A Template for Low-Complexity POPMUSIC Implementation

Éric D. Taillard[1] , Adriana C. F. Alvim[2] , Stefan Voß[3]

1. HEIG-VD, University of Applied Sciences of Western Switzerland

2. Department of Applied Informatics, Federal University of the State of Rio de Janeiro (UNIRIO)

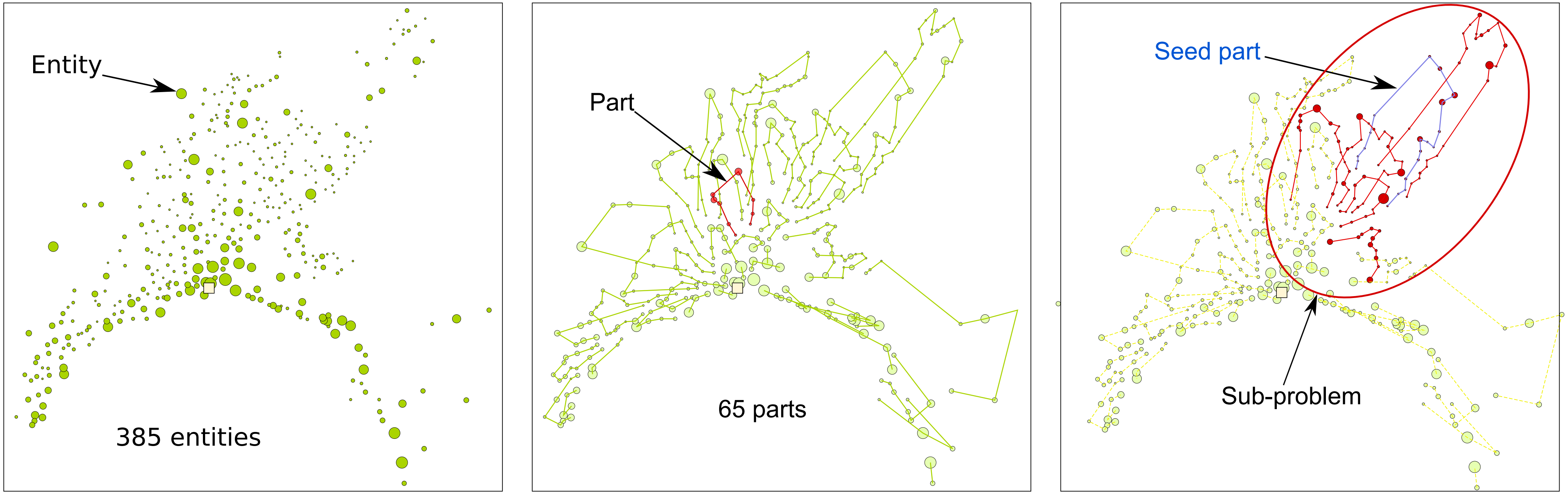
3. University of Hamburg, Department for Business and Economics, Institute of Information Systems

POPMUSIC General idea

POPMUSIC =

Partial OPtimization Metaheuristic Under Special Intensification Conditions

Special Intensification Conditions



A solution can be decomposed into somewhat independent parts

A subset of part (sub-problem) can be optimized almost independently

Pertinence of problem decomposition

Hypothesis

Large problem instances but moderate dimension

⇒ 2 elements close to a third one are also close

⇒ 2 elements far away cannot be both close to a third one

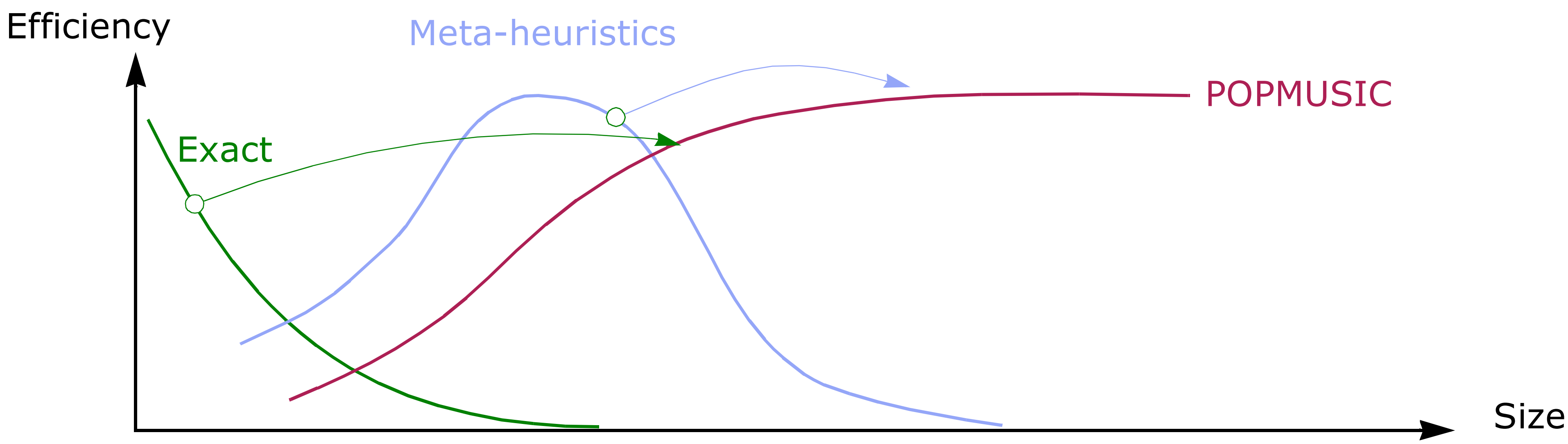
Distant elements are not directly connected together in reasonable solutions

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

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	



Input

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

$U = \{ s_1, s_2, \dots, s_p \}$ // Set of “ un-optimized ” seed parts

While $U \neq \emptyset$, **repeat** // Parts that can 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 $U \leftarrow U \setminus R \cup R^*$

   Else

   Set $U \leftarrow U \setminus s_i$

RELATED CONCEPTS

Candidate list, strongly determined and consistent variables (Glover)

“Chunking” (Woodruff)

Large neighbourhoods (Shaw)

Granular search (Toth & Vigo)

VDNS (Hansen & Mladenovic)

Matheuristics

Exchange (Pochet & Wolsey)

Fix-and-Optimize (Helber & Sahling)

POPMUSIC Algorithmic Complexity

Number of repetition of "While" loop

Empirically proportionnal to problem size

i.e. similar to local search, simplex pivoting

1. Choosing a seed part

Constant time

Random choice, Stack

2. Finding the *r* closest parts

Without specialized data structure : Proportionnal to problem size

3. Optimizing one sub-problem

Exponential with *r* but independent from problem size

In practice : *r* is fixed, so optimizing a sub-problem is performed in constant time

4. Updating *U* set

Proportionnal to *r*

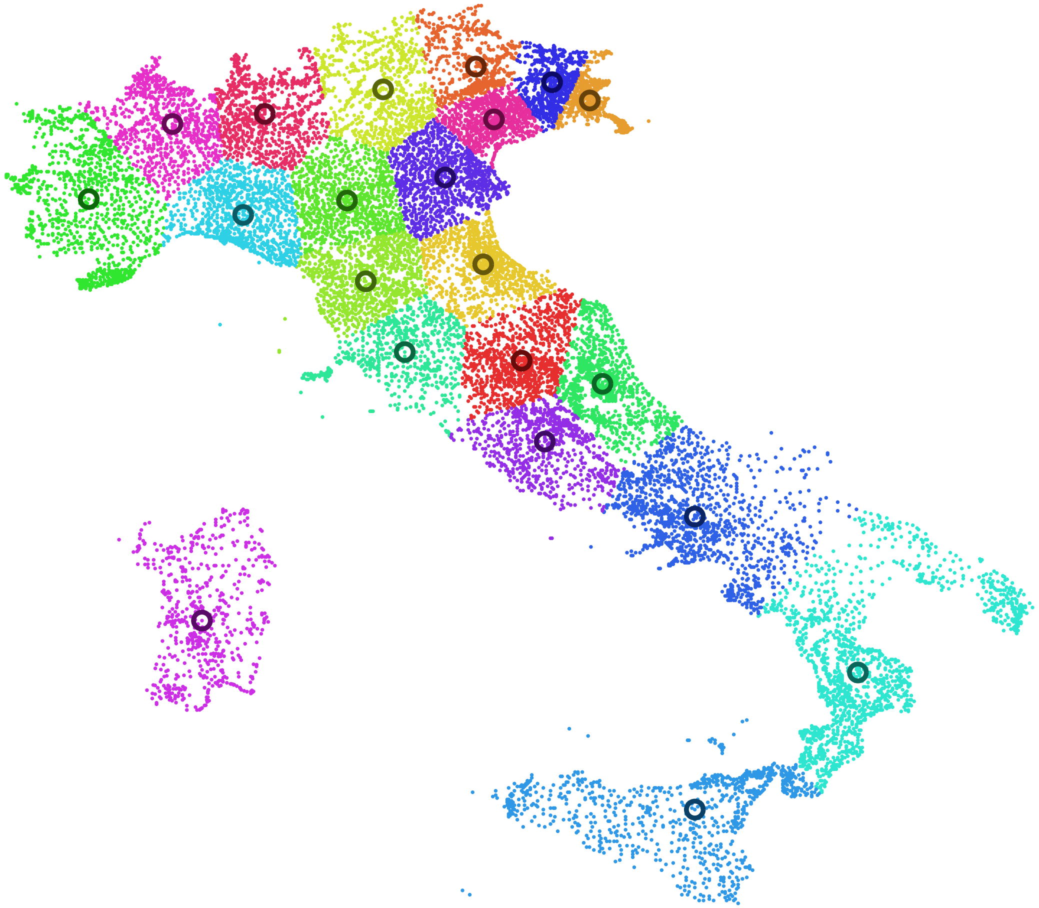
THE P-MEDIAN PROBLEM

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 $\sum_{i=1}^{n} min_{j=1,\ldots,p}(d_{i,c_j})$



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 $30\sqrt{n}$ elements

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

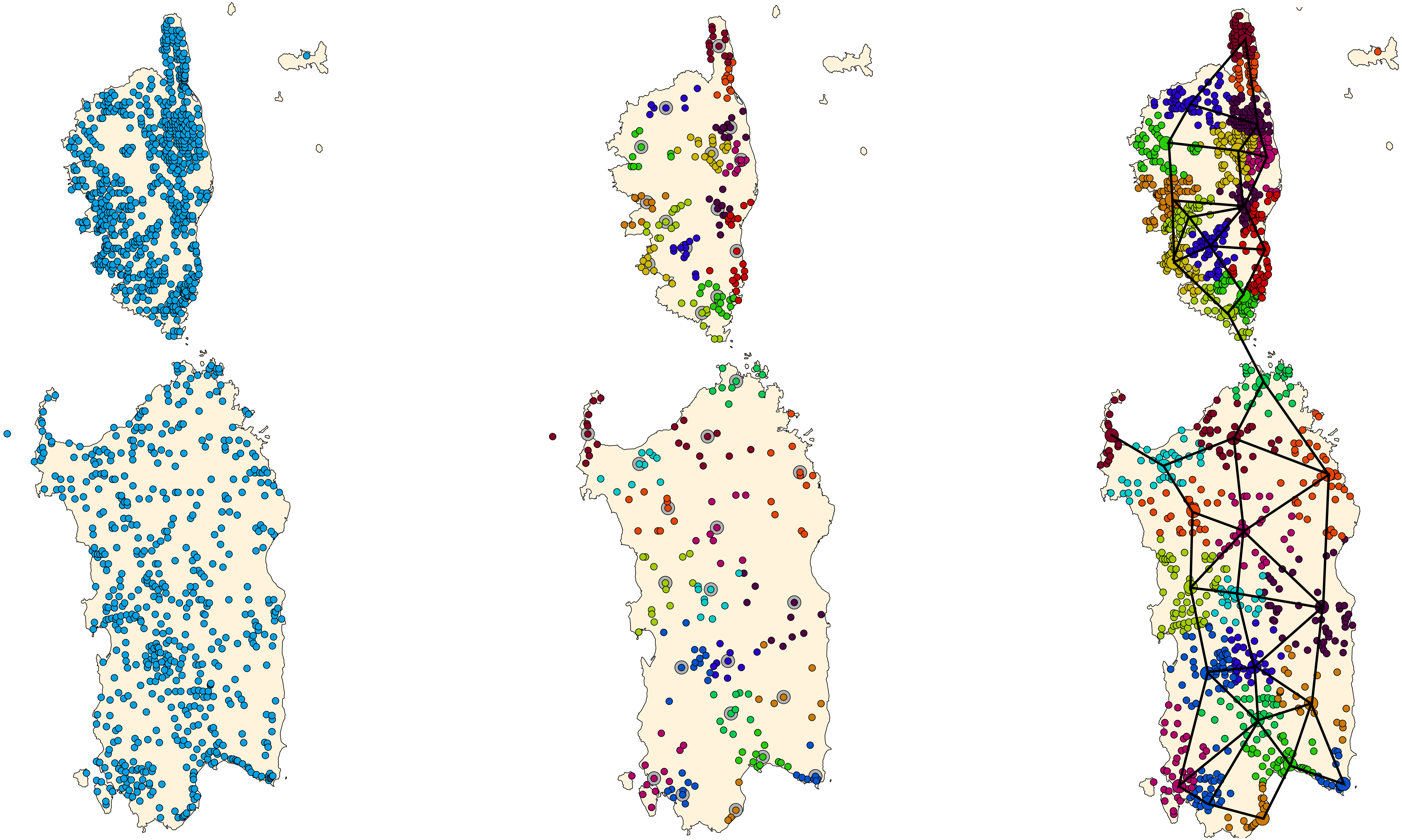
3 Assign each of the n elements to its closest among the p centres
$\Rightarrow \sqrt{n}$ clusters with $\sim\sqrt{n}$ elements each

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

```
Randomly select a sample of e = min(n, 30·p) elements among the n entities
Randomly choose p centres c_1, …, c_p among the e entities
f = 0.6;
λ_j = 0; j = 1, …, p; // Penalty for each centre
for 30 iterations do
    for i = 1 to e do
        Allocate entity i to the centre j minimizing d(i,j) + λ_j
    for j = 1, …, p do
        Find the best position of centre c_j among entities assigned to it
    Compute solution cost C and store solution if best improved
    f ← 0.99 · f
    for j = 1, …, p do
        λ_j ← Max(0, λ_j + f · C · (n_j – e/p)·e^2)
        // n_j : number of entities allocated to centre j
for i = 1 to n do
    Allocate entity i to the centre j of best solution minimizing d(i,j)
```

Complexity

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

Problem instances

From TSP library

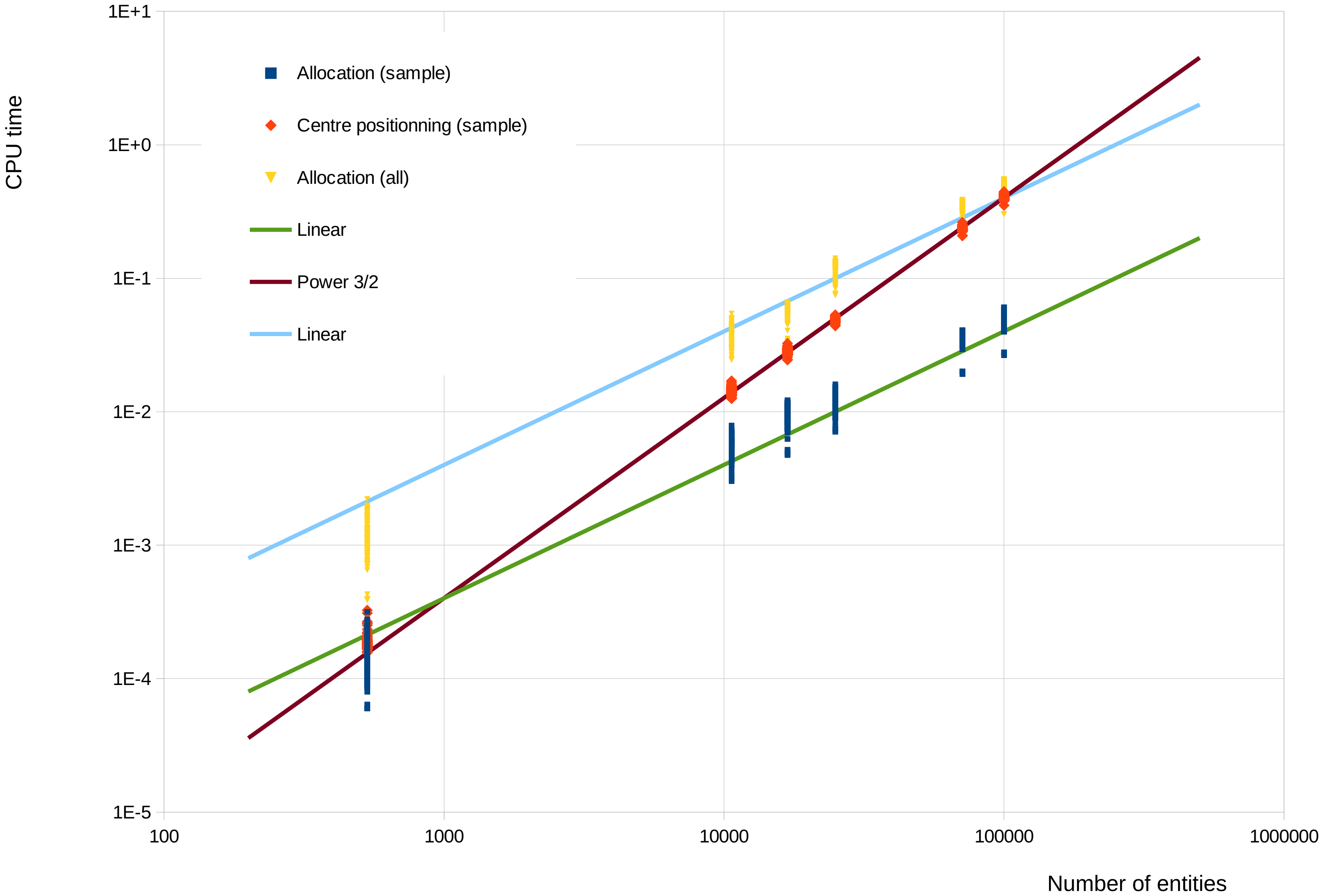
Att532	n = 532
Finland	n = 10639
Italy	n = 16862
Sweden	n = 24978
China	n = 71009
Mona-Lisa	n = 100000

Note :

Assuming Euclidean instances and using KD-Tree, k-means algorithm can be implemented much more efficiently

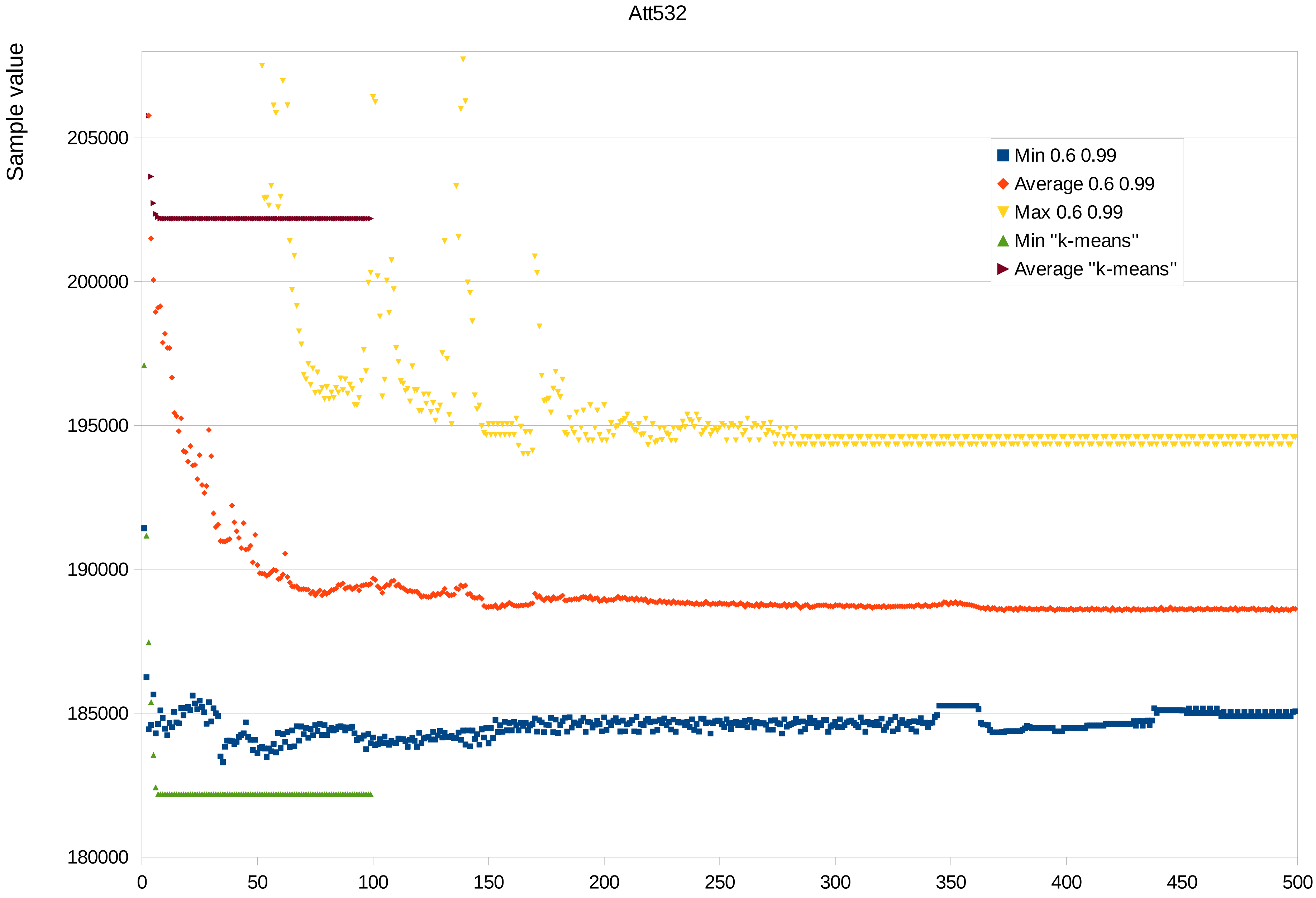
CPU Time

Seconds per iteration on intel i7 930 2.6GHz

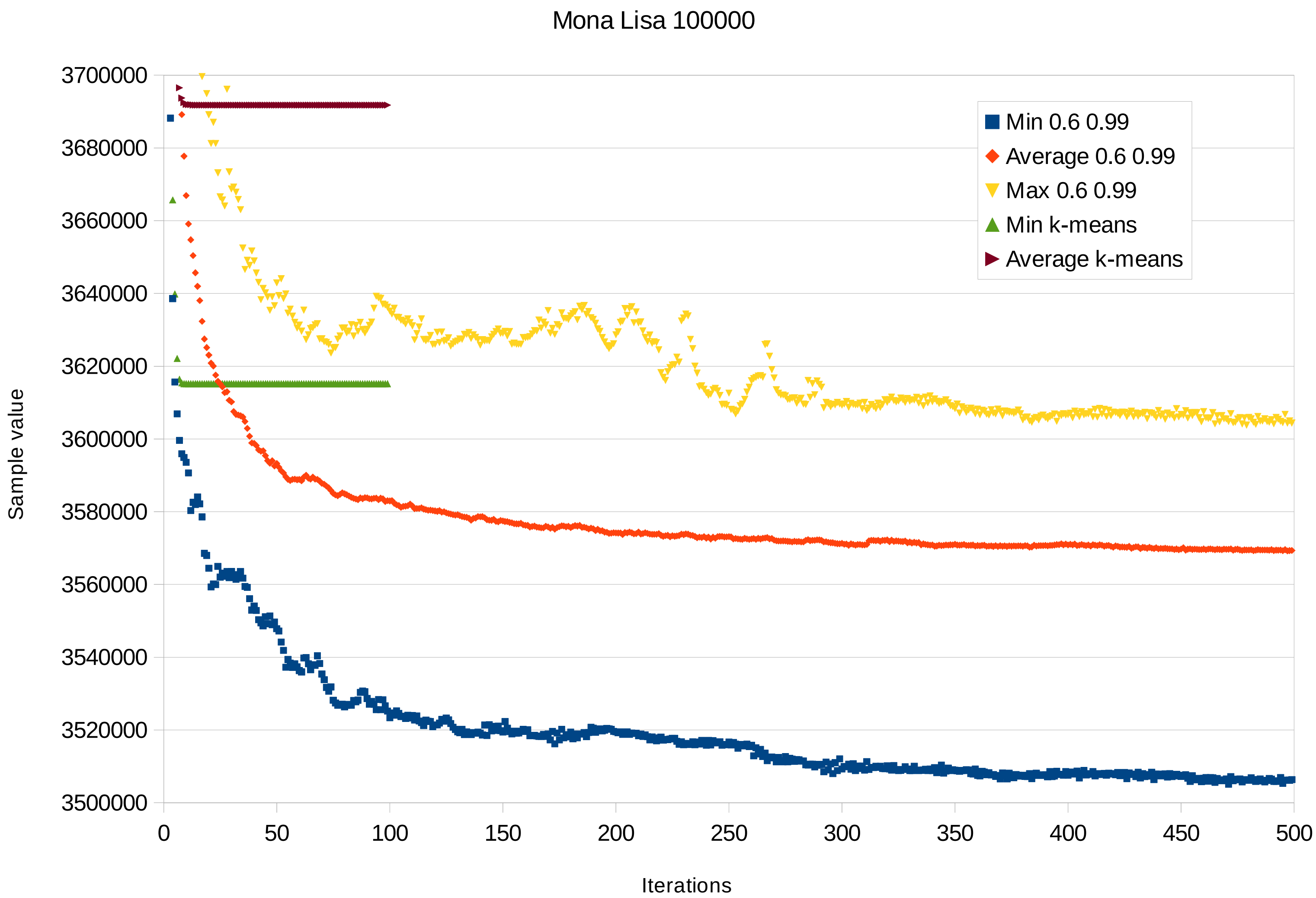


Solution quality

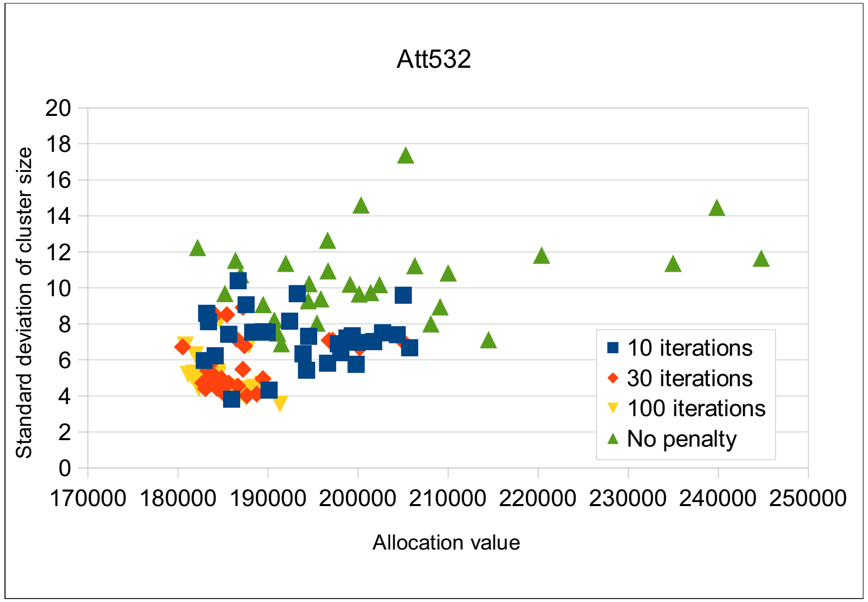
Comparison with k-median heuristic, 30 independent runs

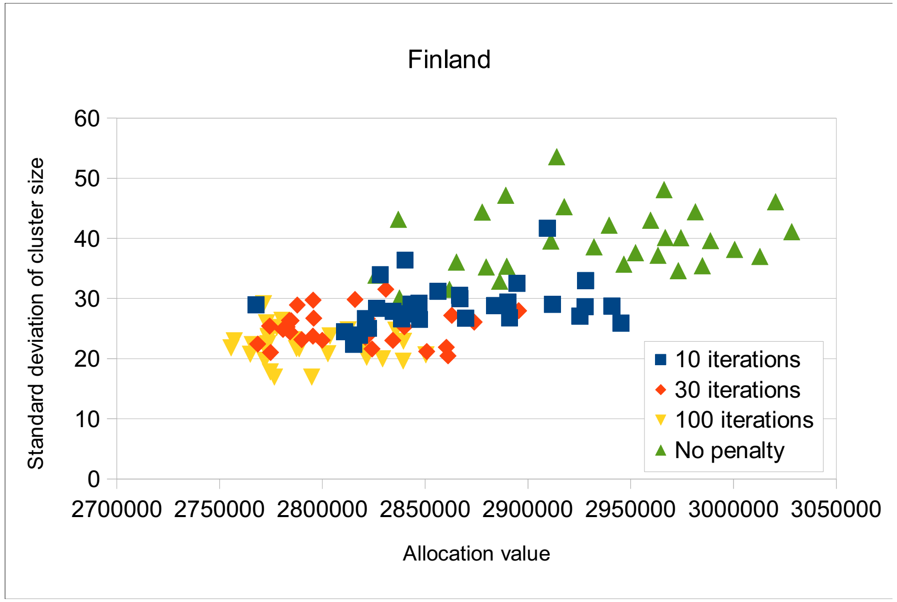


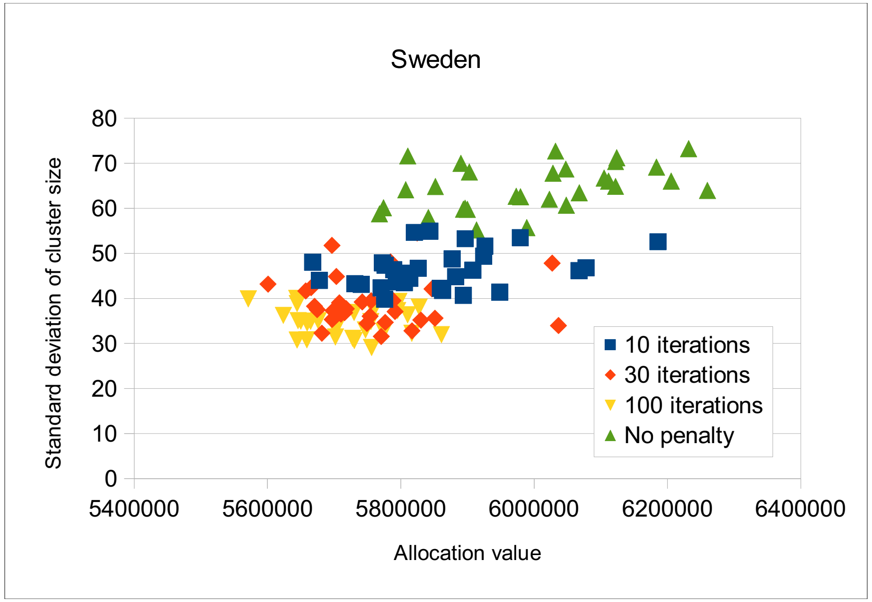
Mona Lisa, 30 independent runs

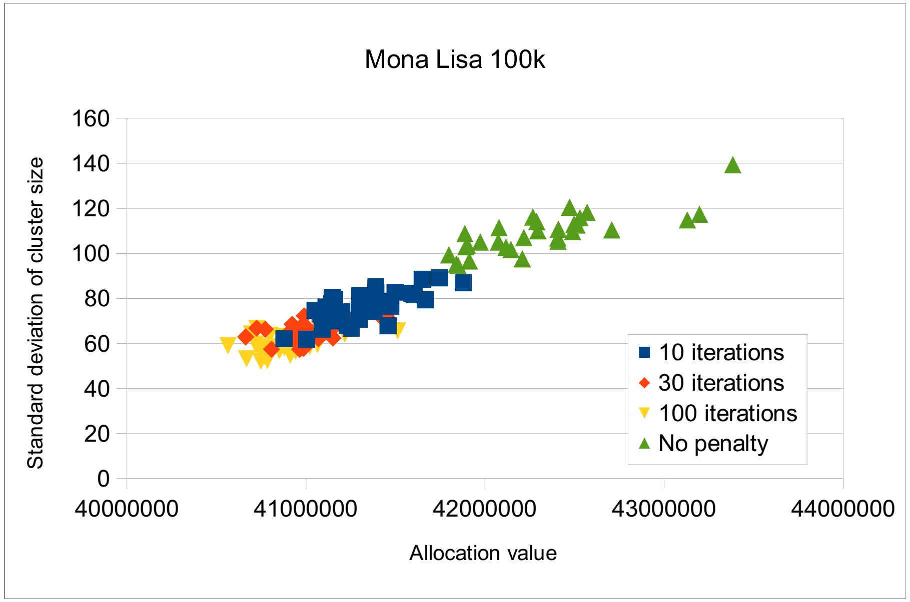


Standard deviation of cluster size



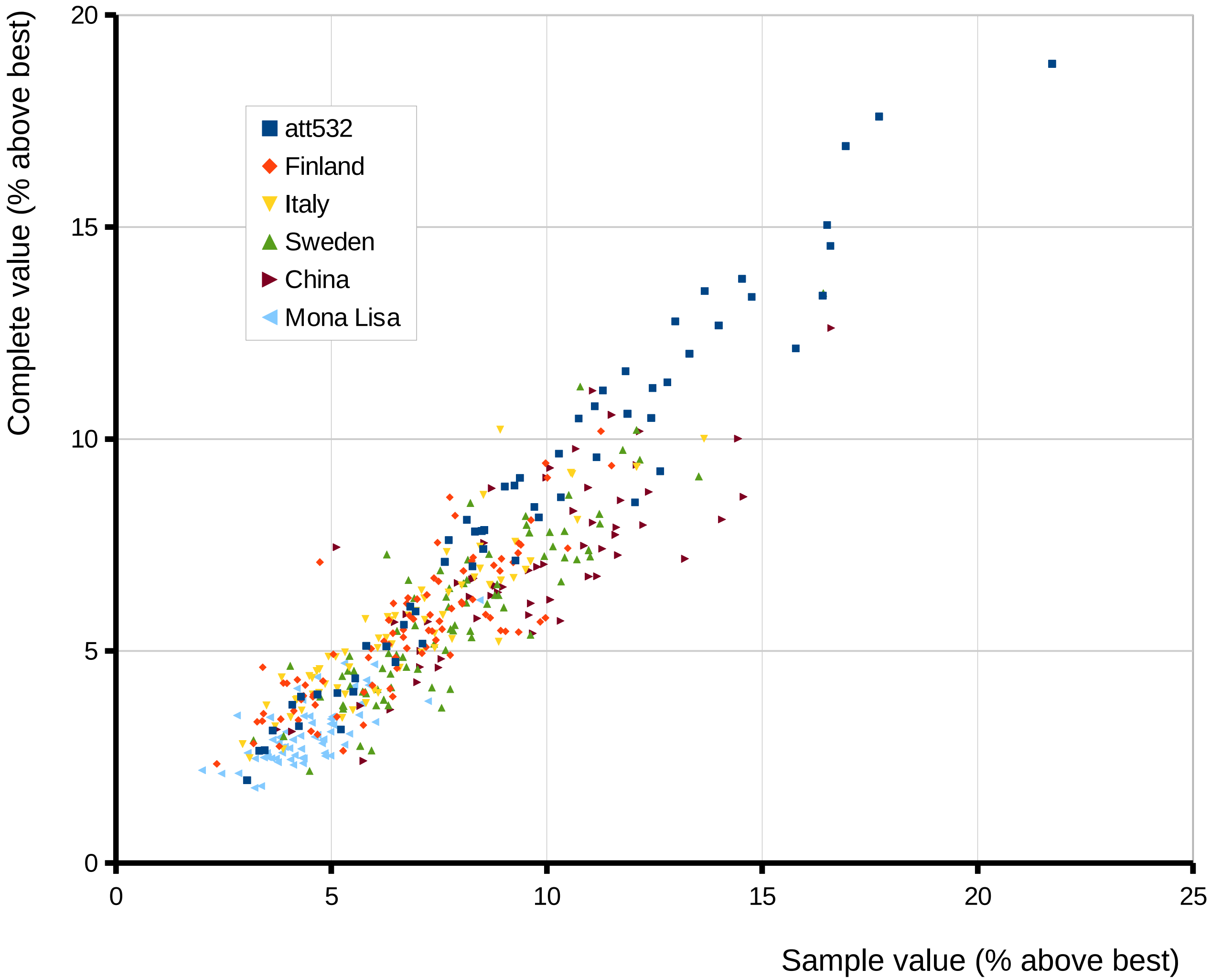




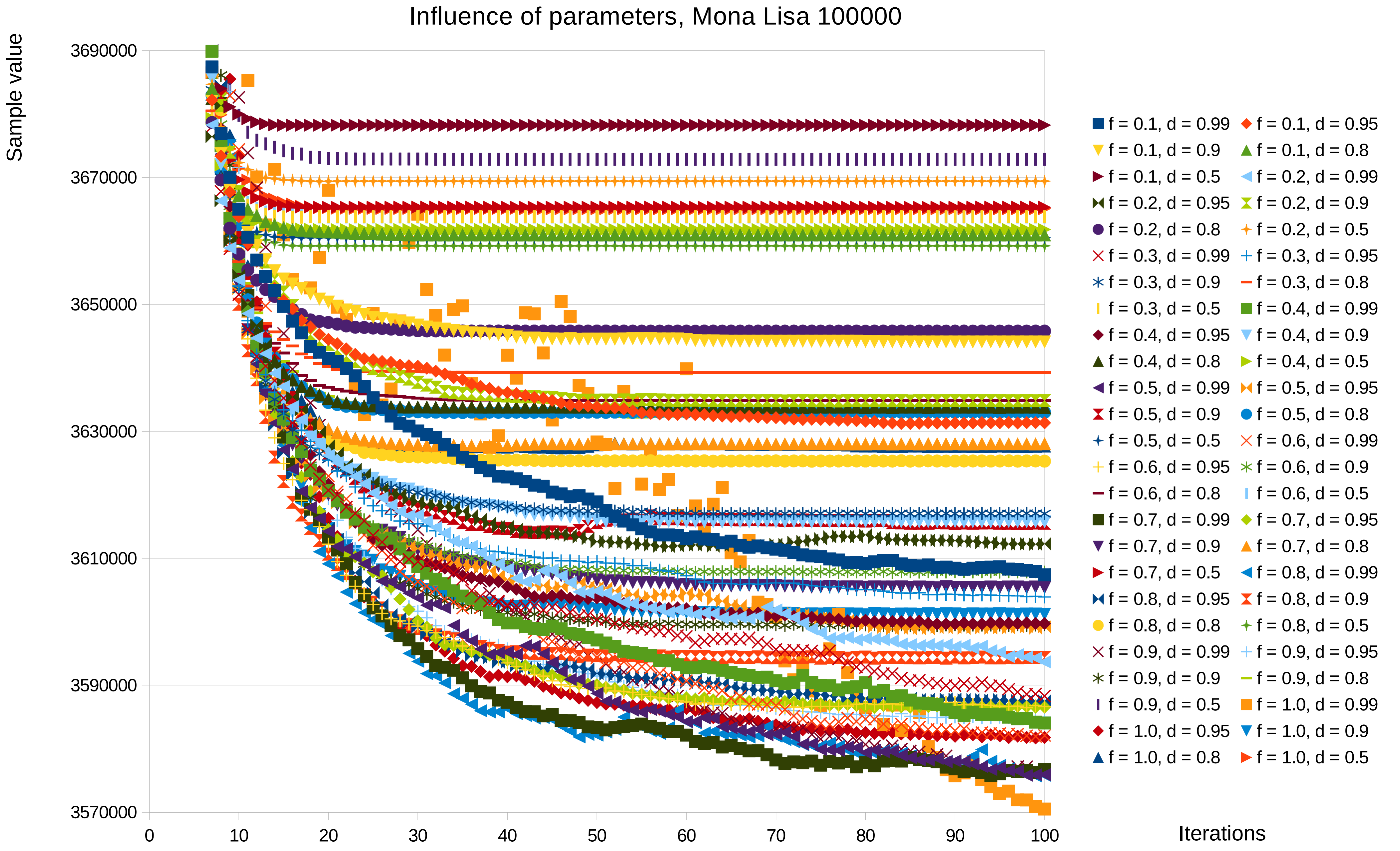


CORRELATION BETWEEN SAMPLE AND COMPLETE COST

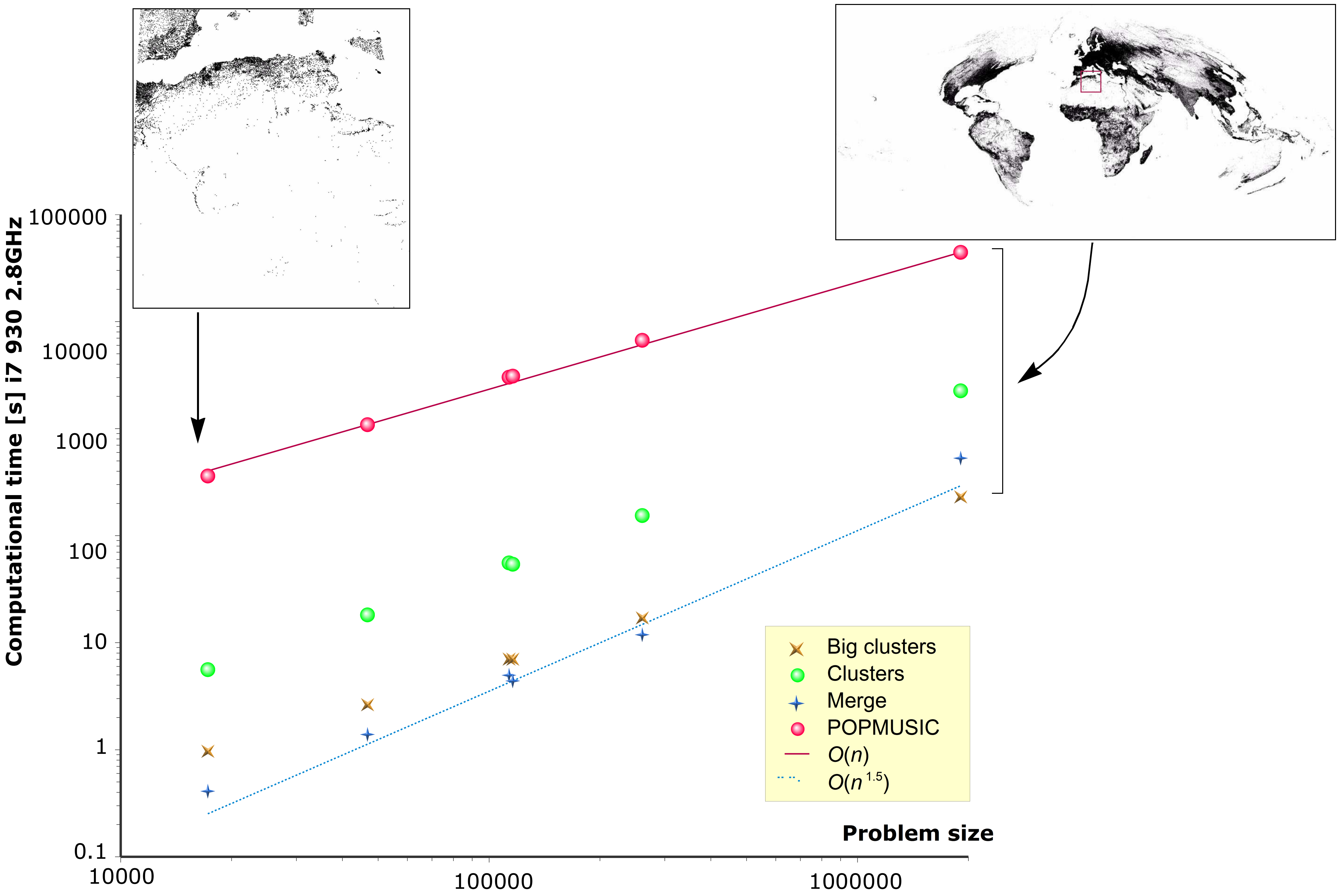
Reference : best solution found during all runs



Parameter Tuning



Application to Location-Routing : Empirical algorithmic complexity



CONCLUSIONS

POPMUSIC complexity

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

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

$\Rightarrow$ Solved with proximity graph

POPMUSIC options and parameter

Natural stopping criterion

Must have an optimization process for sub-problems

Heuristic

Exact $\Rightarrow$ Matheuristic

A single parameter *r*, for defining sub-problem size

$\Rightarrow$ 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

Building different proximity graphs

Parallel implementations