A template for low-complexity implementation of POPMUSIC

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

[1] HEIG-VD, Univ. Applied Sciences of Western Switzerland, Route de Cheseaux 1, CP521, CH-1401 Yverdon-les-Bains, Suisse. eric.taillard@heig-vd-ch

[2] Department of Applied Informatics, Federal University of the State of Rio de Janeiro (UNIRIO), Avenida Pasteur 458, Rio de Janeiro, RJ, 22.290-240, Brazil. adriana@uniriotec.br

[3] University of Hamburg, Department for Business and Economics, Institute of Information Systems, Von-Melle-Park 5, D-20146 Hamburg, Germany. stefan.voss@hamburg.de

Abstract

POPMUSIC is a template for creating heuristics specially designed for dealing with large combinatorial problems that can be partially optimized. The basic idea is to optimize sub-parts of solutions until a local optimum is reached. The useful computational effort should be spent for optimizing these sub-parts. However, it is observed in practice that this computational effort grows almost linearly with problem size while the extraction of sub-parts has an higher algorithmic complexity. This work proposes a template for extracting sub-parts also in linear time.

1 Introduction

It is supposed in POPMUSIC template that a solution s can be decomposed into p parts $s_1, \ldots, s_q$ and that a proximity measure between two parts has been defined. Let U be the set of parts that have not been used as seed-part for building a sub-problem. Initially, U contains all p parts and, at the end, U is empty. Let m be an optimization method that allows to optimize a sub-problem R composed of r parts. With these hypothesis, the POPMUSIC template can be expressed by Algorithm 1

Algorithm 1: POPMUSIC template.

Input: Initial solution s composed of q parts $s_1, \ldots, s_q$, sub-problem optimization method m
Output: Improved solution s

```
1  U = {s_1, ..., s_q};
2  while U ≠ ∅ do
3      Select s_g ∈ U // s_g : seed-part;
4      Build sub-problem R with the r parts of s that are the closest to s_g;
5      Optimize R with method m;
6      if R improved then
7          Update solution s;
8          Remove from U the parts that do not belong to s anymore;
9          Insert in U the parts of optimized sub-problem R;
10     else // R not improved
11         Remove s_g from U;
```

POPMUSIC has a unique parameter, r which governs the size of sub-problems. This parameter is highly dependent on the procedure that is used for their optimization at line 5. When having chosen the sub-problem optimization procedure, it is generally relatively easy to find a good value for r. A very simple technique is to start with a small r and increase its value until the computational time reaches a limit. This strategy is used for instance in [3].

Once the size of sub-problems is fixed, the computational effort for solving one of them can be considered as a constant. Since the while loop at line 2 is repeated a number of times that grows quasi-linearly with problem size [1, 2, 4, 5], it is very important for dealing with large problem instances to

limit the algorithmic complexity of the remaining steps of POPMUSIC, especially the generation of an initial solution and step 4.

Building an initial solution with a good structure is essential for the success of a POPMUSIC implementation. Indeed, the template is supposed to be applied to problem instances meeting *Special Intensification Conditions* — explaining the last 3 letters of the acronym — where an element of the problem is not supposed to have direct interactions with any other elements, but only with a limited number of elements closely related. In the context of location-routing, [2] have proposed to build an initial solution on the base of a p-median solution. However, building an initial solution is highly dependent on the problem under consideration and it is not possible to provide a general template for having a procedure with low algorithmic complexity.

In the present work, we suggest to use also a p-median solution for building closeness relations among the n elements of a problem instance in $O(n^{3/2})$.

2 Creating closeness relations

For creating closeness relations, we make hypothesis that are commonly met when dealing with large problem instances:

- A closeness function $d(i, j)$ that can be computed independently from the problem size is available for measuring the distance between any pair (i, j) of elements of the problem.
- The maximal number of elements of a sub-problem is fixed and does not depend on the total number of elements of the problem.
- Relatively few elements (at most $\sqrt{n}$ on the average) can be considered as close to a given element.

Without a closeness data structure built in a preprocessing phase, a POPMUSIC implementation may have a complexity growing at least quadratically, which is impracticable for large instances. In such a case, instead of spending most computational effort for performing useful work (the optimization of sub-problems), most effort is spent for building sub-problems.

The present work suggests to build the closeness relationship by first solving a kind of p-median problem on the entities of the problem, by choosing $p = \sqrt{n}$. The pure p-median problem aims at choosing p central element for creating p clusters, where each element is assigned its closest centre and with the objective to minimize the sum of the distance between elements and centres.

However, the p-median problem is NP-hard and an optimal solution is not necessarily the best for creating a closeness relation. Indeed, few clusters might contain most of the elements, if the data have a very dense region, while others clusters might contains only few elements. So, the number of elements must be balanced and contain about $p/n = \sqrt{n}$ elements each. To get such a clustering, we suggest to use a variation of the method proposed in [2] for creating an initial solution to large location-routing problem. Algorithm 2 presents how to solve heuristically this p-median problem with soft capacity. When invoked with $p = \sqrt{n}$, Algorithm 2 has a complexity in $O(n^{3/2})$, due to step 14.

3 Computational results

A suggested value for parameter $sample\_size$ is 30, which speed-up a lot the computational time when n is large and allows to produce relatively good solutions. With this parameters and performing $iter = 30$ iterations, the computational time is limited to few seconds for problem instances up to 100,000 entities.

Even if it tries to solve a problem more complex than the pure p-median, this algorithm is generally able to find better solutions than a k-means-like heuristics (an improving method that alternate allocation and centres positioning until all entities are allocated to the best possible centre and all centres are at the best possible place among the entities allocated to them), as shown in Table 1.

Algorithm 2: Generating a p-median with well balanced clusters

Input: n entities, number p of clusters, closeness measure $d(i,j)$ between entities, parameters $sample\_size$ and $iter$
Output: p clusters

Randomly select a sample of $e = \min(n, sample\_size \cdot p)$ elements;
Randomly choose p centres among the e entities;
$f = 0.6$; $\lambda_j = 0; j = 1 \ldots, p$;
for $iter$ *times* **do**
    for $i = 1$ *to* e **do**
        Allocate entity i to the centre j minimizing $d(i,j) + \lambda_j$
    for *Each of the* p *clusters* **do**
        Find the best position of the centre among entities assigned to that centre
    Compute solutions cost C and store solution if it improves best solution found;
    $f \leftarrow 0.99 \cdot f$;
    for $j = 1$ *to* p **do**
        Compute the number n_j of entities allocated to centre j;
        $\lambda_j \leftarrow \max(0, \lambda_j + f \cdot C \cdot (n_j - e/p)/e^2)$;
Allocate all n entities to the best position found for the p centres;

Instance	size	Sample cost	Final allocation	Cluster imbalance
Finland	10639	-4.9	-5.2	-36.7
Italy	16862	-2.4	-3.8	-36.0
Sweden	24978	-2.9	-4.5	-42.1
China	71009	-3.5	-4.7	-29.3

Table 1: Improvement of performances of Algorithm 2 over k-means like heuristics, expressed in percent. Problem instances are issued from National TSP library.

Once $p = \sqrt{n}$ centres positions have been found, several possibilities exist for defining closeness relations between centres while using methods easy to implement. A first possibility is to retain for each centre the r closest ones by complete enumeration. Two entities allocated to the same centre or to centres that are close might be close and the closeness relation can be computed between these entities. The complexity of building relations between entities with that technique is $O(n^{3/2})$.

Another way is to consider that two centres are close if one if the closest of an entity and the other the second closest one. Figure 1 illustrates this way of defining closeness relations.

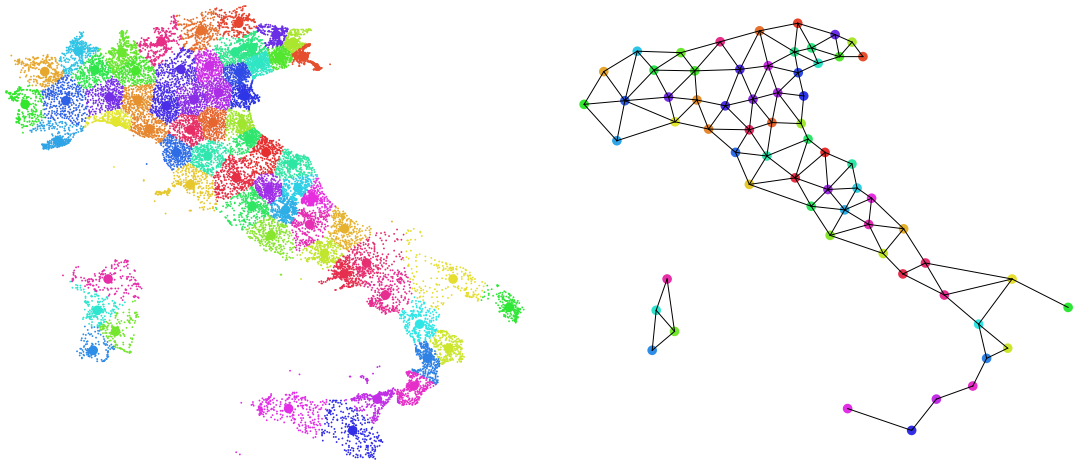
References

[1] Adriana C. F. Alvim and Éric D. Taillard. Popmusic for the point feature label placement problem. *European Journal of Operational Research*, 192(2):396–413, 2009.

[2] Adriana C. F. Alvim and Éric D. Taillard. Popmusic for the world location routing problem. *EURO Journal on Transportation and Logistics*, 2(3):231–254, 2013.

[3] Alessandro Hill and Stefan Voß. An equi-model matheuristic for the multi-depot ring star problem. Technical report, Department of engineering management, University of Antwerp, Belgium, 2014.

[4] Maxence Laurent, Éric D. Taillard, Oliver Ertz, Francis Grin, Daniel Rappo, and Sébastien Roh. From point feature label placement to map labelling. In *Metaheuristic Interantional Conference (MIC'09) proceedings*, 2009.



(a) Problem decomposition into clusters (b) Building proximity network

Figure 1: Illustration of the decomposition of a problem into $\sqrt{n}$ clusters and building a proximity network between the clusters.

[5] Alexander Ostertag, Karl F. Doerner, Richard F. Hartl, Éric D. Taillard, and Philippe Waelti. Popmusic for a real-world large-scale vehicle routing problem with time windows. *JORS*, 60(7):934–943, 2009.