Heuristic Algorithms

Master's Degree in Computer Science/Mathematics

Roberto Cordone

DI - Università degli Studi di Milano

Schedule:	Thursday 14.30 - 16.30 on MS-Teams
	Friday 14.30 - 16.30 on MS-Teams
Office hours:	on appointment
E-mail:	roberto.cordone@unimi.it
Web page:	https://homes.di.unimi.it/cordone/courses/2020-ae/2020-ae.html
Ariel site:	https://rcordoneha.ariel.ctu.unimi.it

Constructive metaheuristics

The constructive algorithms have strong limitations on many problems

What can be done, without abandoning the general scheme?

Iterate the scheme to generate many (potentially) different solutions

- the efficiency decreases: the computational times are summed
- the effectiveness increases: the best solution is returned

The trade-off must be carefully tuned

The iterated scheme can apply

- multi-start, that is different algorithm at each iteration $l = 1, \ldots, \ell$ (this requires to define multiple $\mathcal{F}_{A_l}$ and φ_{A_l})

but it is more flexible to apply metaheuristics, that exploit

- randomization (operations based on a random seed), as in the case of semigreedy algorithms, *GRASP* and *Ant System* (partly, *ART*)
- memory (operations based on the solutions of previous iterations), as in the case of *ART*, cost perturbation and *Ant System*

Termination condition

The iterated scheme can ideally be infinite

In pratice, one uses "absolute" termination conditions

1. a given total number of explorations of the neighbourhood or
   a given total number of repetitions of the local search
2. a given total execution time
3. a given value of the objective
4. a given improvement of the objective with respect to the starting solution

or "relative" termination conditions

1. a given number of explorations of the neighbourhood or repetitions after the last improvement of f^*
2. a given execution time after the last improvement
3. a given minimum value of the ratio between improvement of the objective and number of explorations or execution time
   (*e.g.: f^* improves less than* 1% *in the last* 1 000 *explorations*)

Fair comparisons require absolute conditions (time or number of explorations)

Multi-start

Multi-start (or restart) is a classical, very simple and natural approach:

- define different search spaces $\mathcal{F}_{A^{[l]}}$ and selection criteria $\varphi_{A^{[l]}}(i,x)$
- apply each resulting algorithm $A^{[l]}$ to obtain $x^{[l]}$
- return the best solution $x = \arg\min_{l=1,\ldots,\ell} f\left(x^{[l]}\right)$

A typical case is to tune $\varphi_A(i,x)$ with numerical parameters μ

The construction graph can model this situation with

- including all nodes and arcs admitted by at least one algorithm $A^{[l]}$:

$$\mathcal{F}_A = \bigcup_{l=1}^{\ell} \mathcal{F}_{A^{[l]}}$$

- setting arc weights depending on l: $\varphi_A(i,x,l) = \varphi_{A^{[l]}}(i,x)$
- setting an infinite arc weight for the arcs that are forbidden in a specific algorithm $A^{[l]}$: $\varphi_A(i,x,l) = +\infty$

Example

A whole family of heuristics for the *TSP* can be obtained setting:

- insertion criterium:

$$i_k^* = \arg \min_{i \in \{1,\ldots,|x|\}} \gamma_{i,k} = \mu_1 \left(c_{s_i,k} + c_{k,s_{i+1}} \right) - (1 - \mu_1)\, c_{s_i,s_{i+1}}$$

  where $\mu_1 \in [0;1]$ tunes the relative strength of the
    - increase in cost due to the added node k
    - decrease in cost due to the removed edge (s_i, s_{i+1})

- selection criterium:

$$k^* = \arg \min_{k \in N \setminus N_x} \varphi_A(k,x) = \mu_2\, d(x,k) - \mu_3\, d(x,k) + (1 - \mu_2 - \mu_3)\, \gamma_{i_k^*,k}$$

  where $\mu_2, \mu_3 \in [0;1]$ tune the relative strength (and sign) of the
    - distance of the added node k from the current circuit x
    - increase in cost due to the added node k

to get *CI* for $\mu = (1/2, 0, 0)$, *NI* for $\mu = (1/2, 1, 0)$, *FI* for $\mu = (1/2, 0, 1)$

Constructive metaheuristics

The main constructive metaheuristics are

1. Adaptive Research Technique (*ART*) or Tabu Greedy:
   forbid some moves based on the solutions of the previous iterations

$$\min_{i:x\cup\{i\}\in\mathcal{F}^{[l]}} \varphi_A(i,x) \qquad \text{with } \mathcal{F}^{[l]} = \mathcal{F}^{[l]}\left(x_A^{[1]},\ldots,x_A^{[l-1]}\right) \subseteq \mathcal{F}$$

This is much less popular than the other two

2. semigreedy and GRASP: use a randomized selection criterium

$$\min_{i:x\cup\{i\}\in\mathcal{F}} \varphi_A^{[l]}\left(i,x,\omega^{[l]}\right)$$

3. Ant System (AS): use a randomized selection criterium
   depending on the solutions of the previous iterations

$$\min_{i:x\cup\{i\}\in\mathcal{F}} \varphi_A^{[l]}\left(i,x,\omega^{[l]},x_A^{[1]},\ldots,x_A^{[l-1]}\right)$$

New information on the arcs of the construction graph guides the search
The *ART* uses memory, the *GRASP* randomization, the *AS* both

Adaptive Research Technique

It was proposed by Patterson et al. (1998) for the *CMSTP*

When deceivingly good elements are included in the first steps the final solution can be quite bad; to try and avoid that

- the roll-out approach makes a look-ahead on each possible element (*but a single step can be insufficient to identify the misleading ones*)
- the *ART* forbids some elements to drive subset x on the right path in the search space

(*how to identify the misleading elements?*)

The aim is diversification: forbidding elements of the previous solutions guarantees to obtain different solutions

The prohibitions are temporary, with an expiration time of L iterations; otherwise, building feasible solutions would become impossible

Adaptive Research Technique

Define a basic constructive heuristic A

Let T_i be the starting iteration of the prohibition for each element $i \in B$ and x^* be the best solution found

Set $T_i = -\infty$ for all $i \in B$ to indicate that no element is forbidden

At each iteration $l \in \{1, \ldots, \ell\}$

1. apply heuristic A forbidding all elements i such that $l \leq T_i + L$ (all prohibitions older than L iterations automatically expire); let $x^{[l]}$ be the resulting solution
2. if $x^{[l]}$ is better than x^*, set $x^* := x^{[l]}$ and save $T_i - l$ for all $i \in B$
3. decide which elements to forbid and set $T_i = l$ for them: each element is forbidden with probability π (*any better ideas?*)
4. make minor tweaks to L, π or T_i

At the end, return x^*

Example: *ART* for the *SCP*

c	25	6	8	24	12

A	1	1	0	0	0
	1	1	0	0	0
	1	1	1	0	0
	1	0	1	1	0
	1	0	0	1	0
	1	0	0	0	1

1. the basic heuristic finds the solution $x^{[1]} = \{2, 3, 5, 4\}$ of cost $f\left(x^{[1]}\right) = 50$; forbid (at random) column 2
2. the basic heuristic finds the solution $x^{[2]} = \{3, 1\}$ of cost $f\left(x^{[2]}\right) = 33$ forbid (at random) column 3
3. the basic heuristic finds the solution $x^{[3]} = \{1\}$ of cost $f\left(x^{[2]}\right) = 25$, that is optimal
4. ...

An unlucky run could forbid column 1 *after step* 2

Intensification

The *ART* has three basic parameters

- the total number of iterations ℓ (*tuned mainly by the available time*)
- the length L of the prohibition
- the probability π of the prohibition

An excessive diversification can hinder the discovery of the optimum

Intensification aims to focus the search on the more promising subsets

Diversification and intensification play complementary roles

Intensification can be obtained tuning the parameters based on

- problem data: assign to promising elements (e. g., cheapest)
  - a smaller probability π_i to be forbidden
  - a shorter expiration time L_i of the prohibition
- memory: for promising elements (e. g., appearing in the best known solutions)
  - reduce L_i (if $L_i = 0$, i is never forbidden)
  - periodically restart the algorithm with the $T_i - I$ values associated with the best known solution, instead of $T_i = -\infty$

Parameter tuning

How to assign effective values to the parameters?

The experimental comparison of different values is necessary but complex

1. it requires long experimental campaigns, because
   the number of configurations grows combinatorially with
   - the number of parameters
   - the number of tested values for each parameter
     (*the more sensitive the result, the more values must be tested*)
2. it risks overfitting, that is labelling as absolutely good
   values which are good only on the benchmark istances considered

The excess of parameters is an undesirable aspect, and often reveals an insufficient study of the problem and of the algorithm

More on this point later

Semi-greedy heuristics

A nonexact constructive algorithm has at least one step t which builds a subset $x^{(t)}$ not included in any optimal solution

Since the element selected is the best according to the selection criterium

$$i^* = \arg \min_{i \in \Delta_A^+(x)} \varphi_A(i,x)$$

necessarily $\varphi_A(i,x)$ is incorrect, but probably not completely wrong

The semi-greedy algorithm (Hart and Shogan, 1987) assumes that elements that lead to the optimum are very good for $\varphi_A(i,x)$, even if not strictly the best

How to know which one?

If it is not possible to refine $\varphi_A(i,x)$

- define a suitable probability distribution on $\Delta_A^+(x)$ favouring the elements with the best values of $\varphi_A(i,x)$
- select $i^*(\omega)$ according to the distribution function

Semi-greedy heuristics

Since the set of alternative choices is finite, this means to assign

- probability $\pi_A(i,x)$ to arc $(x, x \cup \{i\})$ of the construction graph (with a sum equal to 1 for the outgoing arcs of each node)

$$\sum_{i \in \Delta_A^+(x)} \pi_A(i,x) = 1 \quad \text{for all } x \in \mathcal{F}_A : \Delta_A^+(x) \neq \emptyset$$

- higher probabilities to the better elements for the selection criterium

$$\varphi_A(i,x) \leq \varphi_A(j,x) \Leftrightarrow \pi_A(i,x) \geq \pi_A(j,x)$$

for each $i, j \in \Delta_A^+(x), x \in \mathcal{F}_A$

This heuristic approach has important properties

- it can reach an optimal solution if there is a path from $\emptyset$ to X^*

  (*this is a basic condition*)

- it can be reapplied several times obtaining different solutions and the probability to reach the optimum grows gradually

  (*each time the probability of always taking wrong ways decreases*)

Convergence to the optimum

The probability of

- following a path γ is the product of the probabilities on the arcs

$$\prod_{(y, y \cup \{i\} \in \gamma)} \pi_A(i, y)$$

- obtaining a solution x is the sum of those of the paths Γ_x reaching x

$$\sum_{\gamma \in \Gamma_x} \prod_{(y, y \cup \{i\} \in \gamma)} \pi_A(i, y)$$

This implies that the probability to reach the optimum:

1. is nonzero if and only if there exists a path of nonzero probability from $\emptyset$ to X^*
2. increases as $\ell \to +\infty$

(*the probability of not reaching it decreases gradually*)

It tends to 1 for probabilistically approximatively complete algorithms

Convergence to the optimum

In this context, a *random walk* is a constructive metaheuristic in which all the arcs going out of the same node have equal probability

- it finds a path to the optimum with probability 1 (if one exists)
- the time required can be extremely long

The exhaustive algorithm is exact and requires finite time

A deterministic constructive heuristic sets all probabilities to zero except for those on the arcs of a single path

- it finds the optimum only if it enjoys specific properties

Randomized heuristics that favour promising arcs and penalize the others

- accelerate the average convergence time
- decreases the guarantee of convergence in the worst case

There is a trade-off between expected and worst result

Arcs with zero probability can block the path to the optimum

Arcs with probability converging to zero reduce the probability to find it

Semi-greedy and GRASP

GRASP, that is Greedy Randomized Adaptive Search Procedure (Feo and Resende, 1989) is a sophisticated variant of the semi-greedy heuristic

- *Greedy* indicates that it uses a constructive basic heuristic
- *Randomized* indicates that the basic heuristic makes random steps
- *Adaptive* indicates that the heuristic uses an adaptive selection criterium $\varphi_A(i, x)$, depending also on x (not strictly necessary)
- *Search* indicates that it alternates the constructive heuristic and an exchange heuristic (differently from the semi-greedy approach)

The use of auxiliary exchange heuristics allows strongly better results

This aspect will be investigated in the following lessons

What probability function?

Several functions $\pi_A(i,x)$ are monotonous with respect to $\varphi_A(i,x)$

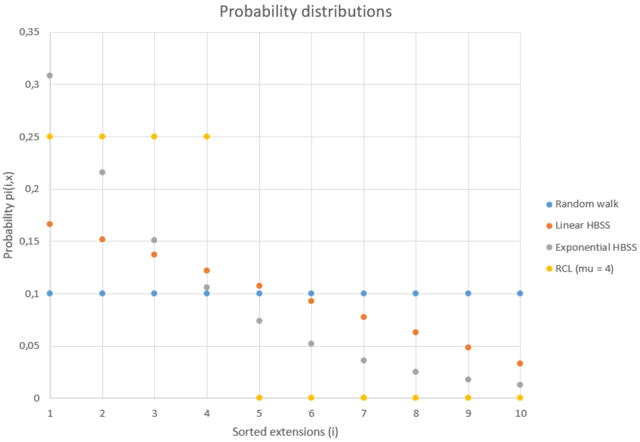
$$\varphi_A(i,x) \leq \varphi_A(j,x) \Leftrightarrow \pi_A(i,x) \geq \pi_A(j,x)$$

- uniform probability: each arc going out of x has the same $\pi_A(i,x)$; the algorithm performs a random path in $\mathcal{F}_A$ (random walk)
- Heuristic-Biased Stochastic Sampling (*HBSS*):
  - sort the arcs going out of x by nonincreasing values of $\varphi_A(i,x)$
  - assign a decreasing probability according to the position in the order based on a simple scheme (linear, exponential, ecc. . .)
- Restricted Candidate List (*RCL*):
  - sort the arcs going out of x by nonincreasing values of $\varphi_A(i,x)$
  - insert the best arcs in a list (*How many?*)
  - assign uniform probability to the arcs of the list, zero to the others

The most common strategy is the *RCL*, even if the zero probability arcs potentially cancel the global convergence to the optimum

Common probability functions

Suppose that at the current step $\left|\Delta_A^+(x)\right| = 10$ elements can be added



Definition of the RCL

Two main strategies are used to define the *RCL*

- cardinality: the RCL includes the best μ elements of $\Delta_A^+(x)$, where $\mu \in \{1, \ldots, |\Delta_A^+(x)|\}$ is a parameter fixed by the user
  - $\mu = 1$ yields the constructive basic heuristic
  - $\mu = |B|$ (i, e., $|\Delta_A^+(x)|$ for each x) yields the *random walk*
- value: the RCL includes all the elements of $\Delta_A^+(x)$ whose value is between $\varphi_{\min}$ and $(1-\mu)\varphi_{\min} + \mu\varphi_{\max}$ where

$$\varphi_{\min}(x) = \min_{i \in \Delta_A^+(x)} \varphi_A(i,x) \qquad \varphi_{\max}(x) = \max_{i \in \Delta_A^+(x)} \varphi_A(i,x)$$

  and $\mu \in [0;1]$ is a parameter fixed by the user
  - $\mu = 0$ yields the constructive basic heuristic
  - $\mu = 1$ yields the *random walk*

General scheme of *GRASP*

Algorithm GRASP(I)

$x^* := \emptyset$; $f^* := +\infty$; { Best solution found so far }

For $l = 1$ *to* ℓ *do*

{ Constructive heuristic with random steps }

$x := \emptyset$;

While $\Delta_A^+(x) \neq \emptyset$ *do*

$\varphi_i := \varphi_A(i, x)$ for each $i \in \Delta_A^+(x)$

$\pi := \text{AssignProbabilities}\left(\Delta_A^+(x), \varphi, \mu\right)$;

$i := \text{RandomExtract}(\Delta_A^+(x), \pi)$;

$x := x \cup \{i\}$;

EndWhile;

$x := \text{Search}(x)$;

If $x \in X$ and $f(x) < f^*$ *then* $x^* := x$; $f^* := f(x)$;

EndFor;

Return (x^*, f^*);

Example: *GRASP* for the *SCP*

c	25	6	8	24	12

A	1	1	0	0	0
	1	1	0	0	0
	1	1	1	0	0
	1	0	1	1	0
	1	0	0	1	0
	1	0	0	0	1

1. start with the empty subset: $x^{(0)} = \emptyset$
2. build a *RCL* with $\mu = 2$ candidates: columns 2 ($\varphi_2 = 2$) and 3 ($\varphi_3 = 4$); select (at random) column 3
3. build a *RCL* with $\mu = 2$ candidates: columns 2 ($\varphi_2 = 3$) and 1 ($\varphi_3 = 6.25$); select (at random) column 1
4. the solution obtained is $x = \{3, 1\}$ of cost $f(x) = 33$

With $\mu = 2$, the optimal solution cannot be obtained; with $\mu = 3$ it can

In fact, it is also possible with $\mu = 2$ if a destructive phase is applied

Reactive semi-greedy algorithm

Once again there are parameters to tune:

- the number of iterations ℓ
- the value μ determining the size of the *RCL*

An idea to exploit memory is to learn from the previous results

1. select m configurations of parameters $\mu_1, \ldots, \mu_m$ and set $\ell_r = \ell/m$
2. run each configuration μ_r for ℓ_r iterations
3. evaluate the mean $\bar{f}(\mu_r)$ of the results obtained with μ_r
4. update the number of iterations ℓ_r for each μ_r based on $\bar{f}(\mu_r)$

$$\ell_r = \frac{\frac{1}{\bar{f}(\mu_r)}}{\sum_{s=1}^{m} \frac{1}{\bar{f}(\mu_s)}} \ell \quad \text{for } r = 1, \ldots, m$$

   increasing it for the more effective configurations

5. repeat the whole process, going back to point 2, for R times

   Other scheme use scores based on the number of best known results

Cost perturbation methods

Instead of forbidding/forcing some choices, or modifying their probability, it is possible to modify the appeal of the available choices

Given a basic constructive heuristic A, at each step of iteration l

- tune the selection criterium $\varphi_A(i,x)$ with a factor $\tau_A^{[l]}(i,x)$

$$\psi_A^{[l]}(i,x) = \frac{\varphi_A(i,x)}{\tau_A^{[l]}(i,x)}$$

- update $\tau_A^{[l]}(i,x)$ based on the previous solutions $x^{[1]},\ldots,x^{[l-1]}$

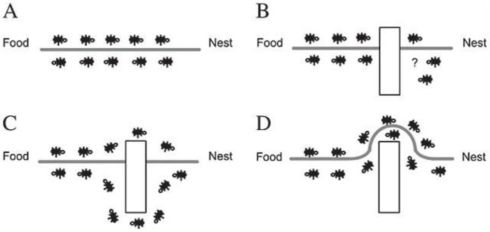
The elements with a better $\varphi_A(i,x)$ tend to be favoured,
but $\tau_A^{[l]}(i,x)$ tunes this effect, promoting

- intensification if $\tau_A^{[l]}(i,x)$ increases for the most frequent elements; this favours solutions similar to the previous ones
- diversification if $\tau_A^{[l]}(i,x)$ decreases for the most frequent elements; this favours solutions different from the previous ones

Ant Colony Optimization

It was devised by Dorigo, Maniezzo and Colorni in 1991
drawing inspiration from the social behaviour of ants

Stigmergy = indirect communication among different agents who are influenced by the results of the actions of all agents



Each agent is an application of the basic constructive heuristic

- it leaves a trail on the data depending on the solution generated
- it performs choices influenced by the trails left by the other agents

The choices of the agent have also a random component

Trail

As in the semi-greedy heuristic

- a basic constructive heuristic A is given
- each step performs a partially random choice

Differently from the semi-greedy heuristic

- each iteration l runs h times heuristic A (population)
- all the choices of $\Delta_A^+(x)$ are feasible (*there is no RCL*)
- the probability $\pi_A(i,x)$ depends on
  1. the selection criterium $\varphi_A(i,x)$
  2. auxiliary information $\tau_A(i,x)$ denoted as trail produced in previous iterations (sometimes by other agents in the same iteration)

The trail is uniform at first ($\tau_A(i,x) = \tau_0$), and later tuned

- increasing it to favour promising choices
- decreasing it to avoid repetitive choices

For the sake of simplicity, the trail $\tau_A(i,x)$ is not associated to each arc $(x, x \cup \{i\})$, but is the same for blocks of arcs (e.g., depending only on i)

Random choice

Instead of selecting the best element according to criterium $\varphi_A(i,x)$, i is extracted from $\Delta_A^+(x)$ with probability

$$\pi_A(i,x) = \frac{\tau_A(i,x)^{\mu_\tau}\,\eta_A(i,x)^{\mu_\eta}}{\sum\limits_{j\in\Delta_A^+(x)} \tau_A(j,x)^{\mu_\tau}\,\eta_A(i,x)^{\mu_\eta}}$$

where

- the denominator normalizes the probability
- the visibility is the auxiliary function

$$\eta_A(i,x) = \begin{cases} \varphi_A(i,x) & \text{for maximisation problems} \\ \dfrac{1}{\varphi_A(i,x)} & \text{for minimisation problems} \end{cases}$$

The promising choices have larger visibility

- the parameters μ_τ and μ_η tune the weights of the two terms

Balancing given and learned information

$$\pi_A(i,x) = \frac{\tau_A(i,x)^{\mu_\tau}\,\eta_A(i,x)^{\mu_\eta}}{\sum_{j\in\Delta_A^+(x)} \tau_A(j,x)^{\mu_\tau}\,\eta_A(i,x)^{\mu_\eta}}$$

Parameters μ_η and μ_τ tune the weight of the data and of memory

- $\mu_\eta \approx 0$ and $\mu_\tau \approx 0$ push towards randomness
- $\mu_\eta \gg \mu_\tau$ favours the data, simulating the basic constructive heuristic which makes sense when the known solutions are not very significant
- $\mu_\eta \ll \mu_\tau$ favours memory, keeping close to the previous solutions which makes sense when the known solutions are very significant

Balancing given and learned information

The Ant Colony System simplifies the scheme setting $\mu_\eta = \mu_\tau = 1$

$$\pi_A(i,x) = \frac{\tau_A(i,x)\,\eta_A(i,x)}{\sum\limits_{j \in \Delta_A^+(x)} \tau_A(j,x)\,\eta_A(i,x)}$$

and selecting the new element $i^{(t)}$ at step t

- at random with probability q
- optimizing $\varphi_A(i,x)$ with probability $(1-q)$

so that

- $q \approx 0$ favours the data
- $q \approx 1$ favours the memory

Trail update

At each iteration ℓ

1. run h istances of the basic heuristic A
2. select a subset $\tilde{X}^{[l]}$ of the solutions obtained,
   in order to favour their elements in the following iterations
3. update the trail according to the formula

$$\tau_A(i,x) := (1-\rho)\,\tau_A(i,x) + \rho \sum_{y \in \tilde{X}^{[l]} : i \in y} F_A(y)$$

where

- $\rho \in [0;1]$ is an oblivion parameter
- $F_A(y)$ is a fitness function expressing the quality of solution y
  (such that $F > \tau$: e.g., $F(y) = Q/f(y)$ for a suitable constant Q)

The purpose of the update is to

1. increase the trail on the elements of specific solutions $(y \in \tilde{X}^{[l]})$
2. decrease the trail on the other elements

The oblivion parameter

$$\tau_A(i,x) := (1-\rho)\,\tau_A(i,x) + \rho \sum_{y \in \tilde{X}^{[l]} : i \in y} F_A(y)$$

The oblivion parameter $\rho \in [0;1]$ tunes the behaviour of the algorithm:

- diversification: a high oblivion ($\rho \approx 1$) cancels the current trail based on the intuition that
  - the solutions obtained are not trustworthy
  - different solutions should be explored
- intensification: a low oblivion ($\rho \approx 0$) preserves the current trail based on the intuition that
  - the solutions obtained are trustworthy
  - similar solutions should be explored

Selection of the influential solutions

$\tilde{X}^{[l]}$ collects the solutions around which the search will be intensified

- the classical Ant System considers all the solutions of iteration $l-1$
- the elitist methods consider the best known solutions
  - the best solution of iteration $l-1$
  - the best solution of all iterations $< l$

The elitist methods

- find better results in shorter time
- require additional mechanisms to avoid premature convergence

Some variants of the Ant System

- $\mathcal{MAX} - \mathcal{MIN}$ Ant System: imposes on the trail a limited range of values $[\tau_{\min}; \tau_{\max}]$, experimentally tuned
- HyperCube Ant Colony Optimization (HC-ACO): normalizes the trail between 0 and 1
- Ant Colony System: updates the trail on two levels
  - the global update (already seen) modifies it at each iteration ℓ
    The purpose is to intensify the search
  - the local update updates the trail at each application g of the basic heuristic in order to discourage identical choices in the following

$$\tau_A(i,x) := (1-\rho)\,\tau_A(i,x) \qquad \text{for each } i \in x_A^{[l,g]}$$

The purpose is to diversify the search

General scheme of the *Ant System*

Algorithm AntSystem(I)

$x^* := \emptyset$; $f^* := +\infty$; { Best solution found so far }

For $l = 1$ *to* ℓ *do*

  For $g = 1$ *to* h *do*

    $x := A(I, \tau_A)$; { Basic heuristic with random steps and memory }

    $x :=$ Search(x); { Improvement heuristic }

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

    $\tau_A :=$ LocalUpdate(τ_A, x); { Local update of the trail }

  EndFor;

  $\tilde{X}^{[l]} :=$ Update($\tilde{X}^{[l]}, x$);

  $\tau_A :=$ GlobalUpdate($\tau_A, \tilde{X}^{[l]}$); { Global update of the trail }

EndFor;

Return (x^*, f^*);

Convergence to the optimum

Some variants of the Ant System converge to the optimum with probability 1 (Gutjahr, 2002)

The analysis is based on the construction graph

- the trail $\tau_A(i,x)$ is laid down on the arcs $(x, x \cup \{i\})$
- no information from the data is used, that is $\eta_A(i,x) \equiv 1$ (*this strange assumption simplifies the computation, but is not necessary*)
- $\tau^{[l]}$ is the trail function at the beginning of iteration l
- $\gamma^{[l]}$ is the best path on the graph at the end of iteration l,
- $(\tau^{[l]}, \gamma^{[l-1]})$ is the state of a nonhomogeneous Markov process:
  - the probability of each state depends only on the previous iteration
  - the process is nonhomogeneous because the dependency varies with l

The proof concludes that for $\ell \to +\infty$, with probability 1

1. at least one run follows an optimum path in $\mathcal{F}$
2. the trail τ tends to a maximum along one of the optimal paths, to zero on the other arcs

First variant with global convergence

The trail is updated with a variable coefficient of oblivion

$$\tau^{[l]}(i,x) := \begin{cases} \left(1-\rho^{[l-1]}\right)\tau^{[l-1]}(i,x) + \rho^{[l-1]}\dfrac{1}{\left|\gamma^{[l-1]}\right|} & \text{if } (x, x\cup\{i\}) \in \gamma^{[l-1]} \\ \left(1-\rho^{[l-1]}\right)\tau^{[l-1]}(i,x) & \text{otherwise} \end{cases}$$

where $\gamma^{[l-1]}$ is the best path found in the graph up to iteration $l-1$ and $\left|\gamma^{[l-1]}\right|$ is the number of its arcs (*to normalise the trail*)

If the oblivion decreases slowly enough

$$\rho^{[l]} \leq 1 - \frac{\log l}{\log(l+1)} \qquad \text{and} \qquad \sum_{l=0}^{+\infty} \rho^{[l]} = +\infty$$

then with probability 1 the state converges to (τ^*, γ^*), where

- γ^* is an optimal path in the construction graph
- $\tau^*(i,x) = \dfrac{1}{|\gamma^*|}$ for $(x, x\cup\{i\}) \in \gamma^*$, 0 otherwise

Second variant with global convergence

Alternatively, if the oblivion ρ remains constant,
but the trail is forced a slowly decreasing minimum threshold

$$\tau(i,x) \geq \frac{c_l}{\log(l+1)} \quad \text{and} \quad \lim_{l \to +\infty} c_l \in (0;1)$$

then with probability 1 the state converges to (τ^*, γ^*)

Here the oblivion is restricted by the minimum threshold

In pratice, all algorithms proposed so far in the literature

- associate the trail to groups of arcs $(x, x \cup \{i\})$ (e.g., to element i)
- use constant values for parameters ρ and $\tau_{\min}$

therefore do not guarantee convergence

The trail τ, and therefore π, can tend to zero on every optimal path