Heuristic Algorithms

Master's Degree in Computer Science/Mathematics

Roberto Cordone

DI - Università degli Studi di Milano

Schedule:	Wednesday 13.30 - 16.30 in classroom Alfa
	Thursday 09.30 - 12.30 in classroom Alfa
Office hours:	on appointment
E-mail:	roberto.cordone@unimi.it
Web page:	https://homes.di.unimi.it/cordone/courses/2026-ae/2026-ae.html
Ariel site:	https://myariel.unimi.it/course/view.php?id=7439

Extending the local search with worsenings

If the neighbourhood and objective remain the same,
the rule of acceptance must change: instead of

$$x' := \arg \min_{x \in N(x)} f(x)$$

select a nonminimal (possibly, even nonimproving) solution

The main problem is the risk of cyclically visiting the same solutions

The two main strategies that allow to control this risk are

- *Simulated Annealing* (*SA*), which uses randomisation to make repetitions unlikely
- *Tabu Search* (*TS*), which uses memory to forbid repetitions

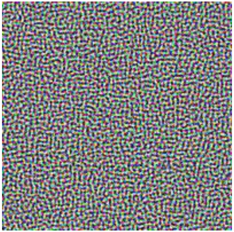
Annealing

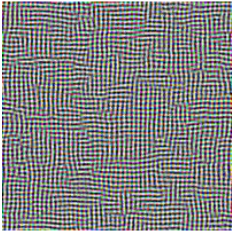
The *Simulated Annealing* (Kirkpatrick, Gelatt, and Vecchi, 1983) derives from Metropolis' algorithm (1953), which aims to simulate the "annealing" process of metals:

- bring the metal to a temperature close to fusion, so that its particles distribute at random
- cool the metal very slowly, so that the energy decreases, but in a time sufficiently long to converge to thermal equilibrium

The aim of the process is to obtain

- a very regular and defectless crystal lattice, that corresponds to the base state (minimum energy configuration)
- a material with useful physical properties





Simulation and optimisation

The situation has similarities with optimisation problems

- the states of the physical system correspond to the solutions
- the energy corresponds to the objective function
- the base state corresponds to the globally optimal solutions (minima)
- the state transitions correspond to local search moves
- the temperature corresponds to a numerical parameter

This suggests to use Metropolis' algorithm for optimisation

According to thermodynamics at the thermal equilibrium
the probability of observing each state i depends on its energy E_i

$$\pi'_T(i) = \frac{e^{\frac{-E_i}{k\,T}}}{\sum_{j \in S} e^{\frac{-E_j}{k\,T}}}$$

where S is the state set, T the temperature and k Boltzmann's constant

It is a dynamic equilibrium, with ongoing state transitions in all directions

Metropolis' algorithm

Metropolis' algorithm generates a random sequence of states

- the current state i has energy E_i
- the algorithm perturbs i, generating a state j with energy E_j
- the current state moves from i to j with probability

$$\pi_T(i,j) = \begin{cases} 1 & \text{if } E_j < E_i \\ e^{\frac{E_i - E_j}{kT}} = \frac{\pi'(j)}{\pi'(i)} & \text{if } E_j \geq E_i \end{cases}$$

that is the transition is

- deterministic if improving (because that is the final purpose)
- based on the conditional probability if worsening

Simulated Annealing applies exactly the same principle

General scheme of *Simulated Annealing*

Algorithm SimulatedAnnealing$\left(I, x^{(0)}, T^{[0]}\right)$

$x := x^{(0)}$; $x^* := x^{(0)}$; $T := T^{[0]}$;

While Stop() $=$ *false do*

  $x' :=$ RandomExtract(N, x); { random uniform extraction }

  If $f(x') < f(x)$ *or* $U[0;1] \leq e^{\frac{f(x)-f(x')}{T}}$ *then* $x := x'$;

  If $f(x') < f(x^*)$ *then* $x^* := x'$;

  $T :=$ Update(T);

EndWhile;

Return $(x^*, f(x^*))$;

As the neighbourhood is used to generate a solution (not fully explored), it is possible to worsen even if improving solutions exist

A precomputed table of values for $e^{\frac{\delta f}{T}}$ can improve the efficiency

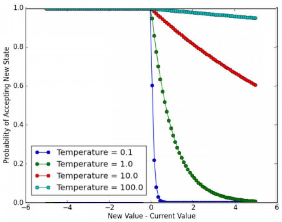
Several update schemes can be designed for the "temperature" T

Acceptance criterium

T rules the probability to accept worsenings

$$\pi_T(x, x') = \begin{cases} 1 & \text{if } f(x') < f(x) \\ e^{\frac{f(x)-f(x')}{T}} & \text{if } f(x') \geq f(x) \end{cases}$$

- $T \gg 0$ diversifies because nearly all solutions are accepted: in the extreme case, it is a *random walk*
- $T \approx 0$ intensifies because nearly all worsening solutions are rejected: in the extreme case, it is a *steepest descent*



Notice the similarity with the ILS

Asymptotic convergence to the optimum

Due to the acceptance rule, the current solution x is a random variable: its "state probability" $\pi'(x)$ combines on all possible predecessors $x^{(t-1)}$

- the "state probability" $\pi'\left(x^{(t-1)}\right)$ of the predecessor
- the probability to choose the move from $x^{(t-1)}$ to x, that is uniform
- the probability to accept the move, that is

$$\pi_T\left(x^{(t-1)}, x\right) = \begin{cases} 1 & \text{if } f(x) < f\left(x^{(t-1)}\right) \\ e^{\frac{f\left(x^{(t-1)}\right) - f(x)}{T}} & \text{if } f(x) \geq f\left(x^{(t-1)}\right) \end{cases}$$

As it depends only on the previous step, the solution is a Markov chain

For fixed temperature T, the transition probabilities are stationary: it is a homogeneous Markov chain

If the search graph for neighbourhood N is connected, the probability to reach each state is > 0: it is an irreducible Markov chain

Under these assumptions, the state probability converges to a stationary distribution independent from the starting state

Asymptotic convergence to the optimum

The stationary distribution favours “good” solutions with the same law imposed by thermodynamics on physical systems at thermal equilibrium

$$\pi_T(x) = \frac{e^{\frac{-f(x)}{T}}}{\sum_{x \in X} e^{\frac{-f(x)}{T}}} \quad \text{for each } x \in X$$

where X is the feasible region and T the “temperature” parameter

The distribution converges to a limit distribution as $T \to 0$

$$\pi(x) = \lim_{T \to 0} \pi_T(x) = \begin{cases} \frac{1}{|X^*|} & \text{for } x \in X^* \\ 0 & \text{for } x \in X \setminus X^* \end{cases}$$

which corresponds to a certain convergence to a globally optimal solution

Asymptotic convergence to the optimum

This result however holds at the equilibrium, in infinite time

In practice, low values of T imply

- a high probability to visit a global optimum, but also
- a slow convergence to the optimum (*many exchanges are rejected*)

In a finite time, the result obtained with low T can be far from optimal

Hence, T starts high and is progressively updated decreasing over time

The starting value $T^{[0]}$ should be

- high enough to allow to reach any solution quickly
- small enough to discourage visiting very bad solutions

A classical tuning for $T^{[0]}$ is to

- sample the first neighbourhood $N\left(x^{(0)}\right)$
- set a parameter $\beta \in (0,1)$
- set $T^{[0]}$ to accept on average a fraction β of the sampled solutions

Temperature update

The temperature is updated by subsequent phases ($r = 0, \ldots, m$)

- each phase applies a constant value $T^{[r]}$ for $\ell^{[r]}$ iterations
- $T^{[r]}$ decreases exponentially from phase to phase

$$T^{[r]} := \alpha T^{[r-1]} = \alpha^r T^{[0]} \text{ with } \alpha \in (0,1)$$

- $\ell^{[r]}$ increases from phase to phase (often linearly) with values related to the diameter of the search graph (therefore to the size of the instance)

Since T is variable, the Markov chain x is not homogeneous, but

- if T decreases slowly enough, it converges to the global optimum
- good parameters to tune the decrease depend on the instance (namely, on $f(\tilde{x}) - f(x^*)$, where $f(\tilde{x})$ is the second best value of f)

But the best parameter values are not known a priori

Adaptive SA variants tune the temperature T based on the results

- set T to a value such that a given fraction of $N(x)$ is accepted
- increase T if the solution has not improved for a certain time (diversification); otherwise decrease it (intensification)

Tabu Search

The *Tabu Search* (*TS*) has been proposed by Glover (1986)

It keeps the basic selection rule of *steepest descent*

$$x' := \arg \min_{x \in N(x)} f(x)$$

without the termination condition

But this implies cycling!

The *TS* imposes a tabu to forbid the solutions already visited

$$x' := \arg \min_{x \in N(x) \setminus X_V} f(x)$$

where X_V is the set of the already visited solutions

A simple idea, but how to manage the tabu efficiently and effectively?

Exchange heuristics with tabu

An exchange heuristic that explores a neighbourhood imposing a tabu on the already visited solutions requires to:

1. evaluate the feasibility of each subset produced by the exchanges (unless guaranteed *a priori*)
2. evaluate the cost of each feasible solution
3. evaluate the tabu status of each feasible promising solution

in order to select the feasible best nontabu solution

An elementary way to implement the evaluation of the tabu is

- save the visited solutions in a suitable structure (tabu list)
- check each explored solution making a query on the tabu list

Potential inefficiency of the tabu mechanism

This elementary evaluation of the tabu however is very inefficient

- the comparison of the solutions at step t requires time $O(t)$
  (*reducible with hash tables or search trees*)
- the number of solutions visited grows indefinitely over time
- the memory occupation grows indefinitely over time

The *Cancellation Sequence Method* and the *Reverse Elimination Method* tackle these problems, exploiting the fact that in general

- the solutions visited form a chain with small variations
- few solutions visited are neighbours of the current one

The idea is to focus on variations

- save move lists, instead of solutions
- evaluate the overall performed variations, instead of the single moves
- find the solutions which have undergone small overall variations
  (recent ones or submitted to variations subsequently reversed)

Potential ineffectiveness of the tabu mechanism

Other subtle phenomena influence the effectiveness of the method

Forbidding the solutions visited can have two different negative effects:

- it can disconnect the search graph,
  creating impassable "iron curtains" that block the search

  (*the prohibition should not be permanent*)

- it can slow down the exit from attraction basins,
  creating a "gradual filling" effect that slows down the search

  (*the prohibition should be extended*)

The two phenomena suggest apparently opposite remedies

How to combine them?

Example

A very degenerate example is provided by the following problem

- the ground set $B = \{1, \dots, n\}$ includes the first n natural numbers
- all subsets are feasible: $X = 2^B$
- the objective combines a nearly uniform additive term $\phi_i = 1 + \epsilon i$ $(0 < \epsilon \ll 1)$ and (only if $x = x^*$) a strong negative term

$$f(x) = \begin{cases} \sum_{i \in x} (1 + \epsilon i) & \text{for } x \neq x^* \\ -1 & \text{for } x = x^* \end{cases}$$

where x^* is suitably chosen in X

Using the neighbourhood of all solutions at Hamming distance ≤ 1

$$N_{H_1}(x) = \left\{ x' \in 2^B : d_H(x, x') \leq 1 \right\}$$

the problem has

- a global optimum x^*, with $f(x^*) = -1$,
  whose attraction basin includes the n solutions x with $d_H(x, x^*) \leq 1$
- a local optimum $\bar{x} = \emptyset$ with $f(\bar{x}) = 0$,
  whose attraction basin includes the other $2^n - n$ solutions

Example

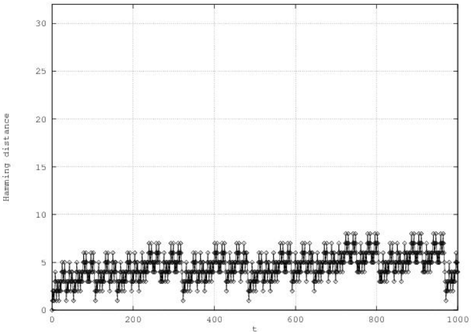
Starting from $x^{(0)} = \bar{x} = \emptyset$ and forbidding all the solutions visited:

- visit methodically most of 2^B, with f and $d(x, \bar{x})$ going up and down
- for $4 \leq n \leq 14$ the search graph is disconnected and the search is stuck (1011 can't be reached), but all solutions are at least explored
- for $n \geq 15$, the search is stuck and some unvisited solutions are not explored, possibly missing the optimum

t	f	x	$d(x, \bar{x})$
1	0	0000	0
2	$1+\epsilon$	1000	1
3	$2+3\epsilon$	1100	2
4	$1+2\epsilon$	0100	1
5	$2+5\epsilon$	0110	2
6	$1+3\epsilon$	0010	1
7	$2+4\epsilon$	1010	2
8	$3+6\epsilon$	1110	3
9	$4+10\epsilon$	1111	4
10	$3+9\epsilon$	0111	3
11	$2+7\epsilon$	0011	2
12	$1+4\epsilon$	0001	1
13	$2+5\epsilon$	1001	2
14	$3+7\epsilon$	1101	3
15	$2+6\epsilon$	0101	2

Example

The objective function profile confirms the limitations of the method



The solution x repeatedly gets far from $x^{(0)} = \bar{x}$ and close to it

- it visits nearly the whole attraction basin of $\bar{x}$
- in the end, it does not get out of it, but gets stuck in a solution whose neighbourhood is fully tabu
- if it removes the oldest tabu, the exploration goes around and the risk of looping gets back

Attribute-based tabu

Some simple devices can be adopted in order to control these problems

Forbidding only the visited solution slows down the search

1. forbid all solutions that share "attributes" with the visited ones, instead of forbidding only the visited solutions
   - define a set A of attributes
   - define for each solution $x \in X$ a subset of attributes $A_x \subseteq A$
   - declare a subset of tabu attributes $\bar{A} \subseteq A$ (empty at first)
   - forbid all the solutions with tabu attributes

$$x \text{ is tabu } \Leftrightarrow A_x \cap \bar{A} \neq \emptyset$$

   - move from the current solution x to x' such that $A_{x'} \cap \bar{A} = \emptyset$ and add to $\bar{A}$ the attributes possessed by x and not by x'

$$\bar{A} := \bar{A} \cup (A_x \setminus A_{x'})$$

(*in this way, x becomes tabu*)

This allows to

- avoid also solutions similar to the visited ones
- get more quickly far away from visited local optima

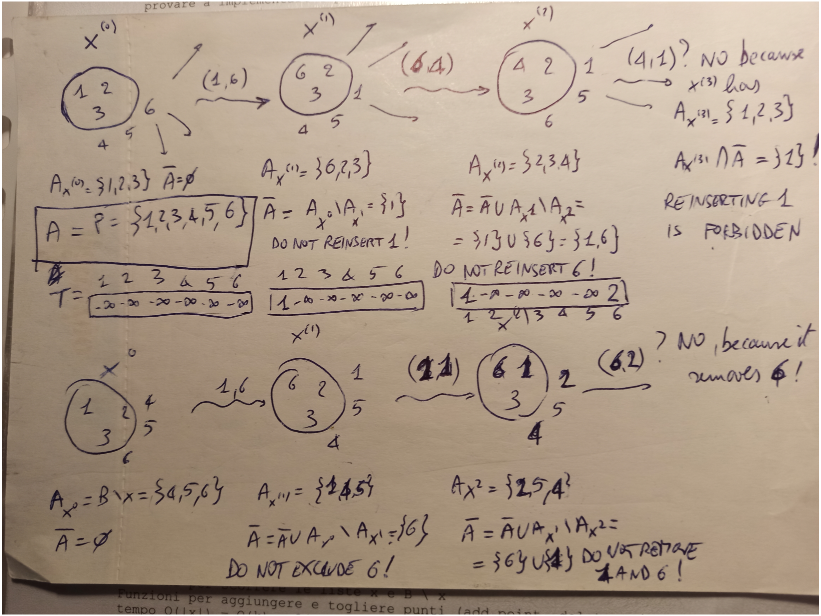
Tabu attributes

The concept of "attribute" is intentionally generic; the simpler ones are

- inclusion of an element in the solution ($A = B$ and $A_x = x$):
  when the move from x to x' expels an element i from the solution, the tabu forbids the reinsertion of i in the solution
  - x has the attribute "presence of i" and x' hasn't got it
  - the attribute "presence of i" enters $\bar{A}$
  - every solution including i becomes tabu
- exclusion of an element from the solution ($A = B$ and $A_x = B \setminus x$):
  when the move from x to x' inserts an element i into the solution, the tabu forbids the removal of i from the solution
  - x has the attribute "absence of i" and x' hasn't got it
  - the attribute "absence of i" enters $\bar{A}$
  - every solution devoid of i becomes tabu

Different attribute sets can be combined, each with its tenure and list (e.g., after replacing i with j, forbid both to remove j and to insert i)

Example



(1,6)
(6,4)
(4,1)? NO because
Ax(3) = {1,2,3}
Ax(3) ∩ Ā = {1}!
REINSERTING 1
IS FORBIDDEN
Ax(0) = {1,2,3} Ā = ∅
Ax(1) = {6,2,3}
Ax(2) = {2,3,4}
A = P = {1,2,3,4,5,6}
Ā = Ax0 \ Ax1 = {1}
DO NOT REINSERT 1!
Ā = Ā ∪ Ax1 \ Ax2 =
= {1} ∪ {6} = {1,6}
DO NOT REINSERT 6!
Ax0 = B \ x = {4,5,6}
Ā = ∅
Ā = Ā ∪ Ax0 \ Ax1 = {6}
DO NOT EXCLUDE 6!
Ā = Ā ∪ Ax1 \ Ax2 =
DO NOT REMOVE
1 AND 6!
NO, because it
removes 6!

Tabu attributes

Other (less frequent) examples of attributes

- the value of the objective function: forbid solutions of a given value, previously assumed by the objective
- the value of an auxiliary function

Complex attributes can be obtained combining simple attributes

- the coexistence in the solution of two elements (or their separation)
- or, if a move replaces element i with element j,
  the tabu can forbid the removal of j to include i,
  but allow the simple removal of j and the simple inclusion of i

Temporary tabu and aspiration criterium

Some simple devices can be adopted in order to control these problems

The tabu mechanism creates regions hard or impossible to reach

2. give a limited length L (tabu tenure) to the prohibition
   - the tabu solutions become feasible again after a while
   - the same solutions can be revisited

     (*but, if $\bar{A}$ is different, the future evolution will be different*)

Tuning the tabu tenure is fundamental for the effectiveness of *TS*

The tabu could forbid a global optimum similar to a visited solution

3. introduce an aspiration criterium: a tabu solution better than the best known one is anyway accepted

   (*of course, there is no risk of looping*)

There are looser aspiration criteria, but they are not commonly used

The tabu could forbid all neighbour solutions

4. if all neighbour solutions are tabu, accept the one with the oldest tabu (it can be interpreted as another aspiration criterium)

Efficient evaluation of the tabu status

The evaluation of the tabu status must be efficient and avoid scanning the whole solution (as for feasibility and cost)

- the attributes are associated to moves, not to solutions: do not check whether the solution includes i, but whether the move adds i

Let T_i be the iteration when attribute $i \in A$ became tabu ($-\infty$ if $i \notin \bar{A}$)

To evaluate the tabu status in constant time simply check

$$t \leq T_i + L$$

If the tabu is on insertions ($A = x$), at iteration t

- forbid the moves that add $i \in B \setminus x$ when $t \leq T_i^{\mathrm{in}} + L^{\mathrm{in}}$
- update $T_i^{\mathrm{in}} := t$ for each i removed ($i \in x \setminus x'$)

If the tabu is on deletions ($A = B \setminus x$), at iteration t

- forbid the moves that delete $i \in x$ when $t \leq T_i^{\mathrm{out}} + L^{\mathrm{out}}$
- update $T_i^{\mathrm{out}} := t$ for each i added ($i \in x' \setminus x$)

As either $i \in x$ or $i \in B \setminus x$, a single vector T is enough for both checks

More sophisticated attributes require more complex structures

General scheme of the *TS*

Algorithm TabuSearch$\left(I, x^{(0)}, L\right)$

$x := x^{(0)}$; $x^* := x^{(0)}$;

$\bar{A} := \emptyset$;

While Stop() $=$ *false do*

  $f' := +\infty$;

  For each $y \in N(x)$ *do*

    If $f(y) < f'$ *then*

      If Tabu$\left(y, \bar{A}\right) =$ *false* *or* $f(y) < f(x^*)$ *then* $x' := y$; $f' := f(y)$;

    EndIf

  EndFor

  $\bar{A} :=$ Update$\left(\bar{A}, x', L\right)$;

  If $f(x') < f(x^*)$ *then* $x^* := x'$;

EndWhile

Return $(x^*, f(x^*))$;

Example: the *TSP*

Consider the neighbourhood $N_{\mathcal{R}_2}$ generated by 2-opt exchanges and use as attributes both the presence and the absence of arcs in the solution

- at first set $T_{ij} = -\infty$ for each arc $(i,j) \in A$
- at each step t, explore the $n(n-1)/2$ pairs of removable arcs and the corresponding pairs of arcs which would replace them
- the move (i,j), which replaces (s_i, s_{i+1}) and (s_j, s_{j+1}) with (s_i, s_j) and (s_{i+1}, s_{j+1}), is tabu at step t if one of the following conditions holds:
  1. $t \leq T_{s_i, s_{i+1}} + L^{\text{out}}$
  2. $t \leq T_{s_j, s_{j+1}} + L^{\text{out}}$
  3. $t \leq T_{s_i, s_j} + L^{\text{in}}$
  4. $t \leq T_{s_{j+1}, s_{i+1}} + L^{\text{in}}$

  So, at first all moves are legal
- selected move (i^*, j^*), update the auxiliary structures setting
  1. $T_{s_{i^*}, s_{i^*+1}} := t$
  2. $T_{s_{j^*}, s_{j^*+1}} := t$
  3. $T_{s_{i^*}, s_{j^*}} := t$
  4. $T_{s_{j^*+1}, s_{i^*+1}} := t$

As n arcs are in and $n(n-2)$ out of the solution, it is better to set $L^{\text{out}} \ll L^{\text{in}}$

Example: the *Max-SAT*

Consider the neighbourhood $N_{\mathcal{F}_1}$, which includes the solutions obtained complementing the value of a variable (all n solutions are feasible)

Since $|x| = |B \setminus x|$ for each $x \in X$

- the tabu tenure for additions and deletions can be the same
- it is sufficient to forbid the change of value of a variable
  and the attribute is the variable

The algorithm proceeds as follows

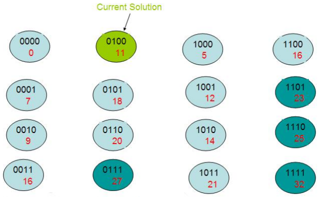
- at first, set $T_i = -\infty$ for each variable $i = 1, \ldots, n$
- at each step t, explore the n solutions obtained complementing each variable
- the move i, which assigns $x_i := \bar{x}_i$, is tabu at step t if $t \leq T_i + L$
  (*at first all moves are nontabu*)
- perform move i^* and set $T_{i^*} := t$

Example: the *KP*

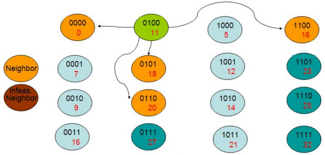
The neighbourhood $N_{\mathcal{H}_1}$ includes all solutions at Hamming distance ≤ 1

Use the object as an attribute, with equal tenures $L^{\text{in}} = L^{\text{out}} = 3$:
vector T saves the iteration of the last move performed on each $i \in B$

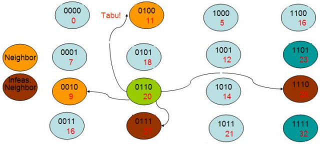
$t = 0$



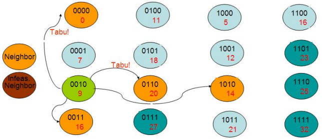
$t = 1 \quad T = [-\infty \ -\infty \ -\infty \ -\infty]$



$t = 2 \quad T = [-\infty \ -\infty \quad 1 \ -\infty]$



$t = 3 \quad T = [-\infty \quad 2 \quad 1 \ -\infty]$



Tuning the *tabu tenure*

The value of the *tabu tenure* L is a crucial parameter

- too large tenures can conceal the global optimum and in the worst case block the search
- too small tenures can hold the exploration back in useless regions and in the worst case produce cyclic behaviours

The most effective value of L is in general

- related to the size of the instance
- slowly growing with size (many authors suggest $L \in O\left(\sqrt{|A|}\right)$)
- but nearly constant on medium ranges of size

Cycles can be broken extracting L at random in a range $[L_{\min}; L_{\max}]$

Adaptive mechanisms update L based on the results of the search within a given range $[L_{\min}; L_{\max}]$

- decrease L when the current solution x improves: the search is probably approaching a new local optimum and we want to favour it (intensification)
- increase L when the current solution x worsens: the search is probably leaving a known local optimum and we want to speed up (diversification)

Variants

Other adaptive strategies work in the long term:

- reactive Tabu Search:
  - use efficient structures to save the solutions visited (*hash table*)
  - detect cyclic behaviours (frequent repetitions)
  - move the range $[L_{\min}; L_{\max}]$ upwards if the solutions repeat too often
- frequency-based Tabu Search:
  - save the frequency of each attribute in the solution in structures similar to the ones used for the tenure (e.g., F_i for each $i \in B$)
  - if an attribute appears very often
    - favour the moves introducing it modifying f as in the *DLS*
    - forbid the moves introducing it, or discourage them by modifying f
- Exploring Tabu Search: reinitialize the search from solutions of good quality which have been explored, but not used as current solution (*i. e., the "second-best solutions" of some neighbourhood*)
- Granular Tabu Search: enlarge or reduce the neighbourhood progressively