An introduction to CarThaGène

A genetic and RH integrated mapping tool

Thomas Schiex

M. Bouchez, P. Chabrier, C. Gaspin

INRA (Toulouse), France

Genetic and RH maps

INRA

Given a dataset (**`dsload`**) of genetic/RH data for a given population/panel on a given set of markers M, a genetic or RH map is defined by:

- a set of markers $N = \{m_1, \dots m_n\} \subseteq M$
- which is ordered (eg. $m_1 < \cdots < m_n$)
- with a distance between each pair of adjacent markers ($d(m_i, m_{i+1})$)

The genetic/RH mapping problem: find a map (order+distances) that best explains the data set.

What is a good map ?

INRA

Non parametric approach: a map that minimizes the number of compulsory crossovers/breaks, maximizes the sum of 2-points LOD…

Parametric approach: given a probabilistic model of the underlying phenomenon, a map that maximizes the probability of the data (likelihood).

Parameters: probability of recombination/breakage between 2 adj. markers θ_{ij} (probability of retention r)

CARTHAGÈNE always uses true multipoint maximum likelihood as the criteria to evaluate a given marker sequence.

Probabilistic models in CARTHAGÈNE

INRA

- Backcross: as in MapMaker. Dedicated EM.
- RIL (sib/self): as in mapMaker. Dedicated EM.
- Intercross: as in MapMaker.
- Phase know outbreds (1:1, 1:2:1, 1:1:1:1 seg. ratio)
- haploid RH: Dedicated EM.
- Diploid RH.

The “Dedicated EM” code can run more than 2 orders of magnitude faster than traditional EM implementation (MapMaker, RHMAP).

Working with multiple populations

INRA

- Consensus mapping (**`dsmergen`**): one map for all populations. Implicit assumption that all populations have the same marker ordering and distances. Can be used only for similar population types (eg. backcross with outbreds)
- Simultaneous mapping (**`dsmergor`**): one map per population. The only assumption is that all populations shares the same markers ordering. Can be used to merge eg. genetic and RH data.

Taking into account extra information CARTHAGÈNE

Some information on the supposedly "true" marker ordering can be injected in CARTHAGÈNE using the **`constraints`** dataset type.

Composed of triplets of markers m_i, m_j, m_k plus a loglikelihood penalty p.

Semantics: maps such that m_j is not in between m_i and m_k have their loglikelihood penalized by p. Only usable with **`dsmergor`**.

Computing linkage groups

INRA

Same as in MapMaker: given a (Haldane/Ray) distance threshold $\theta_{\max}$ and a LOD threshold $\ell_{\min}$, put in the same group 2 “related” markers *i.e.*, markers that have both:

1. a pairwise distance below $\theta_{\max}$
2. a LOD above $\ell_{\min}$

Catching: 2 unrelated markers can be put in the same group if they are related to “related” markers (**`group`**,**`groupget`**).

2-points information is computed when a data set is loaded (**`dsload`**, **`mrklod2p`**, **`mrkfr2p`**).

Ordering markers and the TSP

Ina group of n markers, there are $fracn!2$ differents orders.

Under strong hypothesis (eg. BC data with no missing), the maximum likelihood marker ordering problem is equivalent to the. . .

Wandering Salesman Problem: given n cities and available routes connecting cities (with distances). Find a path that goes exactly once through each city once and that minimizes the overall distance.

One of the most studied optimization problem in computer science. Know to be potentially very hard (NP-hard).

Ordering markers: use TSP link

- **exhaustive search**: not possible for $n > 8$.
- **building heuristics**: build a “clever” map using 2-point information.
- **improving heuristics**: improve an existing map using a systematic simple mechanism.
- **stochastic search**: improve an existing map by stochastic/clever perturbations.

All “improving” methods can be used as map checking methods (good = cannot be improved).

Good maps & the Heap

INRA

A good map is not only a max. likelihood map. It is a reliable map (such that no alternative order has comparable likelihood).

As far as the set of "active markers" (**`mrkselset`**, **`mrkselget`**) is unchanged, CarThAGène always remember the k best maps explored by any of the ordering process.

The set of these k best maps is called "The Heap" and is central in all the mapping process in CarThAGène.

The Heap

INRA

- is browsable (**`heaprints`**, **`heaprintds`**, **`heapget`**. . .)
- its size k is user adjustable (must be a power of 2), can be huge (eg. 2048) w/o bad cpu-intensive side-effects (**`heapsize`**).
- the current best map of the heap is the implicit target of all “order improving” methods (**`heapget 1`**).

After sufficient search, the heap provides information on the map landscape around the best map (and therefore on its support).

Looking to the heap

INRA

`heapprinto` n *comp blank*:
For each map in the heap, compares the sequence of the markers with the best sequence.

- if $n > 0$: only markers that moved are visualized and contiguous segments of more than n markers that moved are put in brackets.
- if *comp* is set, the output is unaligned. This is useful when the maps include a large number of markers.
- if *blank* is set, the segments which have been moved and whose length is $> n$ are only represented by their extremities.

Heuristics building method

INRA

First: one must build an initial map.

- by hand: specify a marker ordering and ask for max. likelihood estimation. **`markselset`** + **`sem`** : single EM.
- using a TSP heuristics (Nearest Neighbor).

  `nicemapl`: uses 2-points LOD (strongest LOD with the last inserted marker is inserted).

  `nicemapd`: uses 2-pt distances (markser closest to the last inserted marker is inserted).

Warning: 2-points distances/LOD may be undefined/null when merging data-sets.

Heuristics building method

INRA

`build` nb:

1. Start from the nb pairs of markers having maximum loglike.
2. In each of the nb maps, insert the marker with the strongest mean LOD in all possible positions.
3. Select the nb best maps and repeat to 2.

Old automatic building procedure. Does comprehensive mapping (always inserts all markers). Now largely superseded by **`buildfw`**.

Framework building method

INRA

Builds reliable but incomplete maps.

buildfw $\Delta_{\min}$ Δ_{keep} S c $(S = \{\}, c = 0)$

1. Start from all possible pairs of markers.
2. For all available maps, a new marker is inserted in all possible positions. The marker "reliability" is defined as the difference δ in loglike between the best and the second best insertion position. A marker can be inserted only if this difference is larger than $\Delta_{\min}$.
3. From all these new maps, keep only those such that $\delta \geq \Delta_{keep}$.
4. repeat to 2.

Framework building method II

INRA

The process stops when no marker with sufficient quality exists.

- S: a marker ordering to start from (rather than all pairs). Used to extend an existing “reliable” map.
- $c = 1$: when no marker with sufficient quality exists, tries to independently insert all remaining markers in all possible intervals.
  For each such marker: reports the best insertion position (**+**) and how far in loglike all other positions are (support for the best position).

The c flag allows to do framework mapping followed by a comprehensive mapping of all remaining markers wrt. the framework.

Heuristics improving methods

INRA

Start from a non empty heap (best taken). All maps considered are candidate for the heap.

- **`flips`** $w\ \Delta_{\max}$ *Iter*: tests all maps obtained by all possible permutations inside a sliding window of size w.
  Reports all permutations that lead to a loglike. within $\Delta_{\max}$ of the best loglike.
  If an improved map is found and if *Iter* is set to 1, the process is reiterated on the new best map.
- **`polish`**: each marker in the map is tested in all possible intervals. Reports the matrix of the Δ in loglike.

Stochastic search: general principles

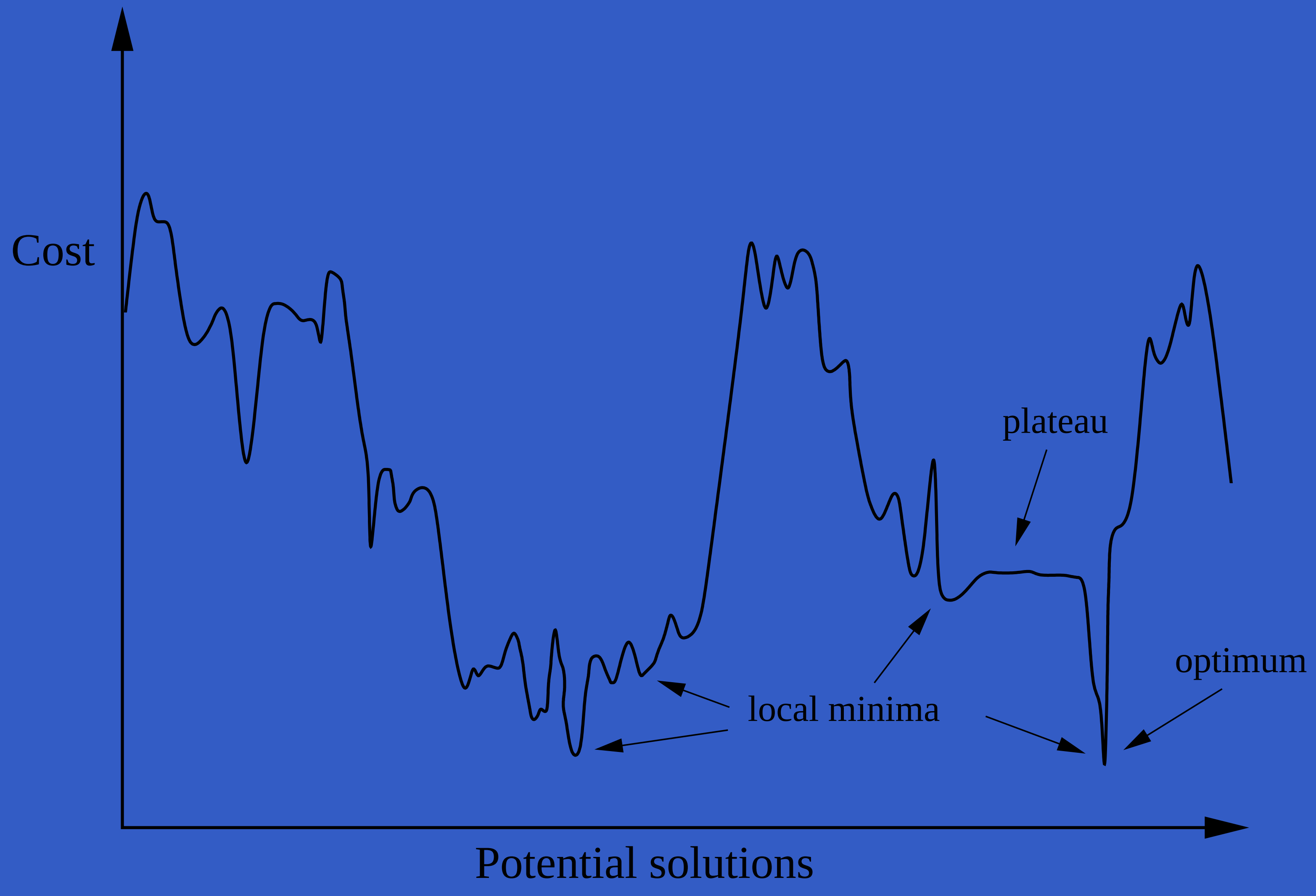
INRA

1. We start from the best order available in the heap.
2. We perturbate this order to get a new order (called a neighbor). The neighbor chosen may be chosen randomly or ”smartly”. The loglike may increase or decrease.
3. depending on some tests (which may include stochasticity) we either “move” to this new order or stay were we are.
4. we repeat from 2.

The whole process may be repeated several time. The possible perturbations (the neighborhood) is crucial: use known TSP neighborhood.

Stochastic search: general principles

INRA

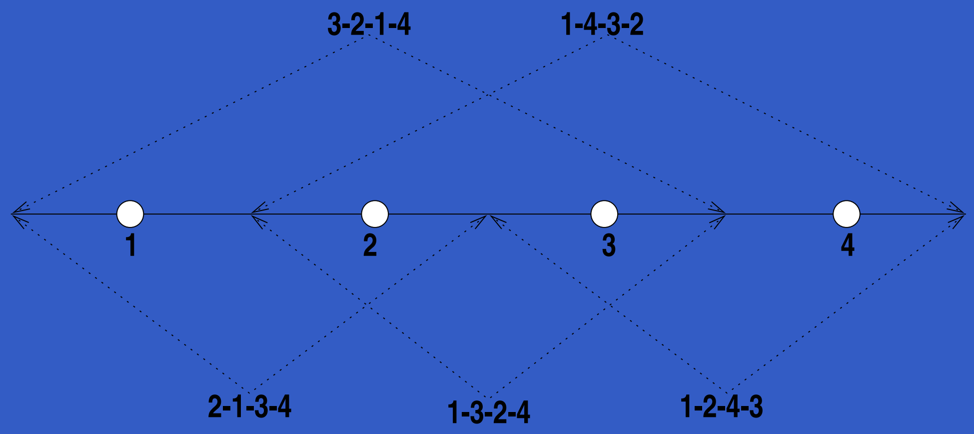


Neighborhood

INRA

Famous 2-OPT and 3-OPT neighborhoods for the TSP adpated for the WSP.

- 2-OPT: choose 2 markers and invert the delimited submap.
- 3-OPT: choose 3 markers and swap the two delimited submaps.



Simulated annealing

INRA

Exploits an analogy with metallurgy/thermodynamics.

- a state of the system – a map m
- the energy of the state – the opposite of the loglike of the map

The probability of accepting a state of increasing energy is determined by the Boltzmann's distribution. We start at an initial temperature T_i from an initial map (state) m with an energy (opposite of loglike) $-\ell(m)$ and slowly cool down the system while perturbating it til it reaches $T_{\min}$.

Simulated annealing

INRA

$m \leftarrow$ initial map;
$T \leftarrow T_i$;
while $T > T_{min}$ **do**
    for $m = 1$ **à** $m = NbTries$ **do**
        $x' \leftarrow$ RandomNeighbor(x);
        $\delta \leftarrow \ell(m) - \ell(m')$;
        if $(\delta \leq 0)$ **then** $m \leftarrow m'$;
        else if *random*$(0, 1) \leq e^{\frac{-\delta}{T}}$ **then**
            $m \leftarrow m'$;
    $T \leftarrow T.\alpha$;
return $(m, \ell(m))$;

Simulated annealing parameters

INRA

annealing *NbTries* T_i $T_{\min}$ α

1. T_i can be chosen arbitrarily. It si automatically adjusted. I usually use 100.
2. $LPlateau$ should be larger than $\frac{n.n-1}{2}$
3. α is close to 1. This fixes the length of the search (fast/slow cooling)
4. $T_{\min}$ should be small enough to avoid a premature end.

You must play with the parameters α and $T_{\min}$. There is no clear methodology to set them up.

Taboo search (`greedy`)

Starting from the best map in the heap and for a given number of steps:

1. Move to the best 2-OPT neighbor
2. unless this move is **taboo** (has been used recently)
3. one can nevertheless violate the taboo if the move improves over the best known solution.

This can be repeated several (*NbLoop*) times.

Taboo search (`greedy`)

INRA

`greedy` *NbLoop NbExtra TabooMin TabooMax*

- The taboo search is repeated *NbLoop* times (try 1 first)
- The number of moves is autoadjusted but you can give extra moves using *NbExtra*. Default: use 0.
- A move is taboo if it has been recently used. The definition of “recently” varies stochastically during search between *TabooMin TabooMax*. These are percentages. Typically try 1 and 20 but your milleage may vary.

Final points

INRA

- Not all the commands are accessible under the graphical interface.
- But, you may type them in the shell available in graphical interface
- The shell level includes a complete simple programming langage (Tcl). You can completely automate your mapping strategy.