Problem Decomposition in Metaheuristics

Éric Taillard

HEIG-VD, University of Applied Sciences of Western Switzerland

Summary of the lecture

Introduction

Metaheuristic components

Few combinatorial problems interesting for decomposition methods

p-median, map labelling, vehicle routing, location-routing

Small and large instances

Building a large initial solution

Delaunay triangulation

p-median with capacity

Improving large solutions

LNS

POPMUSIC

METAHEURISTIC COMPONENTS

Problem modelling, objective and utility functions

Mono-optimization, Multiobjective optimization, Classification

Constructive methods

Random building

Greedy constructive methods

Local searches

Neighbourhood structure

Neighbourhood limitation (candidate list) and extension (ejection chain)

Decomposition methods

Domain decomposition

Building method

Improvement methods ⇒ LNS, POPMUSIC

Learning mechanisms

Learning to model ⇒ Hyper-heuristics

Learning to build ⇒ GRASP, Artificial ants

Learning to improve ⇒ Tabu Search

Learning with solutions ⇒ Genetic algorithms, particle swarm, path relinking

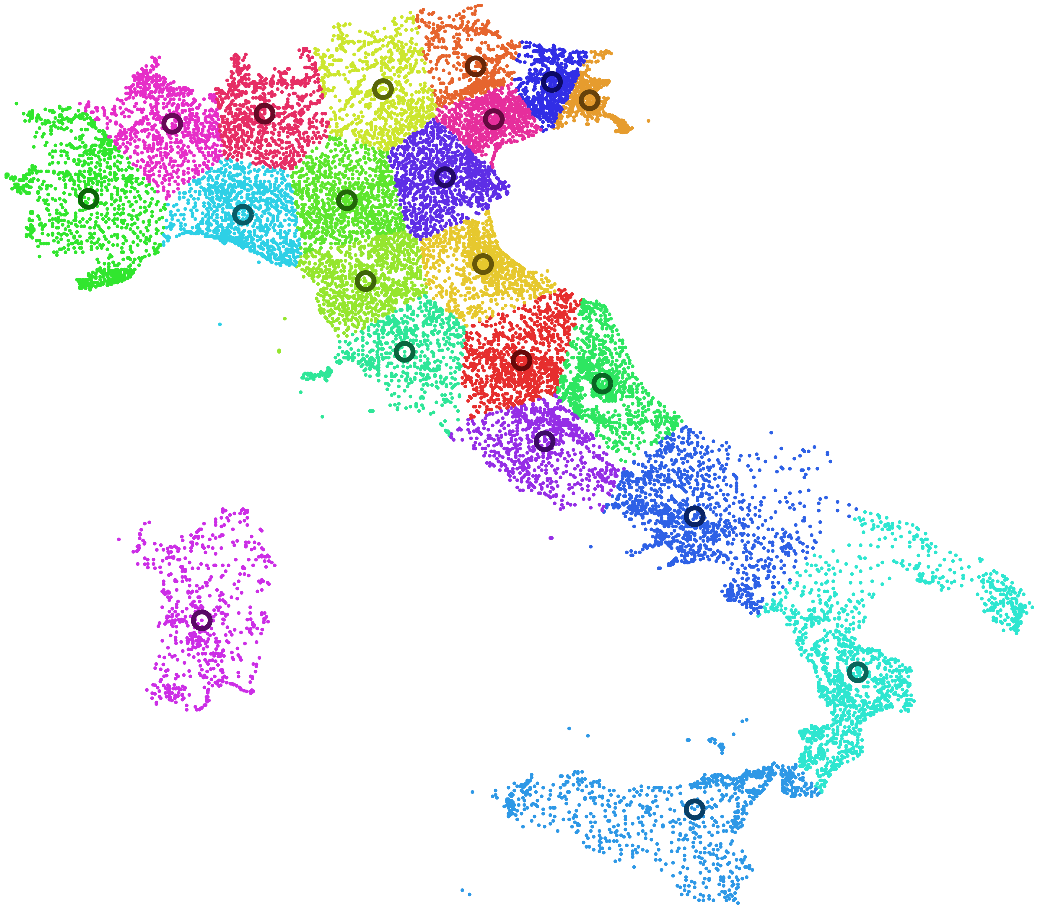
THE P-MEDIAN PROBLEM

Given :

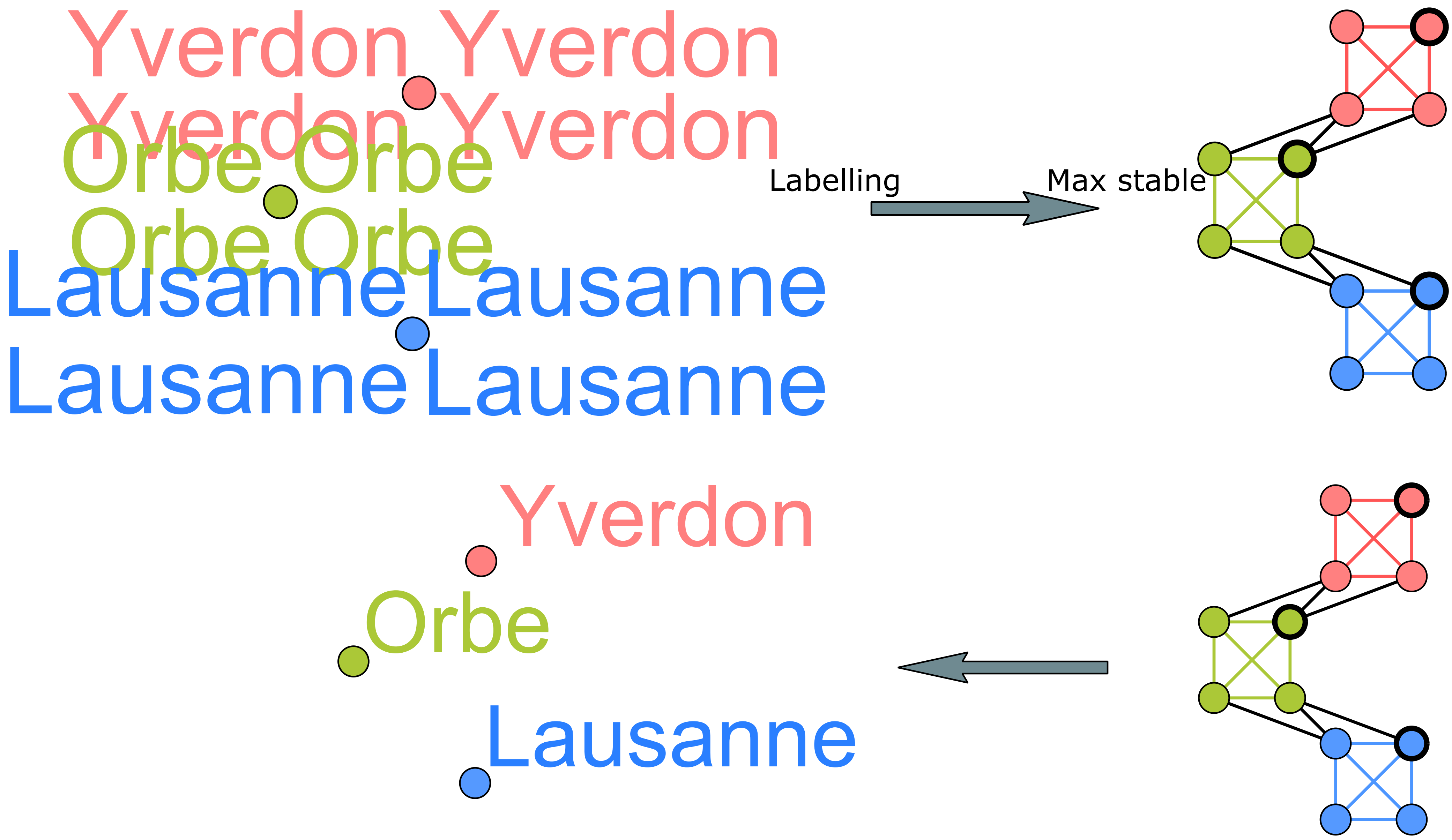
n elements $\in I$ with distance matrix $D = (d_{ij})$ between them

Find :

p central elements $\{c_1, \ldots, c_p\} \in I$ minimizing $\sum_{i=1}^{n} \min_{j=1,\ldots,p}(d_{i,c_j})$



Map labelling



Other problem that can be modelled like this : assigning flight levels and departure times of aeroplanes.

Capacitated vehicle routing problem

Given :

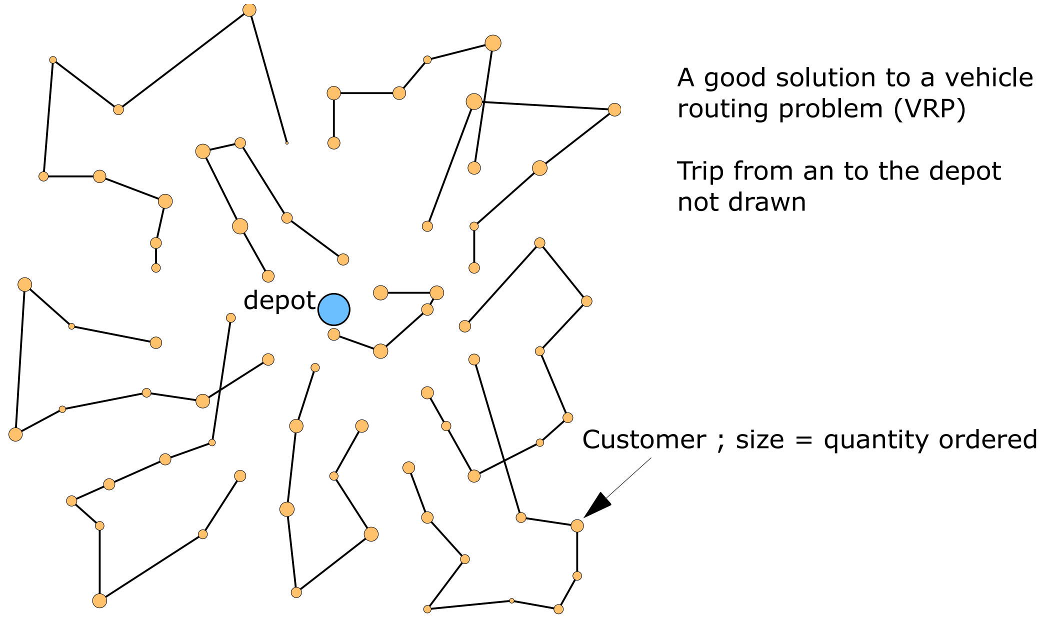
n customers and 1 depot

q_i : quantity ordered by customer i

Distances between each pair of customer

and between depot and customer

Q : vehicle capacity



Find :

Set of tours such that :

Each tour starts from and comes back to the depot

Each customer appears exactly once in the set of tours

The sum of the quantities ordered by the customer on any tour $\leq Q$

+ eventually other constraints on the tour length, time windows, multiple depots, etc.

Objective :

Minimize the total length performed by the vehicle

LOCATION-ROUTING PROBLEM

Given :

n customers

m potential depots locations

q_i : quantity ordered by customer i

Travel costs between each pair of customers and between depots and customers

D : Depot opening cost

Q : vehicle capacity

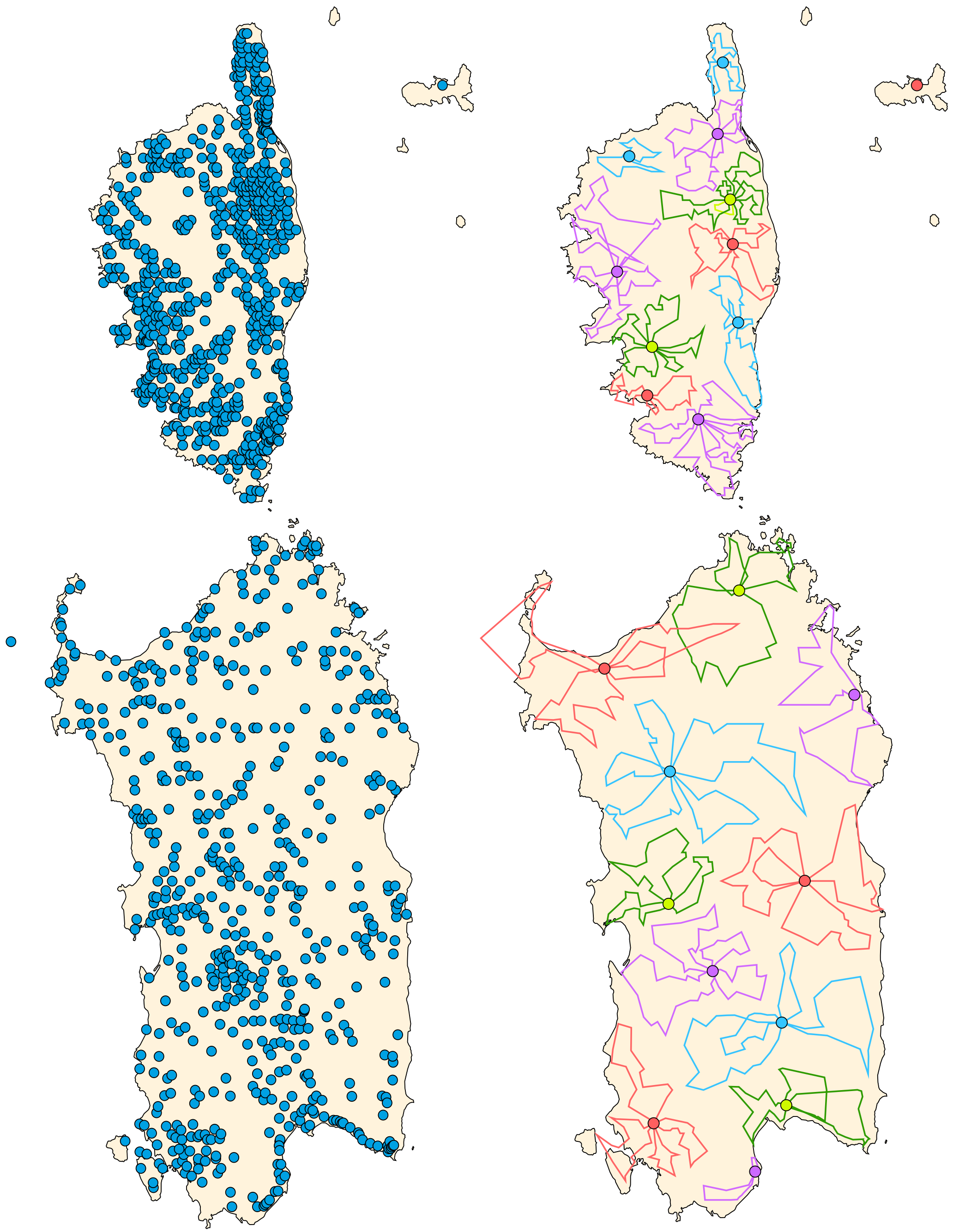
Find :

Subset of depots to open

Set of tours verifying VRP constraints

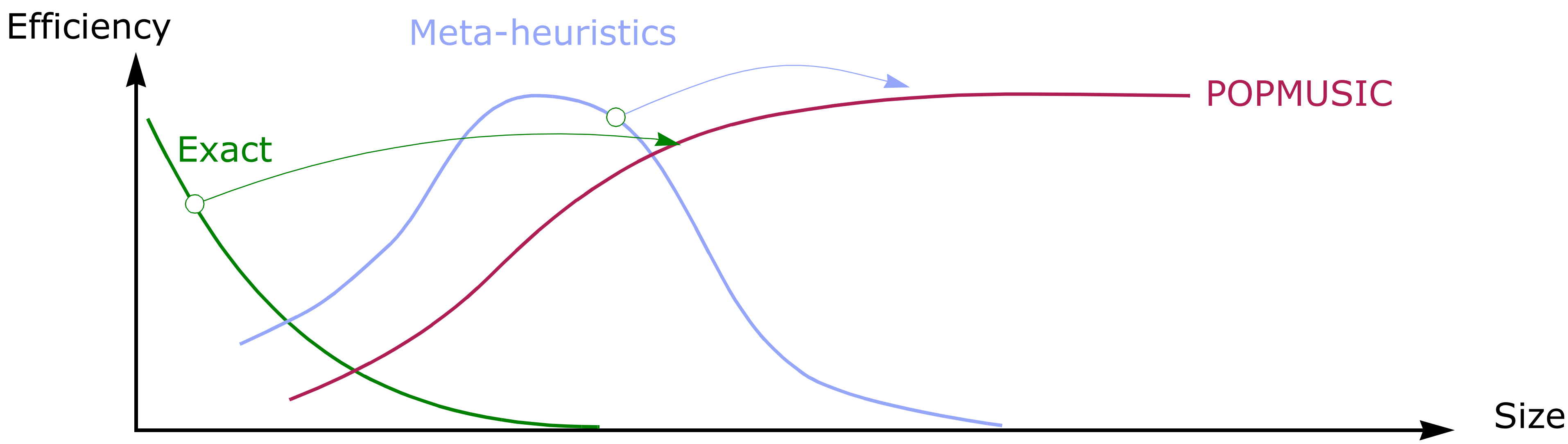
Objective :

Minimize the total costs



CLASSIFICATION OF PROBLEM SIZE

Class	Typical technique	Size (order)	
Toy	Complete enumeration	10^1	
Small	Exact method	10^1—10^2	
Medium	Meta-heuristics	10^2—10^4	Memory limit $O(n^2)$
Large	Decomposition techniques	10^3—10^7	Time limit $O(n^{3/2})$
Very Large	Distributed database	above	



DOMAIN DECOMPOSITION

Part of problem modelling

Difficult to generalize

Frequently used for small problems with several embedded subproblems

Example 1 : Location-routing

Find number and position of depots to open

Solve a multi-depot vehicle routing problem

Example 2 : Location-routing

Find a TSP tour on all customers $\Rightarrow$ Concorde TSP solver

Split the TSP tour into sub-paths with sum of customer demands $\leq Q \Rightarrow$ Dynamic programming

Group the customers of a sub-path into a single super-customer

Find number and position of depot $\Rightarrow$ Uncapacitated warehouse location

Example 3 : Map labelling

Generate several label positions for each object

Build and solve the maximum weight stable set problem

Building an initial solution : Greedy constructive method

Idea

Build a solution, element by element, by adding systematically the most appropriate element

This works optimally for a number of problems (optimum spanning trees, matroids)

A solution is composed of elements $e \in E$

TSP, Steiner tree :	Edges
	Nodes
Colouring :	Vertex with a given colour
	Edge orientation
QAP :	Element at a given position

The method starts with a solution *s* empty or trivial

An incremental cost function $c(e, s)$ that measures (empirically) the quality of adding element *e* on partial solution *s* must be defined

Adding an element generally implies restrictions for the next elements to add

```
s = minimal partial solution                         // Generally : ∅
R = E                          // Set of elements that can be added to s
Repeat
   Evaluate c(s, e) for each e ∈ R
   Choose e′ optimizing c(s, e)
   Set s = s ∪ {e′}
   Remove from R all elements that cannot be added to s any more
Until s is a complete solution
```

Example for the TSP : Nearest neighbour heuristic

$s = \{1\}$

R : set of cities not yet visited (+ city 1, if all cities already visited)

$c(s, e)$ = length of the edge going from the last city of s to city e

Complexity analysis :

Number of main loop **Repeat…Until** : n

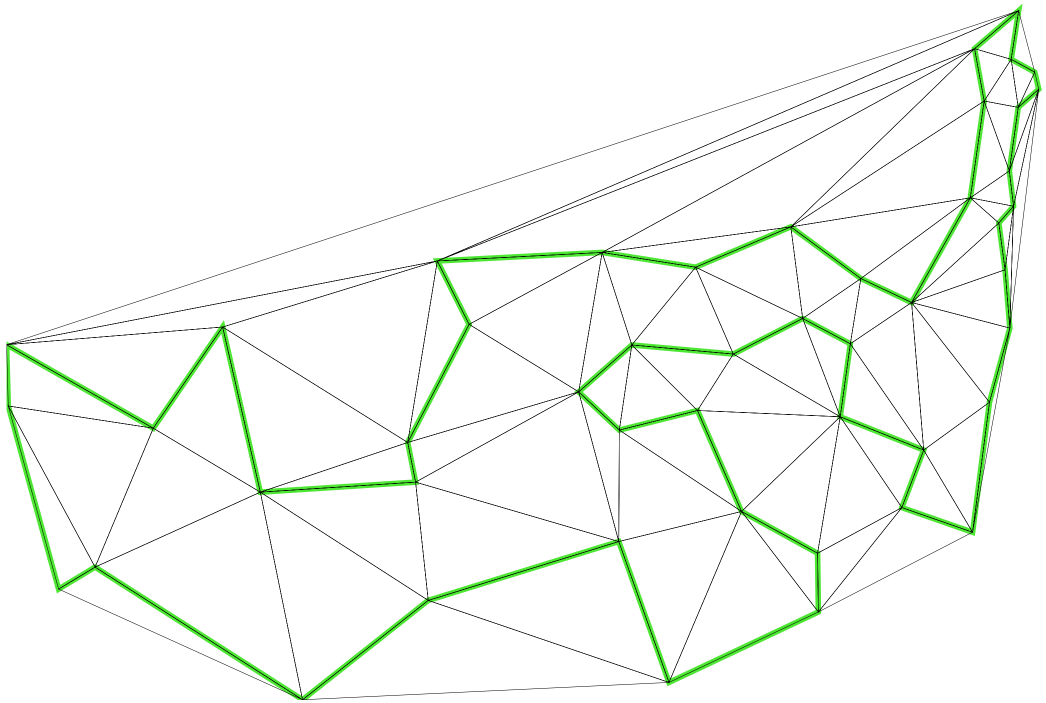
Number of evaluation of $c(s, e)$ at k^{th} iteration : $n - k$

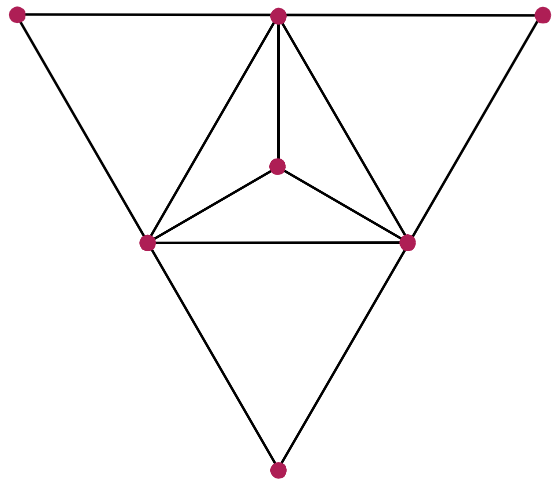
Total : $O(n^2)$

n log n Greedy constructive method for Euclidean TSP

Build a Delaunay triangulation ⇒ $O(n \log n)$ algorithm, ~$6n$ remaining edges

Build a TSP tour on the Delaunay





Non Hamiltonian Delaunay

⇒ **Best insertion**

s = Tour on 2 cities (e.g. 1—2—1, closest cities, most distant cities)

R : Set of cities not yet visited

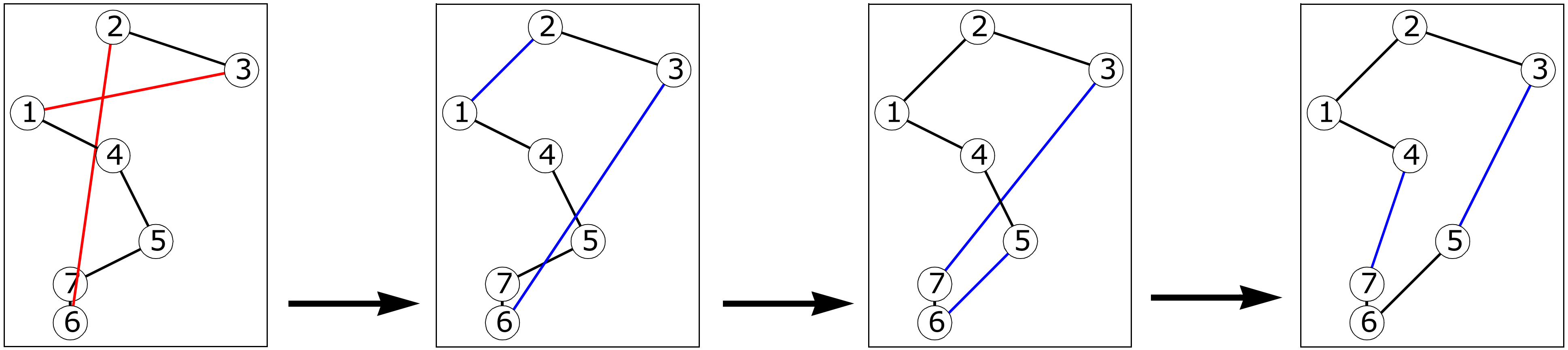
Next city e to insert: randomly chosen in R

$c(s, e)$ = Minimum insertion cost of city e between 2 cities of partial tour s

Choose the smallest $c(s, e)$ ⇒ $O(\log n)$ if done while building the Delaunay

```
Start with a given solution (obtained e.g. with a constructive method)
    Repeat
        Try to find a modification that improves the solution
        If such a modification is found, perform it
    While An improvement is found
```

Example for the TSP



1st improvement 2nd improvement 3rd improvement

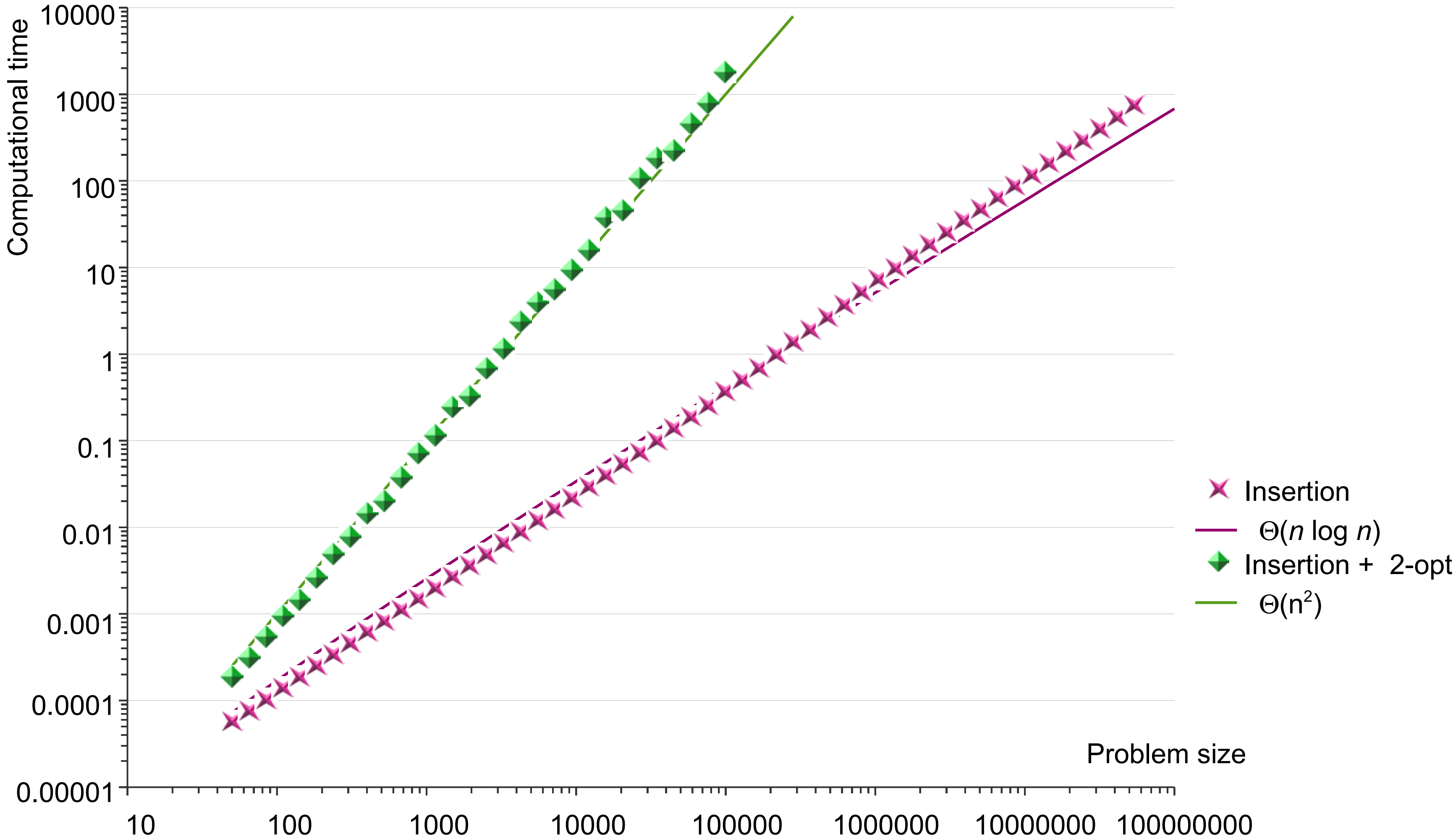
Example of modification :

Replace 2 edges of the tour by 2 others (2-opt *neighbourhood*)

Neighbourhood size

$O(n^2)$

Experiments on TSP



Questions :

How to generate a solution for non Euclidean problems in less than $O(n^2)$?

How improve a solution in less than $O(n^2)$?

Building a solution to large instances via problem decomposition

Hypothesis

Large problem instances but moderate dimension

$\Rightarrow$ 2 elements close to a third one are also close

$\Rightarrow$ 2 elements far away cannot be both close to a third one

Distant elements are not directly connected together in reasonable solutions

Reasonable solutions are composed of sets with about *C* elements (independent of problem size)

Example : The number of letters a postman can deliver in a day does not depend on the total number of people living in the country

TEMPLATE FOR PROBLEM DECOMPOSITION

Input

n elements, function $d(i, j)$ measuring the proximity between elements i and j

Body

1 Create a random sample E of $20\sqrt{n}$ elements

2 Solve a relaxation of a p-median with capacity with $p = \sqrt{n}$ on E

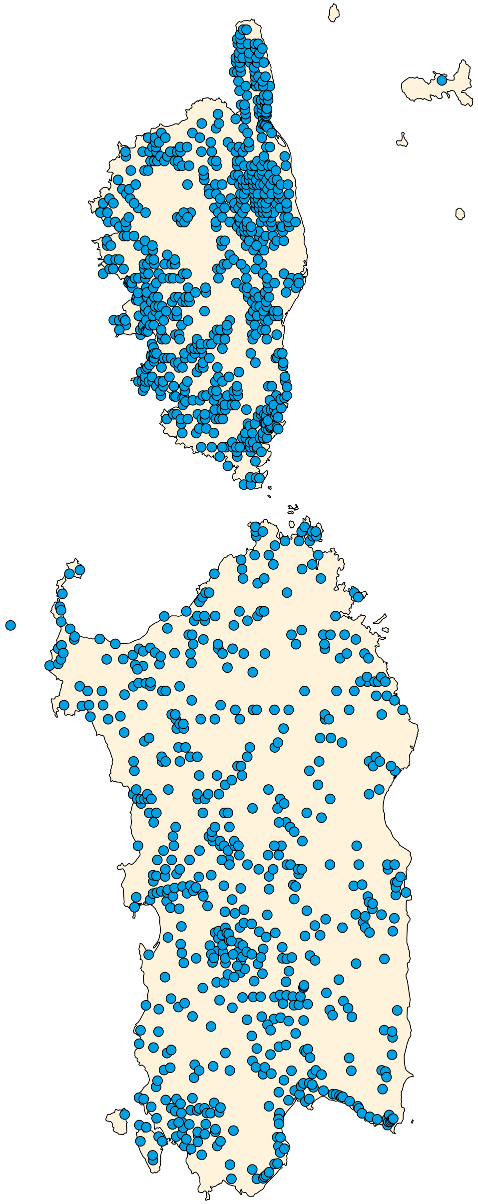
3 Assign each of the n elements to its closest among the p centres
$\Rightarrow \sqrt{n}$ clusters with $\sim\sqrt{n}$ elements each

4 Build a proximity graph G on the centres
$\Rightarrow c_i$ and c_j are neighbours if:
there is an element assigned to c_i which second closest centre is c_j

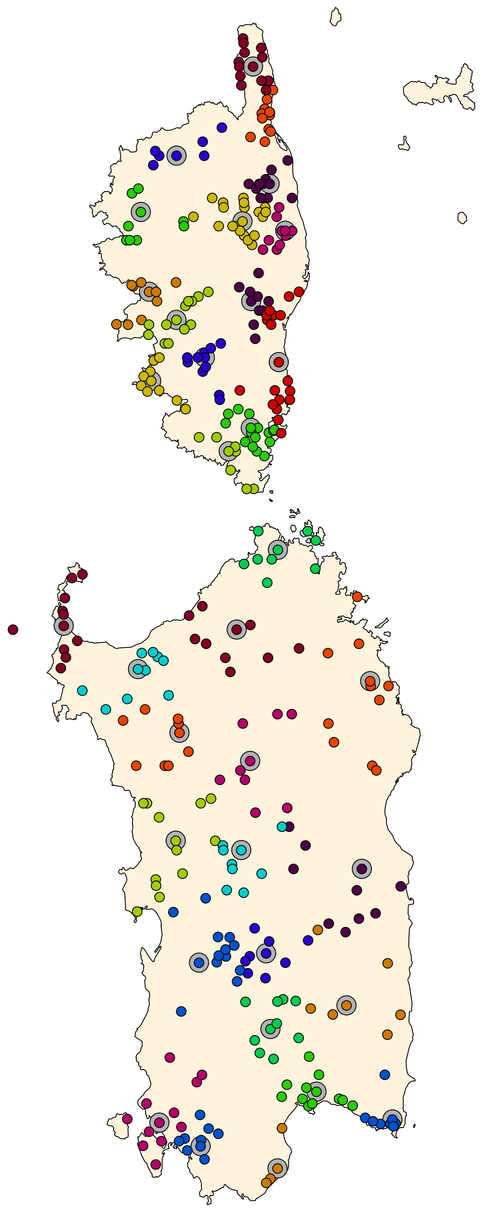
Output

$\sim\sqrt{n}$ clusters, proximity graph G

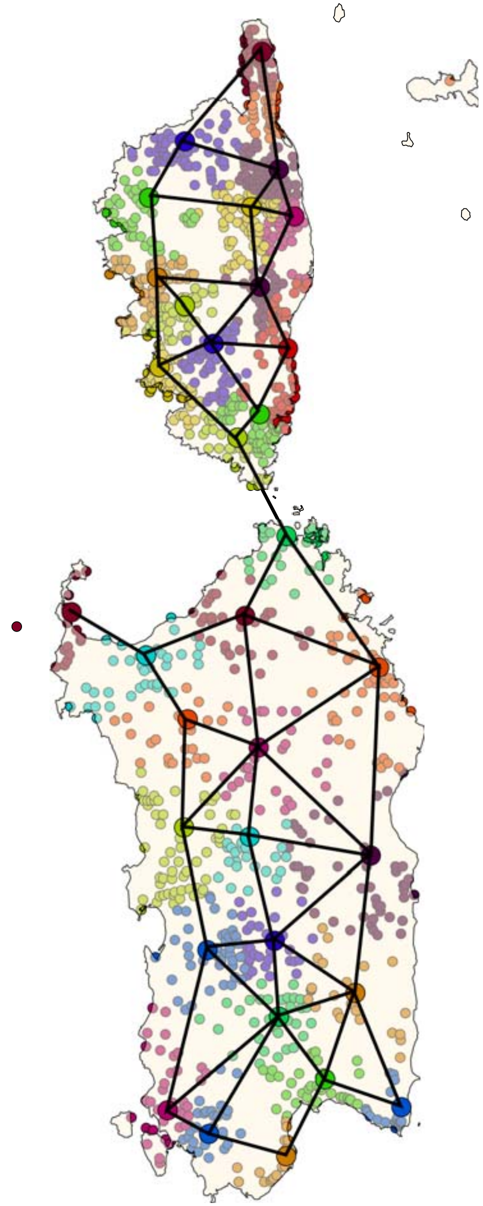
PROBLEM DECOMPOSITION



Initial set of elements



Sample + clustering



Assignement +
proximity graph

Goal :

Decomposing a set $E = \{1, \ldots, n\}$ into p clusters $C_1, \ldots, C_p$ with $\sim n/p$ elements each

Notation

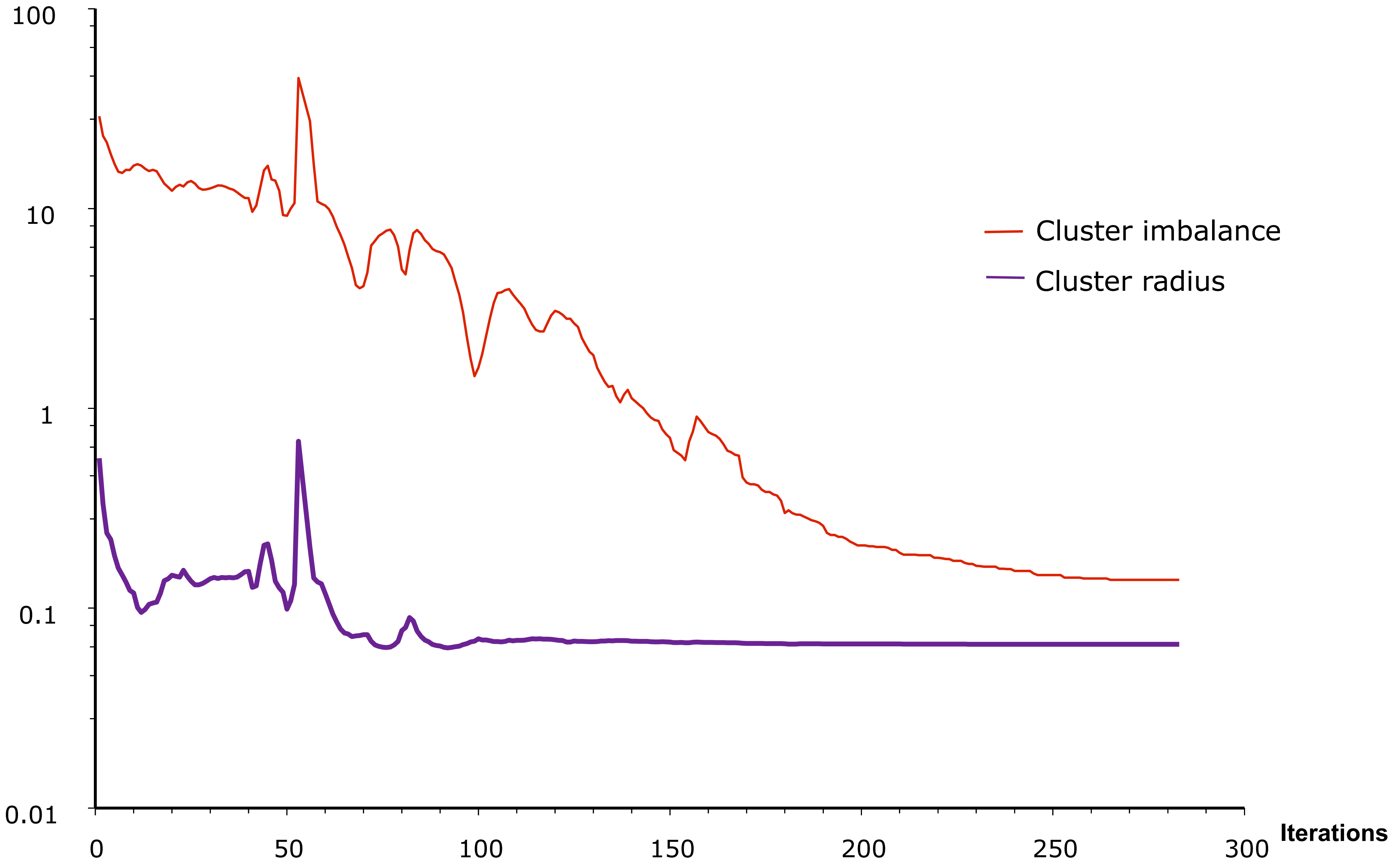
λ_j: penalty of centre j

Randomly choose p centres $c_1, \ldots, c_p$ among the n entities
$f = 1.0$; $\lambda_j = 0$; $j = 1, \ldots, p$;
for 234 iterations **do**
    Allocate entities of E to the closest centre with penalty
    (Create clusters $C_1, \ldots, C_p$)
    for $j = 1, \ldots, p$ **do**
        Find the best position of centre c_j among entities of C_j
    if Current solution improves best known then
        Memorize current solution as best known
    $f \leftarrow 0.98 \cdot f$
    for $j = 1, \ldots, p$ **do**
        $\lambda_j \leftarrow \lambda_j + f \cdot (\text{average distance of elements to centre}) \cdot (p \cdot |C_j|/n - 1)$

Complexity

$\Theta(n \cdot p + n^2/p) \Rightarrow \Theta(n^{3/2})$ if p in $\Theta(\sqrt{n})$

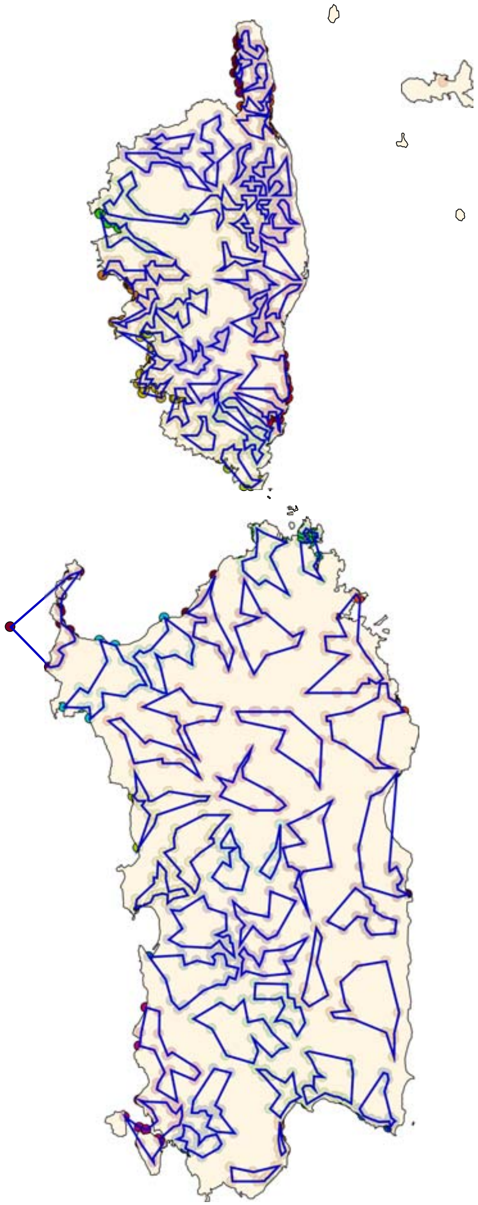
Evolution of sum of distances and size variation of clusters

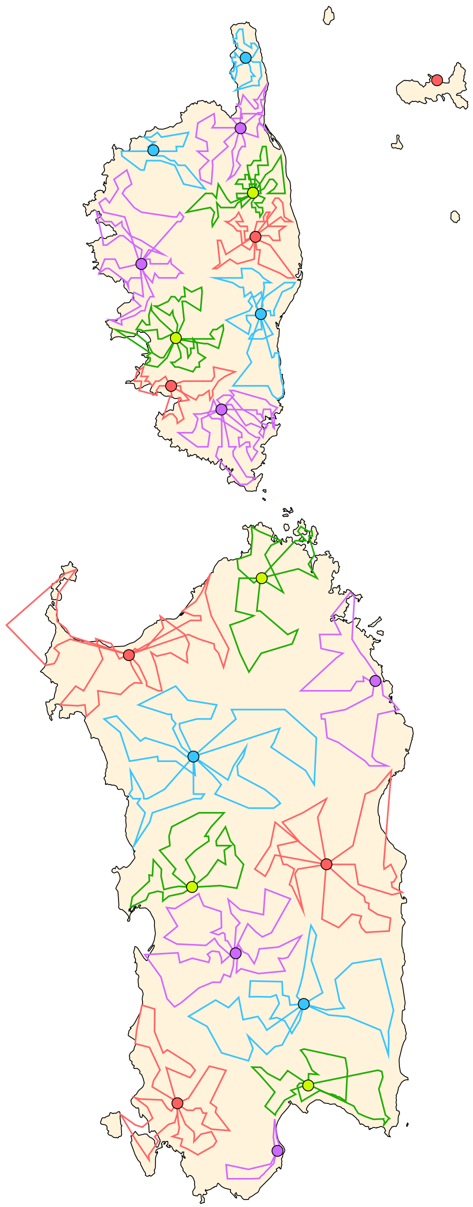


Application to Location-Routing

Decomposition of clusters into smaller clusters that satisfy +/– vehicle capacity

Building independent vehicle tours

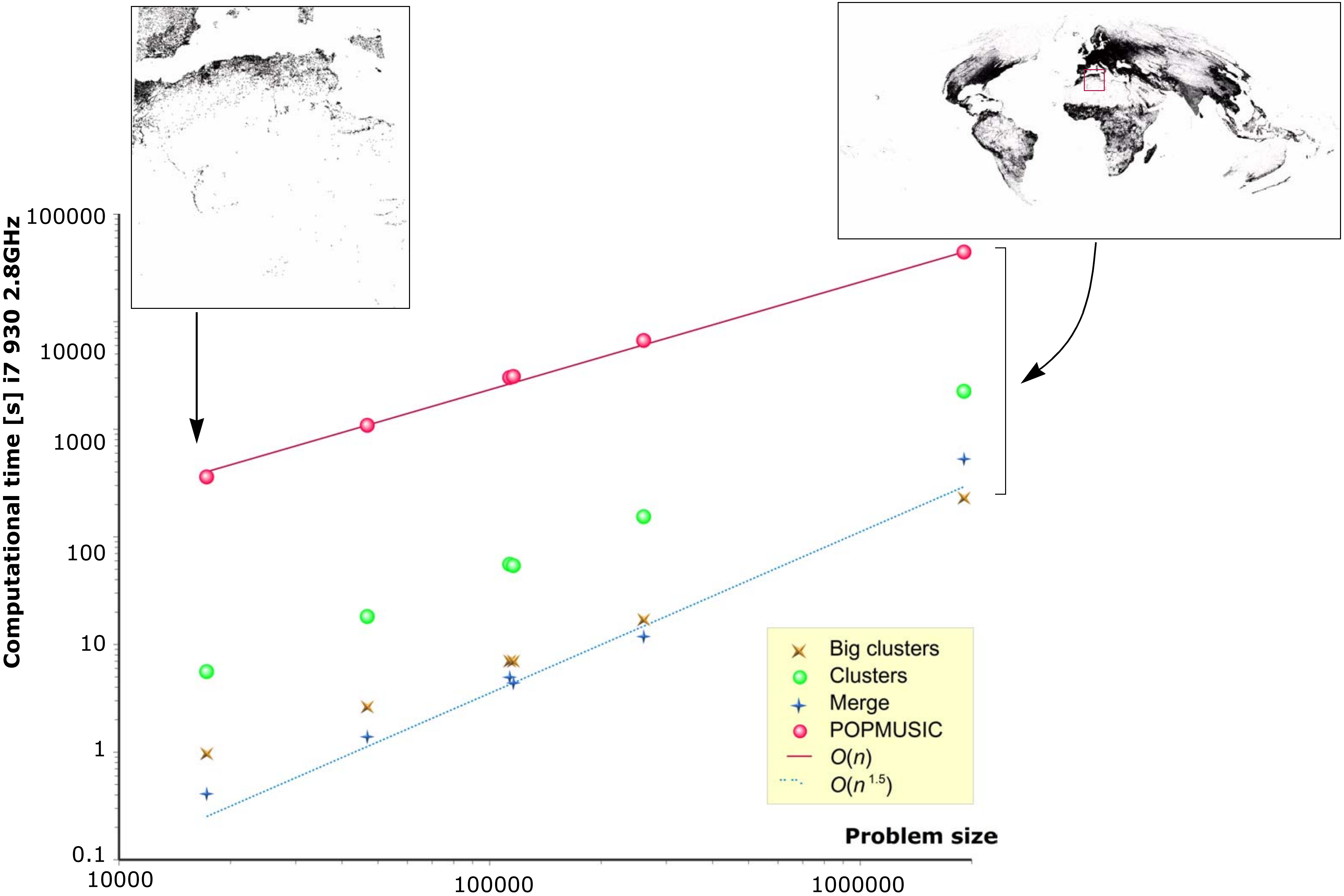




Finding depot location

Connexion of TSP tours on depots

LOCATION-ROUTING : EMPIRICAL ALGORITHMIC COMPLEXITY



EXERCISE

You have an $O(1)$ « distance » function $d(i, j)$ for computing the distance between cities i and j

You have an $O(n^2)$ greedy procedure ***G*** for building a TSP tour

You have an $O(n^{3/2})$ procedure ***P*** for decomposing a set of n entities into $\sqrt{n}$ clusters of $\sim\sqrt{n}$ entities

How to get a TSP solution in $O(n^{3/2})$

SOLUTION IMPROVEMENT

OF LARGE INSTANCES

Large neighbourhood search (LNS)

Popmusic : a generic decomposition technique

Applications

- Clustering
- VRP, location-routing
- Cartographic labelling

LARGE NEIGHBOURHOOD SEARCH (LNS)

Idea

- In an enumeration method for integer or mixed integer linear programming
  - Fix the value of a subset (a majority) of variables
  - Solve optimally the sub-problem on the remaining variables
  - Repeat with other subsets of fixed variables

Evolution

- Destroy a portion (free variables) of the solution
- Try to rebuild the solution by keeping fixed variables
- Repeat with other portions

- Iterated local search
  - Randomly perturb the best solution known
  - Apply an improving method with penalties
  - Repeat after having modified the penalties

LNS for the VRP (Shaw 1998)

Generate an initial solution

Destroy mechanism

Select a random customer

and few close customers

"close" : Euclidean distance + random component

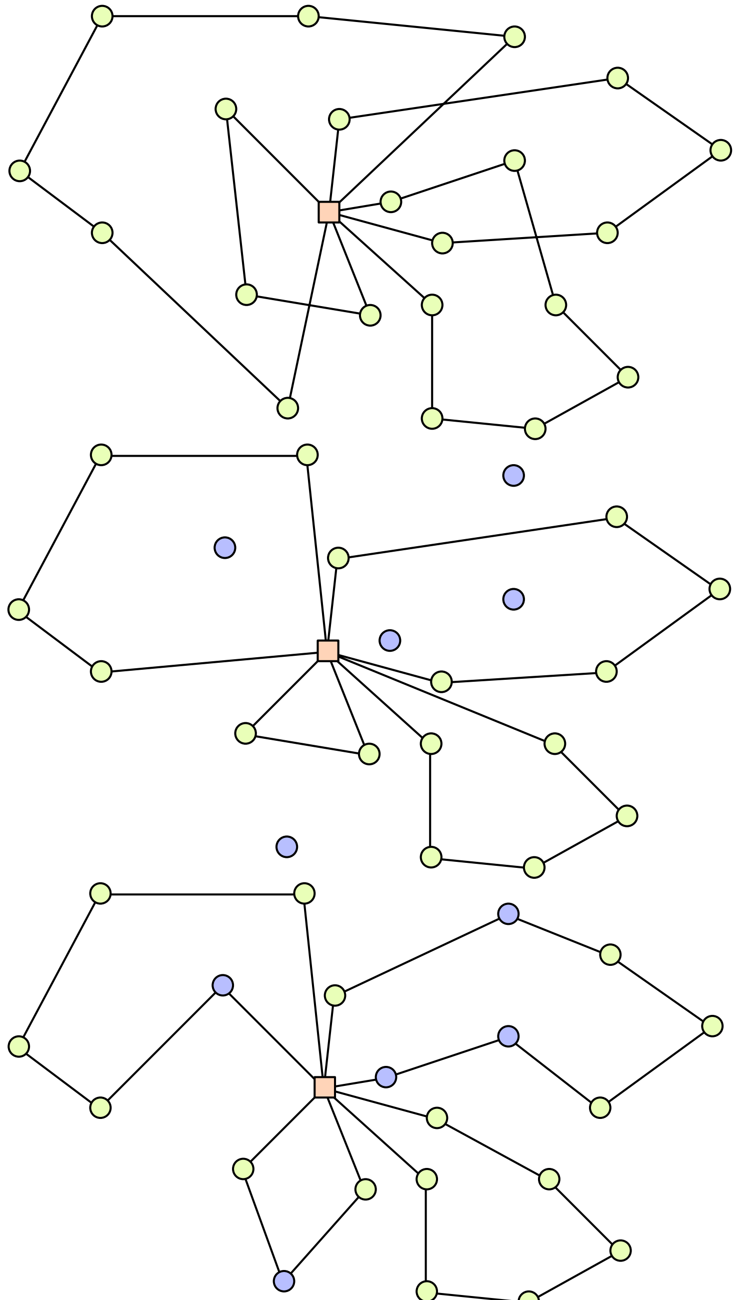
Repair method

Optimal or heuristic re-insertion (with constraint programming)

⇒ Applied to small-medium problem instances only

⇒ No preoccupation on algorithmic complexity

⇒ Destroy + repair = reoptimize a portion of the solution



POPMUSIC GENERAL IDEA

Start from an initial solution

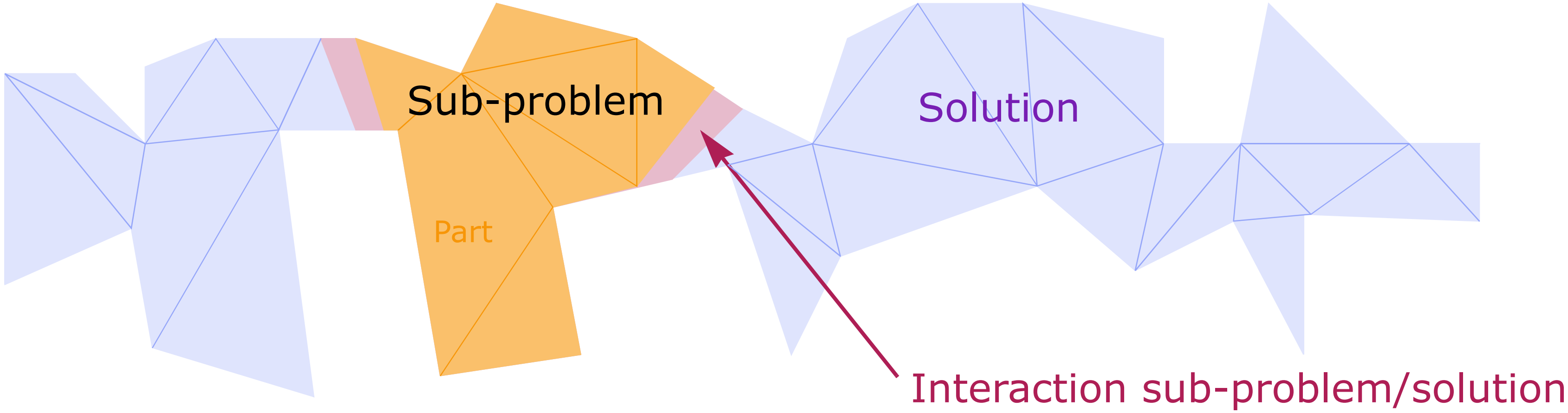
Decompose solution into parts

Optimize a portion (several parts) of the solution

Repeat, until the optimized portions cover the entire solution

Difficulty

Sub-problems are not necessarily completely independent one another



Part :

Elements belonging to a cluster

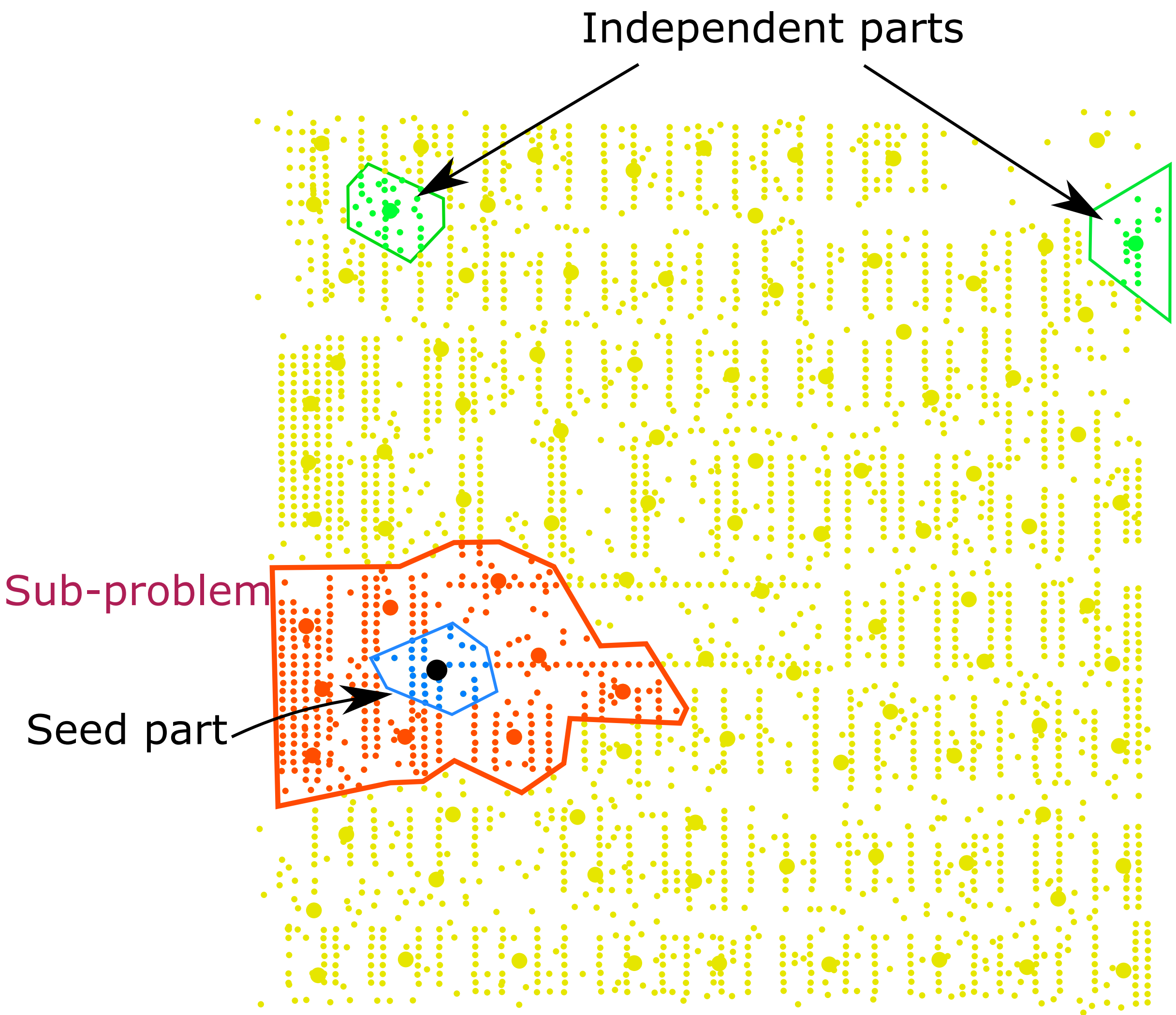
Distance :

Average dissimilarity between elements of different groups,

Distance between centres

Optimization process :

Improving method based on candidate list, relocation of a centre, stabilizing solutions (CLS)



Input

Solution $S = s_1 \cup s_2 \cup \ldots \cup s_p$ // p disjoint parts

$O = \varnothing$ // Set of " optimized " seed parts

While $O \neq S$, **repeat** // Parts may still be used for creating sub-problems

1. Choose a seed part $s_i \notin O$

   // r : parameter
2. Create a sub-problem R composed of the r " closest " parts $\in S$ from s_i
3. Optimize sub-problem R
4. **If** R improved **then**
   Set $O \leftarrow O \setminus R$
   Else
   Set $O \leftarrow O \cup s_i$

POPMUSIC CHOICES

How to get an initial solution

cf. above

Definition of a part

Distance between two parts

Seed part choice

Random, *O* managed as a stack, …

Parameter *r*

Depends on optimization procedure capability

Optimization procedure

Exact method, matheuristic, metaheuristic

Variants :

Slower :

set $O \leftarrow \varnothing$ instead of set $O \leftarrow O \setminus R$

Faster :

set $O \leftarrow O \cup R$ instead of set $O \leftarrow O \cup s_i$

Related concepts

Candidate list, strongly determined and consistent variables (Glover)

"Chunking" (Woodruff)

Large neighbourhoods (Shaw)

VDNS (Hansen & Mladenovic)

Decomposition methods

POPMUSIC FOR VRP (TAILLARD 1993, …)

Part:

Vehicle tour

Distance between parts:

Polar distance between centres of gravity

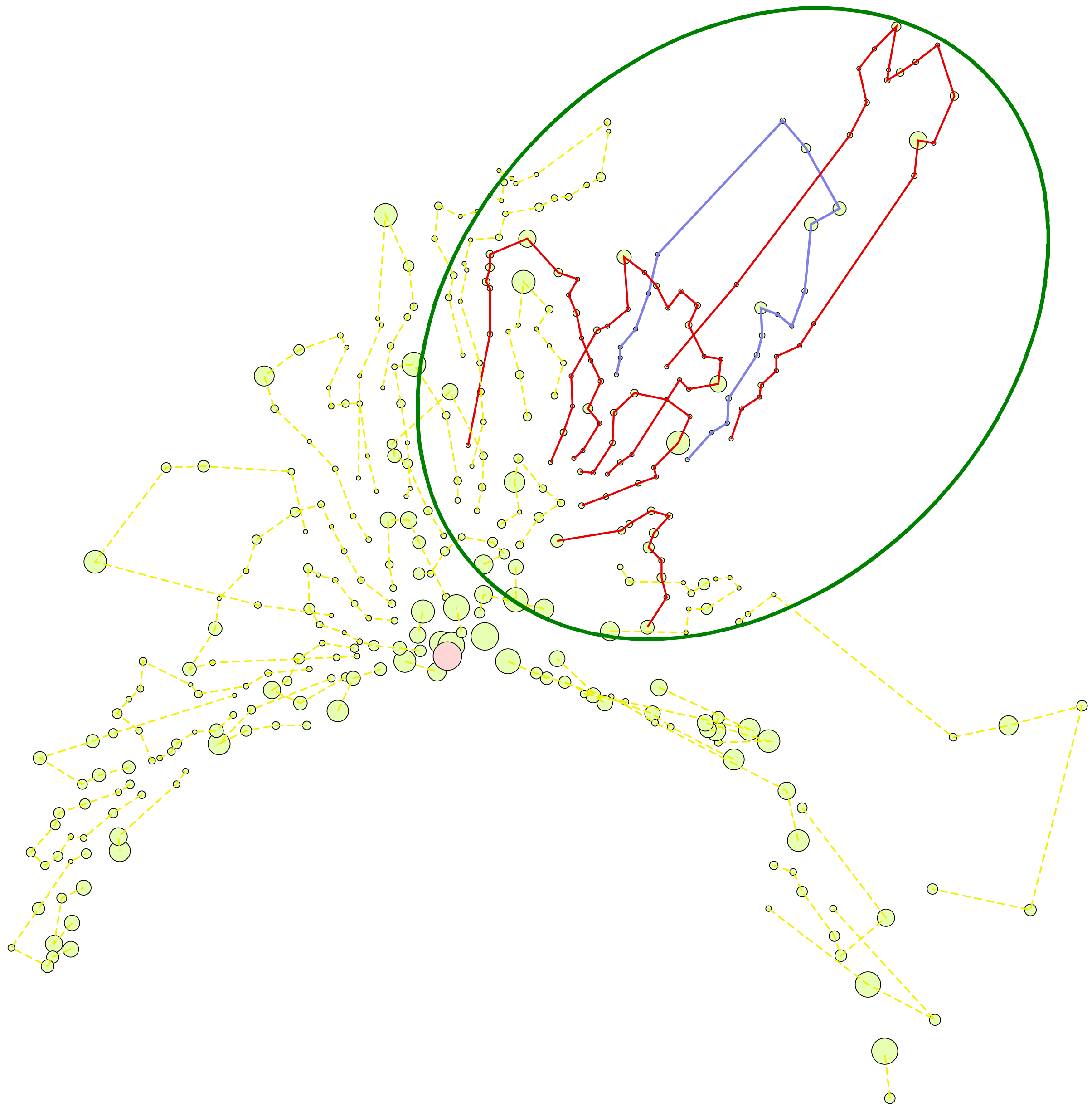
A sub-problem is a smaller VRP

Optimization process:

Basic tabu search

Particularity:

Many simultaneous optimization processes, treating all tours at each iterations



POPMUSIC for Location-Routing (Alvim & Taillard 2012)

Part:

Vehicle tour

Distance between parts:

Minimal distance between customers of different tours

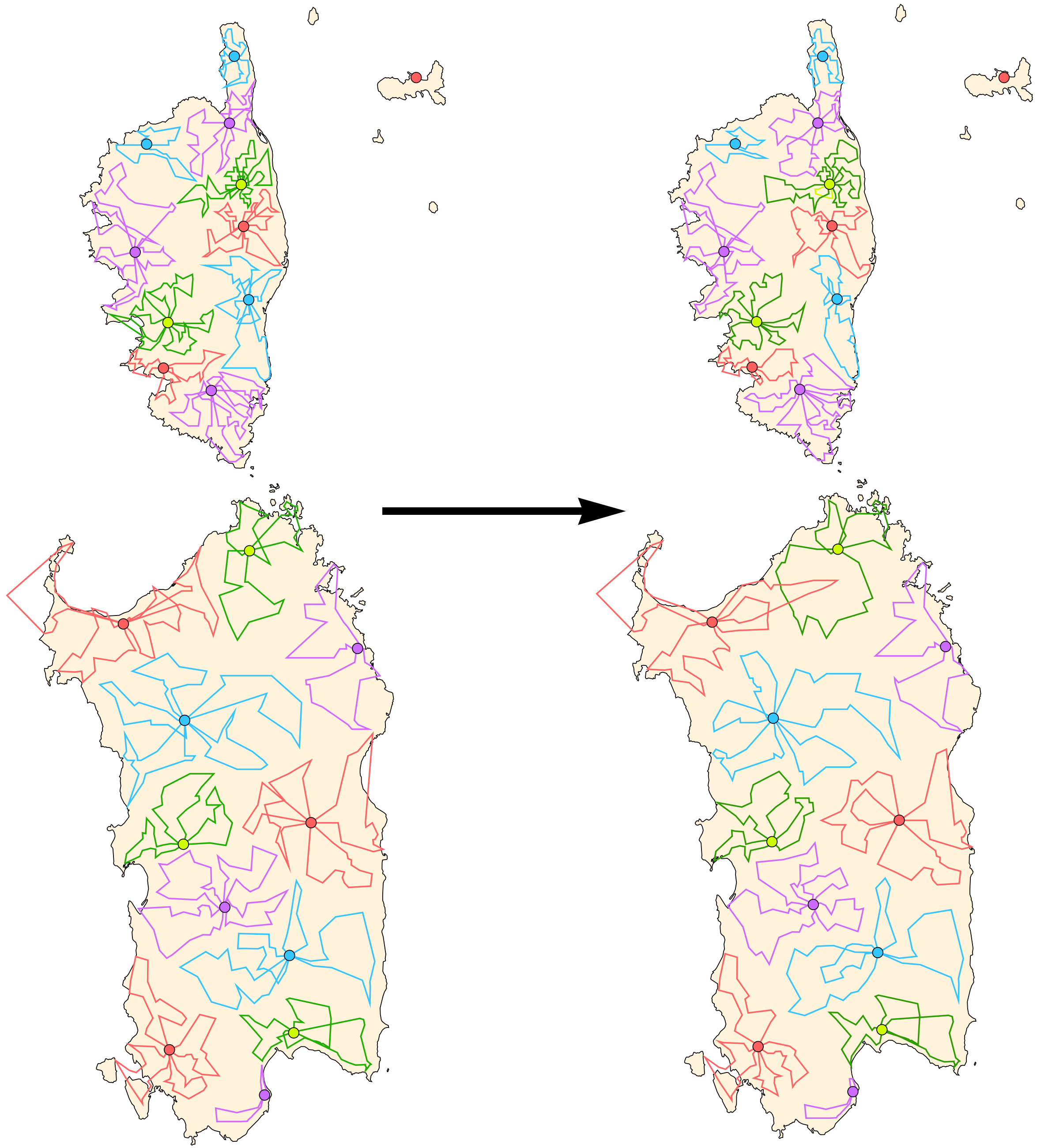
A sub-problem is a smaller MDVRP

Optimization process:

Basic tabu search for MDVRP

Particularity:

No depot relocation



Part:

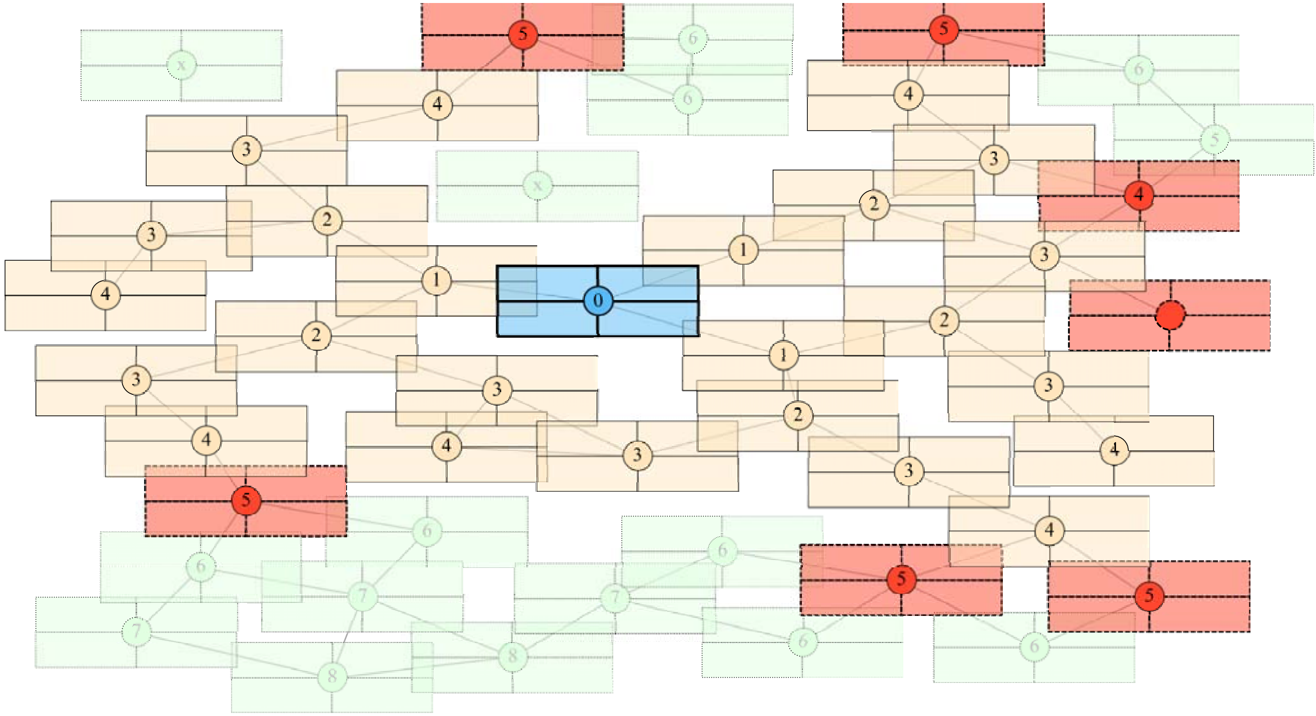
Object to label

Distance between parts:

Minimum number of edges needed to connect parts

Vertex ≡ object

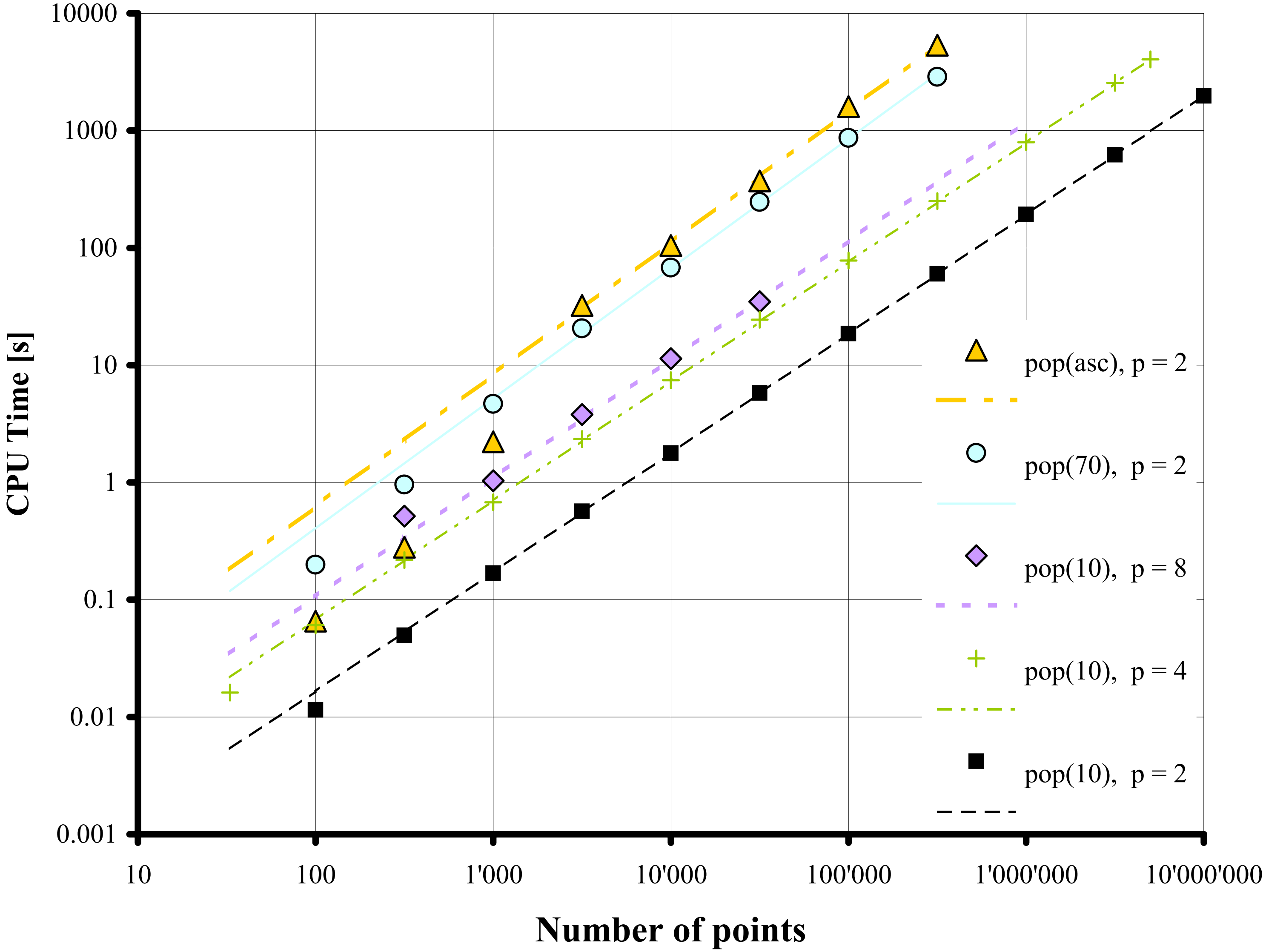
Edge ∃ possible conflict in labelling the objects associated to vertices connected



Optimization process:

Tuned taboo search (Yamamoto, Camara, Nogueira Lorena, 2002), local search with ejection chains

Uniformly generated problem instances, between 30% and 90% of labels without overlap



The complexity grows typically quasi-linearly with problem size

Conclusions

POPMUSIC complexity

Can be implemented in $O(n^{3/2})$

Main difficulty : generating an initial solution, finding the r closest parts

⇒ Solved with proximity graph

POPMUSIC options and parameter

Natural stopping criterion

Must have an optimization process for sub-problems

Heuristic

Exact ⇒ Matheuristic

A single parameter r, for defining sub-problem size

⇒ Easy to tune : sub-problem size must meet best efficiency of optimization process

POPMUSIC drawback

Definition of part and sub-problem dependent on problem under consideration

Future

Application to higher dimensional instances

Up to now : Map labelling 2D, Location-routing $2^{1/2}$D, MDVRPTW 3D

What happens for higher dimensions ?

Application to other problems

Testing different definitions for parts

Study of different options

Definition of distance between parts

Management of non-optimized parts

Parallel implementations