Tabu Search meta-heuristic: Motivations and basic ideas

- Local search ends up in a local optimum: we need to *escape* from local optima in order to search the solution space more extensively and effectively
  - Any iterative search/exploration process should accept also *non-improving moves* to be able to escape from a local optimum
  - As soon as non-improving moves are possible, there is the risk of visiting again a solution, falling back to a previous local optimum, or, more generally, *cycling* $\rightarrow$ waste of resources

- *Tabu search* (TS) (Glover, 1989) is an *iterative, memory-based neighborhood-search method*
  - The optimization process proceeds by generating a sequence of solutions by applying some *local neighborhood search*
  - TS keeps information on the *itinerary* through the visited solutions
  - Such information is used to guide the move from a solution point s to the next by *restricting* the possible local choices to some subset of $\mathcal{N}(s)$ or by *modifying* the neighborhood itself (by locally modifying the objective function)

- In TS, the structure of the neighborhood $\mathcal{N}(s)$ of a solution s is variable from iteration to iteration $\rightarrow$ *dynamic neighborhood search technique*

Tabu search: a very high-level pseudo-code

procedure Tabu_Search$()$

Define a neighborhood function $\mathcal{N}$ and select a local search algorithm *localSearch()* for $\mathcal{N}$;

$t := 0$;

$s_t :=$ Generate a starting solution; // e.g., with a construction heuristic

$s^{best} := s_t$;

$\mathcal{H}_t := (s_t,\ s^{best}, t)$; // initialize the memory data structure

while $(\neg$ termination_criteria$(s_t, \mathcal{H}_t))$

$t := t + 1$;

Depending on $(s_t, \mathcal{H}_t)$, generate a subset $N_t^{\mathcal{H}}(s_t) \subseteq \mathcal{N}(s_t)$;

$s_t :=$ Best $\{s \mid s \in N_t^{\mathcal{H}}(s_t)\}$ with respect to f or some $f_t^{\mathcal{H}} = F(f, s_t, \mathcal{H}_t)$;

if $(f(s_t) < f(s^{best}))$

$s^{best} := s_t$;

Update the memory data structure $\mathcal{H}_t$;

end while

return s^{best};

- Standard steepest ascent/descent is a special case of this scheme
- How do we restrict the current neighborhood $\mathcal{N}(s_t)$ to define $N_t^{\mathcal{H}}(s_t)$?
  - Feasible configurations $s \in \mathcal{N}(s_t)$ are declared *tabu* on the whole or if they contain properties (*attributes*) declared as *forbidden* at the current time step t given $\mathcal{H}_t$

The *Tabu* list structure

- All forbidden configurations or configuration properties are stored step by step in a data structure called *tabu list* (the $\mathcal{H}_t$ data structure holding the *memory* of the search)
- *Explicit memory:* One obvious choice would consist in keeping memory of *all visited configurations*
  - The size of the tabu list would grow linearly with time
  - Selecting every time the best non tabu configuration in the current neighborhood would mean to check every feasible configuration in the neighborhood $\mathcal{N}(s_t)$ vs. the configurations in the tabu list $\rightarrow$ *very expensive* (*hash maps* are an effective way to implement store and retrieval)
- *Attributive memory:* The *move* from solution configuration s_t to configuration s_{t+1} can be expressed in terms of one or more *attributes* (i.e., features of a move)
  - Storing attributes rather than whole solutions it allows to keep in the tabu list a more *compact representation of the the search history*
  - This results in *reduced memory requirements* and *faster searching in the list*
  - Using attributes instead of whole solution means loss of information: *cycles can be generated*

Visited configurations (2-swap moves)	**Tabu list with explicit memory at** $t = 5$	**Tabu list with attributes at** $t = 5$
$s_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$	$\{ \quad (1, 2, 3, 4, 5, 6, 7, 8, 9, 10),$	$\{$
$s_2 = (1, 3, 2, 4, 5, 6, 7, 8, 9, 10)$	$(1, 3, 2, 4, 5, 6, 7, 8, 9, 10),$	$(3, 2), (2, 3),$
$s_3 = (1, 3, 2, 5, 4, 6, 7, 8, 9, 10)$	$(1, 3, 2, 5, 4, 6, 7, 8, 9, 10),$	$(5, 4), (4, 5),$
$s_4 = (6, 3, 2, 5, 4, 1, 7, 8, 9, 10)$	$(6, 3, 2, 5, 4, 1, 7, 8, 9, 10),$	$(6, 1), (1, 6),$
$s_5 = (6, 7, 2, 5, 4, 1, 3, 8, 9, 10)$	$(6, 7, 2, 5, 4, 1, 3, 8, 9, 10) \quad \}$	$(7, 3), (3, 7) \quad \}$

Examples of move attributes

- A move that would take from current solution s_i to the potential next solution configuration s_j can be characterized in terms of a number of different attributes, such as:
  - Change of a variable x_k
  - Change of a variable x_k from 0 to 1 (e.g., inclusion/removal of an edge in a TSP tour)
  - Change of a variable x_k from value v_1 to value v_2
  - Change of a variable x_k by an amount v
  - Move from solution s_i to solution s_j
  - Change from an objective value $v_i = f(s_i)$ to $v_j = f(s_j)$
  - Objective variation is equal to $v = f(s_i) - f(s_j)$
  - Change of a function $g(s_i)$ to $g(s_j)$ (g can be any function strategically defined or a function derived from problem formulation, e.g., the distance (dissimilarity) between s_i and s_j, or between s_i and the last visited local optimum)
  - Combination of attributes (e.g., both x_k and x_j change from 1 to 0: 2-swap or 2-opt move)
  - ...
- Once the move is executed, the related attributes that have been selected to guide the TS algorithm, are marked as tabu and added to the tabu list

Tabu status

- An attribute is associated to a *tabu status*
  - *Tabu-active*
  - *Tabu-inactive*
- The number of restrictions stored in the tabu list T can become so large that the TS algorithm is unable to get to further good configurations, the search might even freeze at some configuration if no further move is allowed $\rightarrow$ *tabu status need to change over time,* $|T|$ *cannot grow indefinitely*
  - *Recency-based restrictions* (*short-term memory*)
  - *Frequency-based restrictions* (*short-term memory*)
  - *Long-term strategies*
  - *Aspiration criteria*

Recency-based restrictions and tabu tenure

- *Tabu tenure:* items marked as tabu loose their tabu stamp after some time; they become tabu-inactive and can thus be again part of new configurations
- The tenure can be set *per attribute* or *globally* for all attributes (T = circular array, $|T| < K$)
- The tenure is a critical parameter: *short tenures* allows the exploration of solutions close to a local optimum (*intensification*), while *long tenures* can help to drift away from it (*diversification*)
  - *Static*, usually depending on problem size (e.g., 7, $\sqrt{n}$, $n/10$ iterations)
  - *Stochastic* (e.g., $5 + U(0,1) \cdot 4$, $\sqrt{n} + U(0,1) \cdot \sqrt{n}$)
  - *Dynamic:*
    - *Random:* the tenure τ is uniformly sampled from a range $[\tau_{min}, \tau_{max}]$ and is either maintained constant for a certain number of iterations (e.g., $\alpha\tau_{max}$) before selecting a new one, or a new τ is sampled for every new tabu attribute included in T
    - *Systematic:* a sequence of tenure values (e.g., alternating increasing and decreasing values) is defined and repeatedly used during the search
    - *Time dependent:* the tenure is progressively decreased according to the time (or the number of iterations) in order to progressively reduce diversification level
  - *Adaptive:* the tenure is increased when a solution is re-visited, decreased if no repetitions occur for a sufficient long time (*Reactive TS*, Battiti and Tecchiolli, 1994); instead of storing all solutions, one reference solution can be selected at a time

Frequency-based restrictions, long-term strategies

- *Frequency-based restrictions:*
  - Attributes are set tabu-active if they occur with a certain frequency (in the last iterations)
  - Frequencies can be:
    - Absolute
    - Relative to the occurrences of the most common attribute
    - Relative to the average occurrence of attributes

- *Long-term strategies:*
  - Permanent storing of complete configurations in the tabu list
  - Information about choices made in the past that have been proved to lead to good or bad results (*learning* from experience)
  - Storing a set of *elitist solutions*

Aspiration criteria

- Aspiration criteria are used to *locally override tabu restrictions*
- Aspiration criteria can be very important for the success of a TS implementation
- *Aspiration by* Default:
  - Applies if all moves are classified tabu
  - The *least tabu* move is selected
- *Aspiration by* Objective:
  - *Global form:* Applies if the solution improves over the best so far solution
  - *Regional form:* Search space is partitioned into regions; applies if the solution improves over the best so far solution in the same region
  - *Recency form:* Applies if the solution improves the best solution found in the last k iterations
- *Other Aspiration criteria:*
  - Aspiration by Similarity with elite configurations
  - Aspiration by Search Direction
  - Aspiration by Influence

Use of memory for *intensification* and *diversification*

- Size and properties of the neighborhood are locally modified in order to perform intensification and/or diversification steps

- *Intensification:* usually applied when no configurations with a quality comparable to that of stored elite configuration(s), have been found in the last iterations
  - Choice rules for neighborhood moves are locally modified in order to encourage move combinations and solution properties historically found to be good
  - Jump to or initiate a return to regions in the configuration space in which some stored elite solutions lie: these regions can then be searched more thoroughly

- *Diversification:* encourages the system to examine unvisited regions of the configuration space and thus to visit configurations that might differ strongly from all configurations touched before
  - Random perturbation after stagnation or long-term cycling
  - *Coupling intensification and diversification:* instead of jumping to one of the stored elite configurations, the system jumps to a configuration that has been created by changing one of the elite configurations in some significant way, i. e., slightly enough to search the neighborhood of the elite configuration and strongly enough so that the new configuration contains properties that are not part of the elite configuration from which it was constructed

Very simple TSP example

- Artificial TSP instance with 7 cities
- A solution is represented as a permutation
- The 2-Swap neighborhood is used
- Tabu tenure is set to 3 iterations
- Best solution aspiration criterion is used

- We look for solutions of high quality
- Since a swap move `swap(i,j)` is symmetric with respect to i and j, when `swap(i,j)` is included in the tabu list it means that both `swap(i,j)` and `swap(j,i)` are set tabu-active

Simple example of TS for TSP: Step 1

Solution: 2 5 7 3 4 6 1 (quality 10)

- *Tabu list:*
  - empty
- *Best 5 candidate moves in the 2-swap neighborhood:*
  - swap (5,4) (+6)
  - swap (7,4) (+4)
  - swap (3,6) (+2)
  - swap (2,3) (0)
  - swap (4,1) (-1)
- *Selected swap:* swap(5,4)

Simple example of TS for TSP: Step 2

Solution: 2 4 7 3 5 6 1 (quality 16)

- *Tabu list:*
  - swap (5,4) (3 iterations)
- *Best 5 candidate moves in the 2-swap neighborhood:*
  - swap (3,1) (+2)
  - swap (2,3) (+1)
  - swap (3,6) (-1)
  - swap (7,1) (-2)
  - swap (6,1) (-4)
- *Selected swap:* swap(3,1)

Simple example of TS for TSP: Step 3

Solution: 2 4 7 1 5 6 3 (quality 18)

- *Tabu list:*
  - swap (3,1) (3 iterations)
  - swap (5,4) (2 iterations)
- *Best 5 candidate moves in the 2-swap neighborhood:*
  - swap (1,3) (-2, tabu)
  - swap (2,4) (-4)
  - swap (7,6) (-6)
  - swap (4,5) (-7, tabu)
  - swap (5,3) (-9)
- *Selected swap:* swap(2,4)

Simple example of TS for TSP: Step 4

Solution: 4 2 7 1 5 6 3 (quality 14)

- *Tabu list:*
  - swap (2,4) (3 iterations)
  - swap (3,1) (2 iterations)
  - swap (5,4) (1 iterations)
- *Best 5 candidate moves in the 2-swap neighborhood:*
  - swap (4,5) (+6, tabu, aspiration)
  - swap (5,3) (+2)
  - swap (7,1) (0)
  - swap (1,3) (-3, tabu)
  - swap (2,6) (-6)
- *Selected swap:* swap(4,5)

Simple example of TS for TSP: Step 5

Solution: 5 2 7 1 4 6 3 (quality 20)

- *Tabu list:*
  - swap (4,5) (3 iterations)
  - swap (2,4) (2 iterations)
  - swap (3,1) (1 iterations)
- *Best 5 candidate moves in the 2-swap neighborhood:*
  - swap (7,1) (0)
  - swap (4,3) (-3)
  - swap (6,3) (-5)
  - swap (5,4) (-6, tabu)
  - swap (2,6) (-8)
- *Selected swap:* swap(7,1)

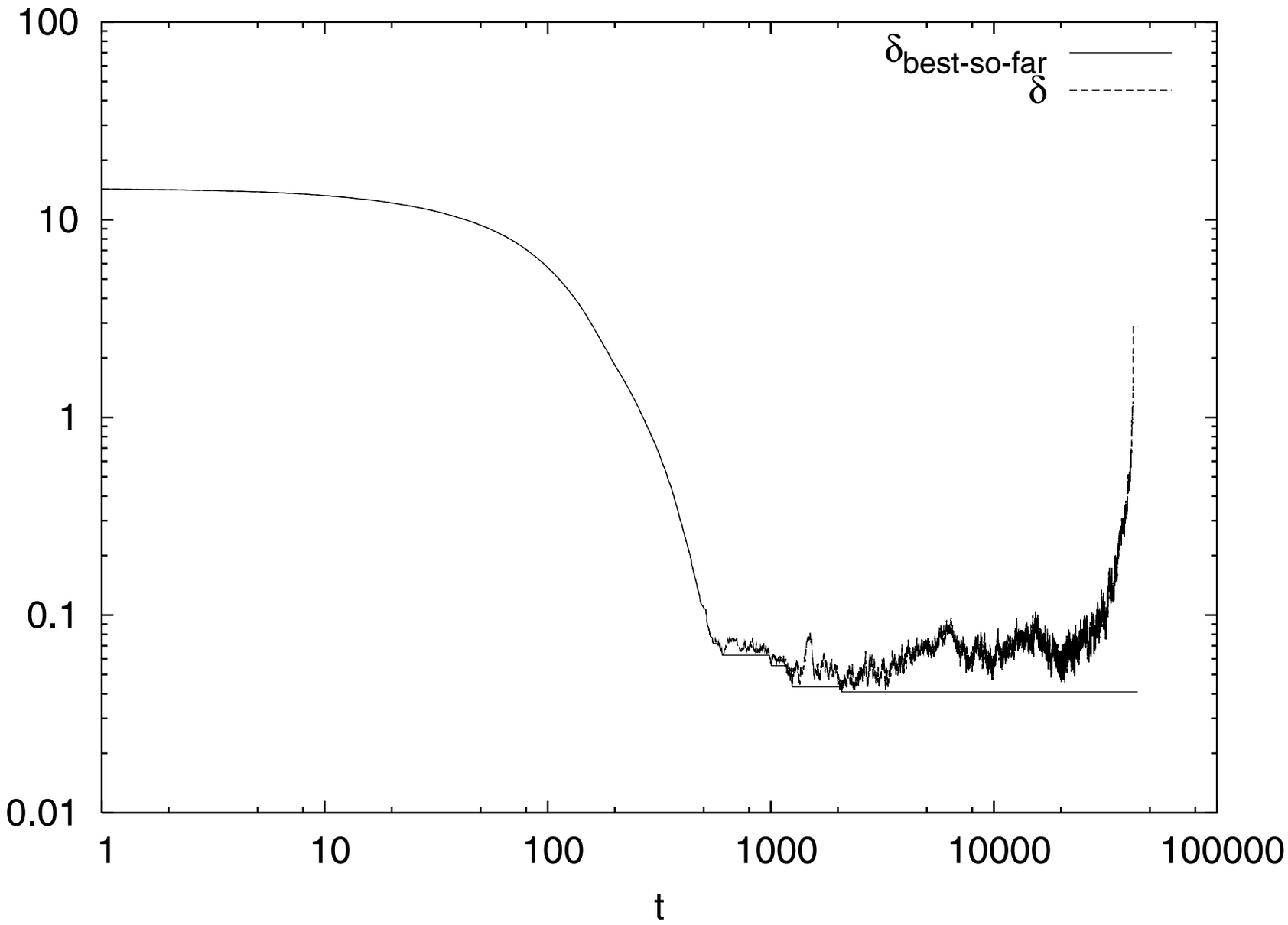
A more complex TSP example

- 2-exchange neighborhood
- At each step, $100n$ configurations are selected at random from current 2-exchange neighborhood
- *Critical questions:* According to which criteria tabu attributes should be defined? Why should a tentative new configuration be tabu?
  - *Answer:* The selected configuration contains properties that were part of former configurations; but as one intends to investigate as well other configurations without these properties, these properties and therefore also other configurations containing them, are declared tabu
  - In the case of the TSP, such properties are the edges used in the tour configurations. Thus, the tabu structure is defined through an edge matrix:

$$\eta(i,j) = \begin{cases} 0 & \text{if the edge between node } i \text{ and node } j \text{ is tabu} \\ 1 & \text{otherwise} \end{cases},$$

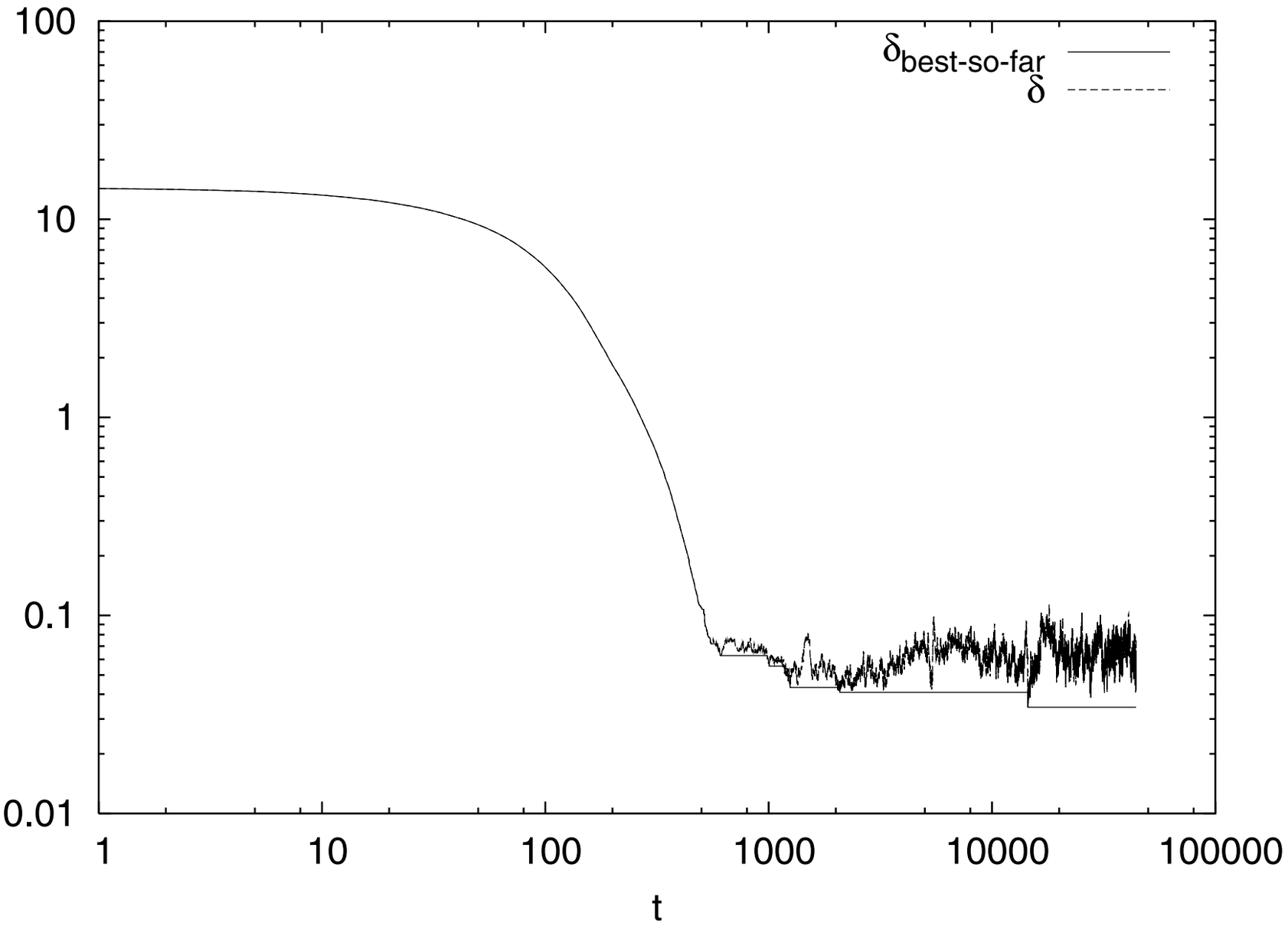
  - At the beginning: $\eta(i,j) = 1, \quad \forall i,j = 1,\ldots,n$
  - Starting from a random configuration, iteratively $100n$ random configurations are drawn from all possible solution in the current neighborhood. If (k,k_+) and (l,l_+) are the selected edges for the 2-opt move, then $\eta(k,l) = 0,\ \eta(l,k) = 0,\ \eta(k_+,l_+) = 0,\ \eta(l_+,k_+) = 0$
  - PCB442 instance from TSPLIB

No tabu tenure (unlimited tabu list)



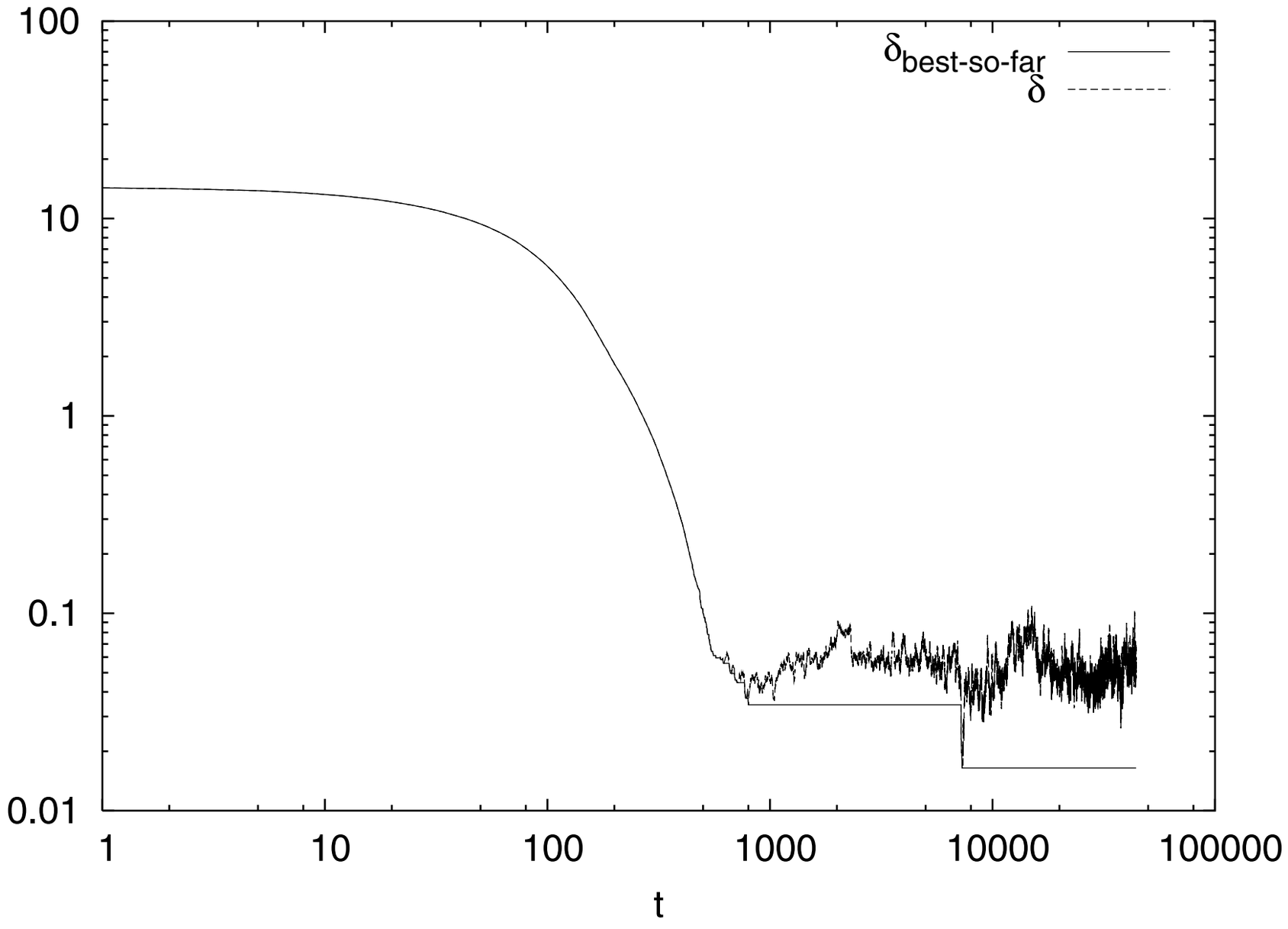
- Too many restrictions on edge selection: after a while no improvements are observed anymore

+ Tabu tenure



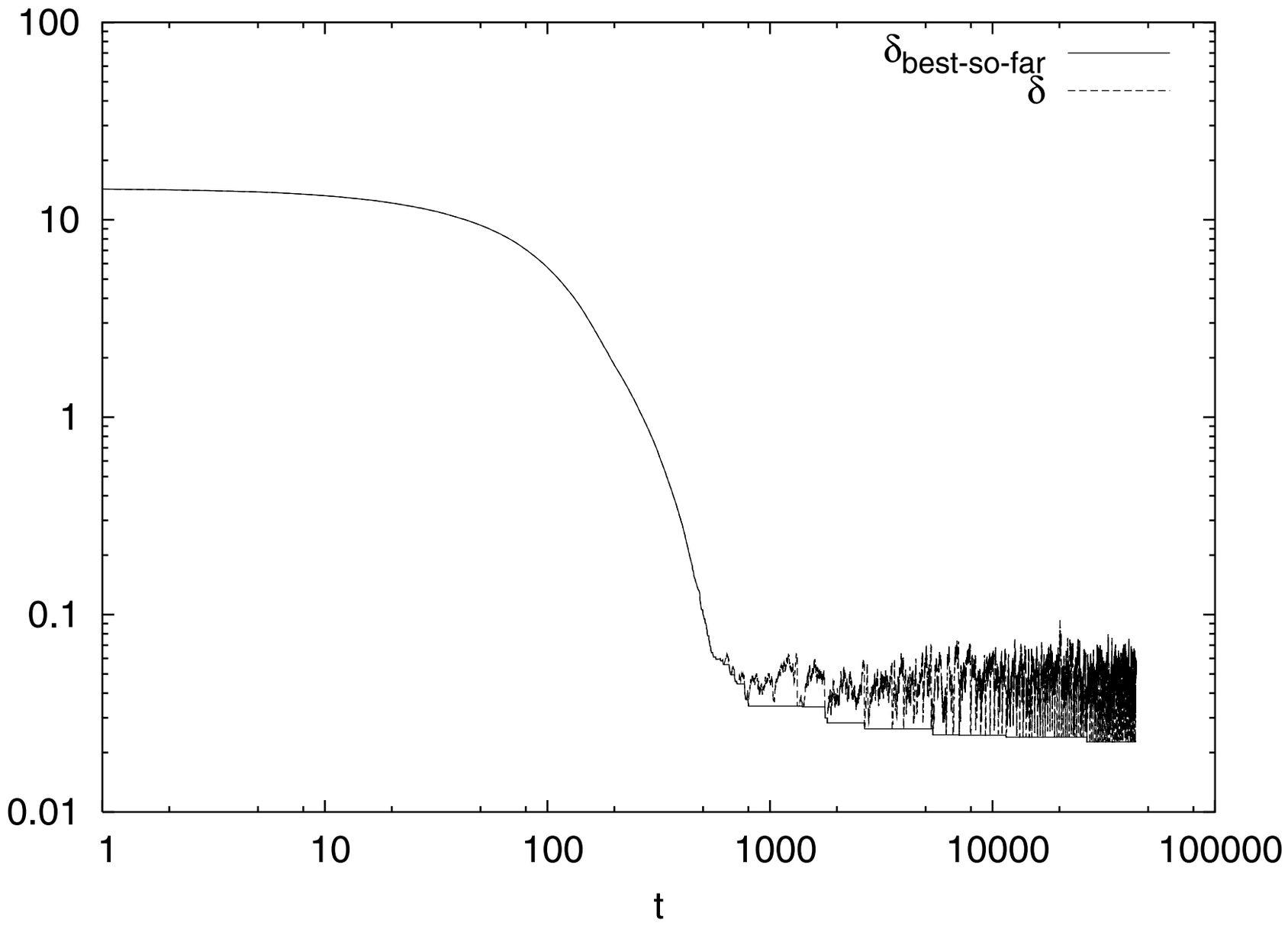
- Tabu tenure: constant, $\tau = 10n$
- The system does not 'freeze'

+ Aspiration criterion



- Aspiration: if it improves over the best so far configuration
- Further improvement of solution quality

+ Intensification - Diversification



- Intensification: if after a swap a new better solution hasn't been found, move to best so far solution
- Diversification: the best so far solution is taken, 10% of the edges are randomly removed, and the fragments are rejoined into a tour by using an insertion heuristic

Final considerations on Tabu search

- TS has been applied to a large variety of optimization problems
- A number of state-of-the-art algorithms
- TS performance critically depends on a number of parameters and components
- Not so easy to design and tune
- A number of reference articles and books / book chapters, the main reference is:
  - F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic, 1997