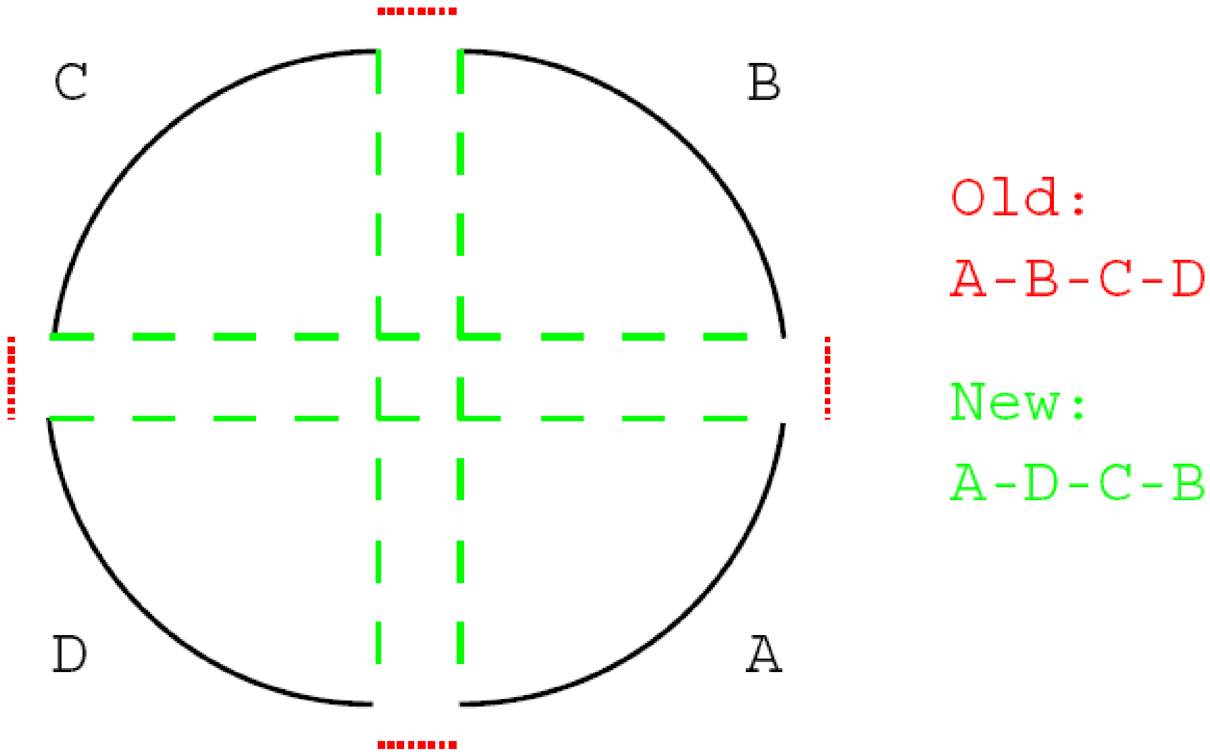
- Initial solution: greedy heuristic
- Local Search: 2-opt



- Perturbation: double-bridge move (a specific 4-opt move)
- Acceptance criterion: accept s* if f(s*) ≤ f(current)

ILS Example: TSP (3)

- Double-bridge move for TSP:



About Perturbations

- The strength of the perturbation is important
  - Too strong: close to random restart
  - Too weak: Local Search may undo perturbation
- The strength of the perturbation may vary at run-time
- The perturbation should be complementary to the Local Search
  - E.g., 2-opt and Double-bridge moves for TSP

About the Acceptance Criterion

- Many variations:
  - Accept s* only if f(s*)<f(current)
    - Extreme intensification
    - Random Descent in space of local optima
  - Accept s* always
    - Extreme diversification
    - Random Walk in space of local optima
  - Intermediate choices possible
- For TSP: high quality solutions known to cluster
  - A good strategy would incorporate intensification

ILS Example: TSP (4)

- $\Delta_{avg}(x)$ = average deviation from optimum for method x
- RR: random restart
- RW: ILS with random walk as acceptance criterion
- Better: ILS with First Improvement as acceptance criterion

instance	Δ_{avg}(RR)	Δ_{avg}(RW)	Δ_{avg}(Better)
kroA100	0.0	0.0	0.0
d198	0.003	0.0	0.0
lin318	0.66	0.30	0.12
pcb442	0.83	0.42	0.11
rat783	2.46	1.37	0.12
pr1002	2.72	1.55	0.14
d1291	2.21	0.59	0.28
fl1577	10.3	1.20	0.33
pr2392	4.38	2.29	0.54
pcb3038	4.21	2.62	0.47
fl3795	38.8	1.87	0.58
rl5915	6.90	2.13	0.66

ILS: The Local Search

- The Local Search used in the Iterated Local Search metaheuristic can be handled as a ”Black Box”
  - If we have any improvement method, we can use this as our Local Search and focus on the other parts of the ILS
  - Often though: a good Local Search gives a good ILS
- Can use very complex improvement methods, even such as other metaheuristics (e.g., SA)

Guidelines for ILS

- The starting solution should to a large extent be irrelevant for longer runs
- The Local Search should be as effective and fast as possible
- The best choice of perturbation may depend strongly on the Local Search
- The best choice of acceptance criterion depends strongly on the perturbation and Local Search
- Particularly important: the interaction among perturbation strength and the acceptance criterion

A Comment About ILS and Metaheuristics

- After seeing Iterated Local Search, it is perhaps easier to understand what a metaheuristic is
- ILS required that we have a Local Search algorithm to begin with
  - When a local optimum is reached, we perturb the solution in order to escape from the local optimum
  - We control the perturbation to get good behaviour: finding an improved local optimum
- ILS ”controls” the Local Search, working as a ”meta”-heuristic (the Local Search is the underlying heuristic)
  - Meta- in the meaning ”more comprehensive”; ”transcending”

Summary

- Simulated Annealing
  - Overview and repetition
- Threshold Accepting
  - Deterministic variation of SA
- Generalized Hill-Climbing Algorithm
  - Generalization of SA
- Iterated Local Search
  - Searches in the space of local optima