EURO J Transp Logist (2013) 2:231–254
DOI 10.1007/s13676-013-0024-2

RESEARCH PAPER

POPMUSIC for the world location-routing problem

Adriana C. F. Alvim · Éric D. Taillard

Received: 21 May 2012 / Accepted: 1 March 2013 / Published online: 9 April 2013

Abstract POPMUSIC—partial optimization metaheuristic under special intensification conditions—is a template for tackling large problem instances. This template has been shown to be very efficient for various combinatorial problems like *p*-median, sum of squares clustering, vehicle routing and map labelling. In terms of algorithmic complexity, one of the most complex part of POPMUSIC template is to find an initial solution. This article presents a method for generating an appropriate initial solution to the location-routing problem by producing in $O\ (n^{3/2})$ an approximate solution to the capacitated *p*-median problem. The method is tested on LRP instances with millions of entities.

Keywords Location routing · Meta-heuristics · POPMUSIC · TSP · VRP · *p*-Median · Large-scale optimization · Taboo search

Introduction

Location-routing problems are well-known optimization problems. The reader is referred to [8] for a survey. This reference reports that practical location-routing problems may involve thousands of entities that have to be serviced from a set of hundreds of depots. However, high-quality heuristics like GRASP, GRASP with

A. C. F. Alvim
Department of Applied Informatics, Federal University of the State of Rio de Janeiro, Avenida Pasteur 458, Rio de Janeiro, RJ 22.290-240, Brazil
e-mail: adriana@unriotec.br

É. D. Taillard (✉)
Department of Industrial Systems, University of Applied Sciences of Western Switzerland, Route de Cheseaux 1, Case postale, CH-1401 Yverdon, Switzerland
e-mail: eric.taillard@heig-vd.ch

Path Relinking [12], memetic algorithms with population management [11], granular taboo search [13] treats small instances including few hundreds of entities and a dozen of potential depot locations. Even clustering approaches like [2] are not considering larger instances.

The goal of this paper is to show that much larger instances can be treated by embedding these high-quality heuristics in POPMUSIC template.

Several versions of location-routing problems exist, and we have considered the following one in order to generate large and freely available instances based on the World TSP [19]. Let N be a set of n customers (or entities). Each customer has a non-negative demand q_i. It is supposed that a function $d(i, j)$ measuring the distance between customers i and j is available. Set N is also the set of n possible depot locations with unlimited capacity each and where is based an unlimited vehicle fleet, each vehicle having a given capacity Q. The opening cost of a depot is D. The cost of a route is the length of traversed edges. The objective is to find which depots should be opened and which routes should be constructed to minimize the total cost (depots opening cost plus routes cost) such that:

- Each demand q_i must be served by a single vehicle.
- Each route starts and ends at the same depot.
- The total demand of a route does not exceed Q.

The location-routing problem (LRP) considered in this article is a simplified version of a real problem that was submitted to us several years ago. A company producing potato chips has the policy of verifying the freshness of its products and refurnishing itself the shelfs in the selling points. The number of selling points in Switzerland is more than 32,000. The stores are refurnished with small vehicles that carry the goods from local depots. Any room that can be rented can be used as a depot. The depots are refurnished directly from the production places with large trucks, but this is done independently from the LRP. To simplify the problem, we consider that each selling point can be used as depot (with a given renting cost). In practice, it is observed that a vehicle can service few dozen of selling points before turning back to the depot. These simplifications allow to work on problem instances generated on public data whose distribution is not too far from real life. The aim of the present work is to show the pertinence of POPMUSIC approach for dealing with large instances (more than 10^6 entities). The approach presented in this paper can also be used for solving large problem with additional complications, provided that an optimization method for solving small instances is available.

The paper is organized as follows: Sect. "POPMUSIC template" recalls the POPMUSIC template and show how it can be used in the context of the LRP. The main advantage of POPMUSIC approach is that the algorithmic complexity for optimizing a given initial solution is typically linear with the problem size. Such a complexity is lower than that of other approaches specifically designed for limiting the algorithmic complexity of high-quality heuristics. For instance, the approach of [20] for the Vehicle Routing Problem, which is a sub-problem of the LRP, includes a step with quadratic algorithmic complexity. The memory requirement of this approach is also quadratic, meaning that problem instances with, let us say, 10^5 entities cannot be solved.

The main difficulty in implementing a POPMUSIC approach for large instances is to get an initial solution of adequate quality with a complexity that is lower than $O(n^2)$. For instance, the extended Clarke and Wright constructive algorithm used in [12] has a complexity in $O(n^3 m)$, where m is the number of potential depot site ($m = n$ for the instances treated in this article). Section "Efficient generation of a solution to the LRP" presents a technique based on solving approximatively a p-median problem with capacity for getting an initial solution to the LRP in $O(n^{3/2})$.

Computational results are presented in Sect. "Numerical results". Concluding remarks and future research avenues are presented in the last section.

POPMUSIC template

Our first works on POPMUSIC method start in the beginning of the 1990s [14] with an application to the vehicle routing problem with capacity (VRP). In this reference, we used the fact that a subset of VRP tours is also a VRP. By decomposing a given VRP solution into subsets of independent tours, and solving these subproblems in parallel, it is possible to find excellent solutions to VRP instances with few hundreds of customers. [15] shows that a similar principle can be applied to centroid clustering problems (sum of squares clustering, p-median, multi-source Weber problem). The size of the instances considered in this reference is few order of magnitude larger: several thousands of entities and thousands of centers. The implementation discussed in [15] has a complexity growing quadratically with the number of centers and quasi-linearly with the number of entities. This complexity, however, remains quadratic with the number of entities for general p-median instances with data given by a $n \times n$ matrix. Later, in [16], the POPMUSIC template was formalized. Algorithm 1 presents the POPMUSIC template.

Algorithm 1: `POPMUSIC` template

Data: Initial solution S composed of q disjoint parts $s_1, \ldots, s_q$
Result: Improved solution S

```
1  U = s_1, ..., s_q;
2  while U ≠ ∅ do
3      Select s_s ∈ U // s_s: seed part;
4      Build a subproblem R composed of the r parts of S which are the closest
       to s_s;
5      Try to optimize R;
6      if R has been improved then
7          Update solution S;
8          Remove from U the parts not belonging to S anymore;
9          Include in U the parts of optimized subproblem R;
10     else // R not improved
11         Remove s_s from U;
```

The basic idea of POPMUSIC is to locally optimize sub-parts of a solution, once a solution of the problem is available. These local optimizations are repeated until no improvements are found. Let us suppose that a solution S can be represented as a

set of disjoint *parts* $s_1, \ldots, s_q$. Let us also suppose that a distance measure can be defined between two parts. The central idea of POPMUSIC is to select a part s_s, called *seed part*, and a number $r < q$ of the closest parts from the seed part s_s to form a subproblem called R. If parts and subproblems are defined in an appropriate way, an improvement in the solution of the whole problem can be found for every improvement in the subproblem.

To avoid generating twice the same subproblem, a set U of parts is stored. U contains the seed parts that can be used to define a subproblem that can potentially improve the solution. Once U is empty, all subproblems have been examined without success and the process stops. If subproblem R has been successfully improved, a number of parts from $s_1, \ldots, s_q$ have been changed and further improvements may be found in the neighbourhood of these parts. In this case, all the parts used for building R are inserted in U before continuing the process.

Potentially, the complexity of this template can be very high since set U is not reduced at each iteration. However, several implementations [1, 15, 18] shown empirically that the number of iterations of an algorithm based on this template grows quasi-linearly with q.

To translate this template into a (pseudo-) code for a given problem, several choices must be made:

- The procedure for obtaining the initial solution.
- The definition of a part.
- How a seed-part is selected.
- The definition of the distance between parts for creating a subproblem.
- The improvement procedure for optimizing subproblems.

For the LRP, various definitions for a part can be imagined:

- An entity.
- A tour (with all entities constituting the tour).
- A depot (with all tours and entities assigned to the depot).

In the present work, we choose to decompose a solution into tours. So, a part is a tour. The distance between two parts is the minimal distance between two entities belonging to different tours. In order to filter the number of candidate parts, only the tours connected to the r depots that are the closest from the seed part are considered.

The set U is managed as a stack, this means that the last part that has been introduced in U will be the next to be chosen as seed-part. In the present work, we have also tried to select the seed-part randomly in U. The improvement procedure for the subproblems, which are therefore multi-depot VRP (MDVRP), is based on taboo search. The last performs t iterations for a MDVRP containing t entities. The moves considered in this taboo search are the swap of two entities belonging to different tours or the move of an entity from one tour to another. In some situations, this kind of move allows to reduce the number of tours (consequently, the number of vehicles used). The implementation of this taboo search follows the lines of those used in [14]. The complexity of this implementation is $O(t^3)$.

The only parameter of POPMUSIC is r, the number of parts put together to create a subproblem. The value $r = 6$ has been chosen for all experiments in this paper. This value is a good trade-off between the ability of taboo search to find good solutions in a reduced computational effort. Indeed, if subproblems are too small, improvements cannot be found and if the subproblems are too large, their improvements can take a prohibitive computational effort.

Since r is chosen independently from the problem size and since a quasi-linear number of POPMUSIC iterations is observed, the body of POPMUSIC is also quasi-linear. So, the most difficult portion of an adaptation of POPMUSIC to the LRP is to get a feasible initial solution of decent quality. In terms of algorithmic complexity, the initialization is the most complex part. The next section shows how to generate a feasible initial solution of decent quality in $O(n^{3/2})$.

Efficient generation of a solution to the LRP

Since the instances considered in this article are geometrical ones (coordinates on the earth ellipsoid), it would have been possible to use computational geometry algorithms for building a feasible LRP solution, for instance with the help of a Delaunay triangulation. The last can be obtained in $O(nlog\ n)$. We chose a more general method, where we make the hypothesis that we only have a function $d(i, j)$ providing the distance between entities i and j. In this work, we use the Euclidean distance $d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$.

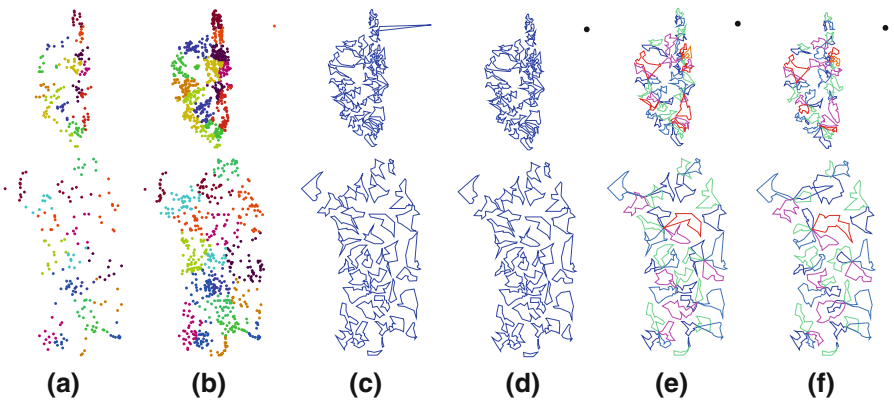

Fig. 1 Main phases of the proposed method illustrated on a small instance including entities on Corsica and Sardinia Islands. **a** Finding superclusters on a sample of entities. **b** Finding superclusters on all entities. **c** Decomposition of supercluster into clusters and building TSP on clusters. **d** Splitting large TSP tours. **e** Merging tours and creating a feasible MDVRP solution. **f** Optimizing MDVRP solution with POPMUSIC (color figure online)

Algorithm 2 has been designed for finding a feasible LRP solution that is suitable for initial POPMUSIC solution. The main phases of this algorithm are illustrated on Fig. 1.

The most difficult point of this algorithm is the implementation of a heuristic procedure for finding good solutions to the capacitated p-median problem (CPMD).

Algorithm 2: Generating a feasible *LRP* solution for large instances

Data: n entities with quantity q_i and distance measure $d(i,j)$, $i,j = 1,\ldots,n$
Result: Feasible *LRP* solution
`// Sampling`
Take a sample E of $20\sqrt{n}$ entities (Figure 1(a));
`// Center location for super-clusters`
Solve a relaxation of a *p*-Median with capacity (*CPMD*) on E with $p = \sqrt{n}$;
`// Super-cluster building` (Figure 1(b))
Assign all n entities to the closest among p centers found at previous step ;
`// Clusters building`
for $k = 1,\ldots,\sqrt{n}$ **do**
    Decompose super-cluster SC_k into $p_k = \lceil \sum_{i \in SC_k} q_i / T \rceil$ clusters by solving a relaxation of a *CPMD*
`// TSP building` (Figure 1(c))
for *all clusters obtained at previous step* **do**
    Find a traveling salesman tour on the cluster entities
`// Splitting large TSP` (Figure 1(d))
for *all tours found at previous step with length larger than* D **do**
    Decompose the entities of the tour into $1, 2, \ldots$ clusters by solving *CPMD*s.
    Find a *TSP* tour for each cluster and for each decomposition.
    Retain the decomposition for which the sum of *TSP* lengths $+ d \cdot D$ is the lowest, where d is the number of clusters in the decomposition
`// Locating depots`
Open a depot for each tour (at the position of *CPMD* center);
repeat
    for *each depot* **do**
        Try to merge the depot with a depot at distance less than D; the tours attached to the removed depot are attached to the new depot position with a cheapest insertion rule. The merging of 2 depots is done in a greedy manner, if the resulting cost is diminished.
until *No merge with cost decrease exists*;
`// Feasible MDVRP` (Figure 1(e))
Cut each tour for which the sum of quantities is larger than Q (vehicle capacity) into 2 tours;

The CPMD can be formulated as follows, where x_{ij} are variables indicating if entity i is assigned to center j ($x_{ij} = 1$) or not ($x_{ij} = 0$):

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} d(i,j) \cdot x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{1}$$

$$x_{ij} \leq x_{jj} \quad \forall i, j \in \{1, \ldots, n\} \tag{2}$$

$$\sum_{j=1}^{n} x_{jj} = p \tag{3}$$

$$\sum_{i=1}^{n} q_i \cdot x_{ij} \leq Q \quad \forall j \in \{1, \ldots, n\} \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in \{1, \ldots, n\} \tag{5}$$

The objective is to minimize the star distances (sum of the distance of all entities to their allocated center). Set of constraints (1) ensures that an entity is allocated to a single center. Set of constraints (2) ensures that an entity cannot be allocated to a center that is not opened. Note that variable x_{jj} indicates if entity j is considered as a center. Constraint (3) ensures that exactly p centers are opened. Set of constraints (4) ensures that a center does not accept a demand larger than its capacity. Note here that all centers have the same capacity Q.

Finding a feasible solution to the CPMD is NP-complete, since the bipartition problem can be polynomially transformed into CPMD. In a first phase, solutions violating slightly constraint (4) can be admitted since the vehicle capacity limitation is satisfied by splitting overloaded tours in step 25 of Algorithm 2. To avoid splitting too many tours and a degradation of LRP solution, we suggest to replace vehicle capacity Q in constraint (4) by a target value $T \leq Q$. This value is a parameter of the method and allows to take into consideration the variance of the demands. To find a solution slightly violating constraints (4), we consider a Lagrangean relaxation of CPMD:

$$\text{maximize}_{\lambda \geq 0} \text{minimize}_{x} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} d(i,j) \cdot x_{ij} + \sum_{j=1}^{n} \lambda_j \cdot \left(\sum_{i=1}^{n} q_i \cdot x_{ij} - T \right) \tag{6}$$

subject to (1) (2) (3) (5)

A gradient method can be used to find good λ's values, by adjusting the value of λ_j as follows: $\lambda_j \leftarrow \max\left(0, \lambda_j + \ell\left(\sum_{i=1}^{n} q_i \cdot x_{ij} - T\right)\right)$, where ℓ is the step length. In other words, penalty λ_j is increased if constraint (4) is violated for center j and it is decreased (without taking negative values) if center j has still additional capacity. Step length ℓ must be proportional to distance unit and inversely proportional to capacity unit.

To find good solutions to (6), we propose Algorithm 3. This algorithm alternates the allocation of entities to centers, the centers repositioning and the $\lambda' s$ update.

Algorithm 3: Generating an almost feasible solution to the *CPMD*

Data: n entities with quantity q_i, $(i = 1, \ldots, n)$, p, T
Result: Allocation variables x^*: p clusters with quantities close to T

1 Randomly choose p centers among the n entities
2 $f = 1.0$; $\lambda_j = 0; j = 1 \ldots, n$;
3 **for** *150 iterations* **do**
4 Allocate entities to the closest center (with penalty; update x_{ij} variables);
5 **for** *Each of the* p *clusters* **do**
6 Find the best position of the center among cluster entities (update x_{jj} variables)
7 **if** *Current solution* x *is better than* x^* **then**
8 $\mathrm{x}^* \leftarrow \mathrm{x}$
9 $f \leftarrow 0.99 \cdot f$;
10 **for** $j = 1, \ldots, p$ **do**
11 $\lambda_j \leftarrow \max\left(0, \lambda_j + f \cdot \frac{1}{n}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} d(i,j) \cdot x_{ij}\right) \cdot \frac{1}{T}\left(\sum_{i=1}^{n} q_i \cdot x_{ij} - T\right)\right)$

Line 2 of Algorithm 3 introduces variable *f* which is used for setting gradient step length. Initially, the step length ℓ is given by the average distance of entities to centers divided by target cluster capacity. This length is diminished at each iteration by multiplying *f* by 0.99.

Preliminary computational experiments showed that the solution is stabilized after few hundreds of iterations: all entities are allocated to their nearest center (distance + penalty), the centers are positioned at best among the entities and the $\lambda' s$ are not changing enough for modifying the entities allocations. So, we decided to perform a total of 150 iterations. At Line 7, a solution is considered to be better if it diminished the number of clusters for which the sum of quantities is larger than Q, or, for the same number of cluster capacity violations, if the sum of distances is diminished.

As the results of Algorithm 3 depend on a random positioning of the centers, the last is executed 5 times and the best of 5 returned solutions is retained. Finally, for diminishing the number of clusters for which the allocated quantity is larger than Q, a local search is applied. This local search transfers repeatedly an entity from an overloaded cluster to its second nearest center, if the last has enough capacity for accepting this entity. The local search stops when no improving move is found.

Complexity analysis of Algorithm 3

Applied to a CPMD instance with n entities and p centers, the complexity of Algorithm 3 is $O(n \cdot p + n^2/p)$. Indeed, Step 4 can be trivially implemented in $O(n \cdot p)$ and Step 6 in $O\,(n^2/p^2)$ since each cluster has n/p entities on average. The complexity of all the other steps is lower.

Complexity analysis of Algorithm 2

Once a CPMD procedure is available, finding an initial LRP solution of adequate quality is relatively easy, but the size of subproblems to solve must be carefully chosen in order to maintain the complexity as low as possible. The sampling of entities in Step 1 is not there for diminishing the global complexity of Algorithm 2,

but for speeding it up. Finding the positions of supercluster centers consists in calling Algorithm 3 with $O(\sqrt{n})$ entities and $\sqrt{n}$ centers. The complexity of Step 2 is therefore $O(n)$. Without sampling, this phase is in $O(n^{3/2})$ and would take most of the computational time.

The assignment of all n entities to the nearest of $\sqrt{n}$ (super-)center in Step 3 can be trivially implemented in $O(n^{3/2})$.

The decomposition of super-clusters into clusters in Step 4 consists in calling $O(\sqrt{n})$ times Algorithm 3 with $O(\sqrt{n})$ entities and $\sqrt{n}$ centers. Therefore, the complexity of finding $O(n)$ clusters respecting approximatively the capacity Q of the vehicles can be done in $O(n^{3/2})$.

For a given value of Q, the maximal number of entities per tour is also fixed. This means that finding all TSP tours and eventually splitting long tours (Steps 6 and 8) can be done in $O(n)$.

The complexity of Step 16 (merging depots) depends on depot opening cost D. It is useless to merge two depots at a distance larger than D. Making the hypothesis that, for all depots, there is a constant number of other depots at distance lower than D, Step 16 can be implemented with a complexity lower than $O(n^2)$. However, if the opening cost is very high, this step could degenerate in $O(n^2)$. If the number of depots is low $[\text{in } O(\sqrt{n})]$, another depot positioning strategy should be adopted, for instance by solving a PMD problem. Such an approach has been suggested in [13]. However, the instances proposed in this work have a value of D relatively limited and we observed a computational effort for this step that is less than 1.2% of global computational effort.

Finally, making the initial solution feasible in Step 17 can be trivially implemented in $O(n)$. So, the complexity of finding an initial solution to the LRP is in $O(n^{3/2})$. This complexity is empirically verified by numerical experiments in the next section.

Numerical results

Since there are no large size LRP instances publicly available, we have generated new benchmarks on the base of the World TSP data [19]. Six different entities sets have been considered, by taking various windows (longitude, latitude) among the entities of the World TSP. Table 1 gives the main characteristics of these 6 instances which are illustrated in Fig. 2.

Each city i of the TSP World instance is specified by its longitude λ_i and its latitude θ_i, given in degrees and decimal form. Our world LRP instance is specified by its Euclidean x_i, y_i and z_i positions (expressed in meters) computed with the Geodetic Reference System (GRS80 [4]):

$$x_i = r_i * \cos(\lambda_i) * \cos(\theta_i)$$

$$y_i = r_i * \sin(\lambda_i) * \cos(\theta_i)$$

$$z_i = r_i * (1.0 - e^2) * \sin(\theta_i)$$

where $r_i = 6378137.0/\sqrt{1.0 - e^2 * \sin^2(\theta_i)}$, and $e^2 = 0.00669438$.

Table 1 Characteristics of the problem instances considered in this paper

Instance number	Number of entities	Minimal longitude	Maximal longitude	Minimal latitude	Maximal latitude
1	17,237	−5	15	20	40
2	46,750	−5	15	30	45
3	113,193	15	30	40	50
4	115,858	−5	15	40	50
5	260,374	−5	30	30	50
6	1,904,711	−180	180	−90	90

Entities are chosen among those of the World TSP data

For testing the sensitivity of our method with respect to the type of demands for the entities, two types of problem instances were considered. The first type of instances (unit demands) have a uniform demand of 1 for all entities. The second type of instances (variable demands) have demand q_i for each entity that is generated using a pseudo-random generator $q_i = (107 \cdot \lfloor |x_i| \rfloor + 97 \cdot \lfloor |y_i| \rfloor + 163 \cdot \lfloor |z_i| \rfloor) \bmod 29 + 1$. For the second type, we can consider that the quantities are randomly, uniformly generated between 1 and 29 with an average of 15.

For testing the sensitivity of our method with respect to the vehicle capacity, we have considered $Q = 10; 20; 40$ for unit demand instances and $Q = 150; 300; 600$ for variable demand instances (so, a vehicle tour visits 10, 20 or 40 entities on average).

Finally, for testing the sensitivity of our method with respect to the depot opening cost, we have considered $D = 50,000; 100,000; 200,000$.

Parameter settings

Since a method based on POPMUSIC embeds several algorithms, we provide here the various choices we made for the options and parameters used.

POPMUSIC body

- A part: a vehicle tour
- Distance between parts: minimal distance between 2 entities belonging to different tours
- Size of MDVRP subproblems: $r = 6$
- MDVRP improvement procedure: Taboo search
- U set is managed as a stack.

Taboo search for MDVRP

The taboo search used for optimizing MDVRP instances follows the lines of [14].

- Moves: Transfer an entity from one tour to another or exchange 2 elements belonging to different tours

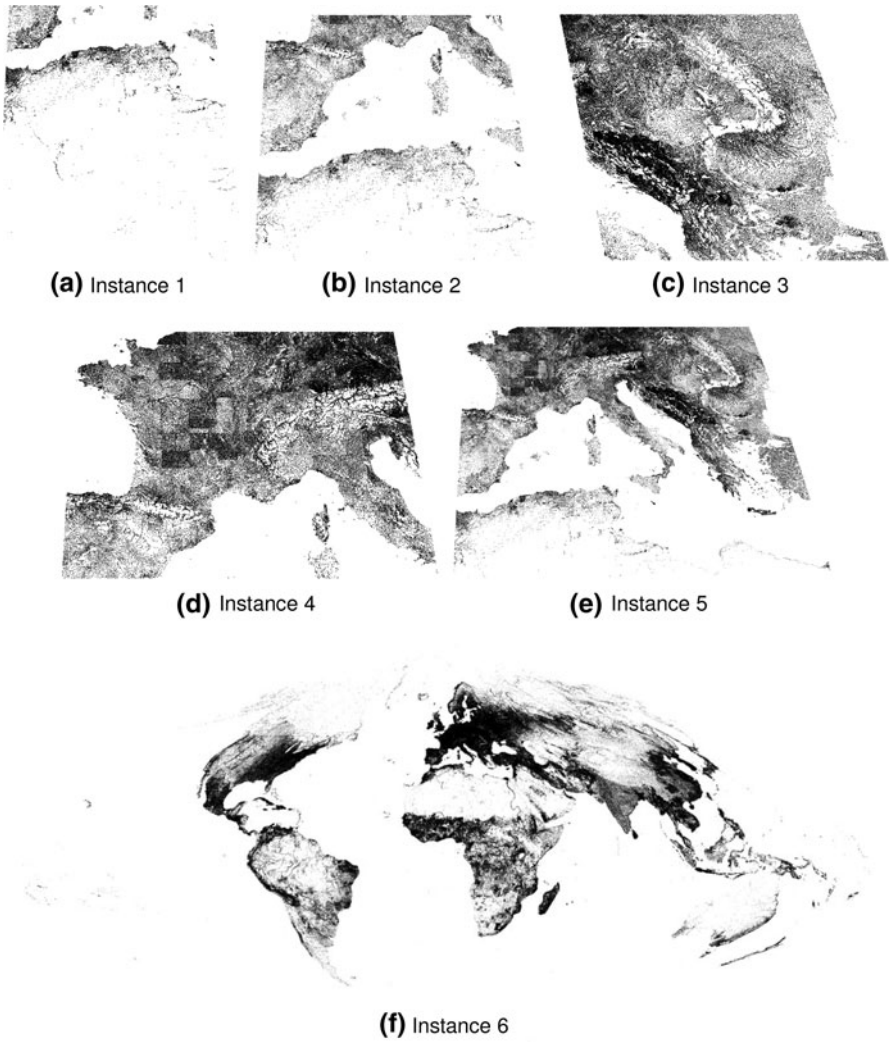

Fig. 2 The 6 regions of the world considered in this paper for creating new LRP benchmarks

- Number of iterations: t for an instance with t entities
- Taboo status: moving again an entity
- Taboo duration: $0.5 \cdot t \cdot u^3 + 0.1t$ where u is a random number uniformly distributed between 0 and 1
- Aspiration criterion: a taboo move improves the best solution or a transfer move empties a tour.

CPMD

- Number of iterations (allocation–λ modification–relocation): 150
- Gradient step size decrease: 0.99
- Target capacity T for the clusters: $T = Q$ for unit demand instances and $T = Q - 10$ for variable demand instances.
- The procedure is repeated 5 times and the best solution is taken.

Analysis of results

We show the sensitivity of our method with respect to the instances, parameter settings and options by considering a standard problem: The demands are variable, $Q = 300$ and $D = 100{,}000$. Additional computational results on different problem instances, obtained by varying the values of Q, D and demands are provided in Sect. Detailed computational results. The method was implemented in C, compiled under Windows 7 with *gcc* compiler and *-O2* option. Only one core of Intel i7 (930, 2.83 GHz) was used. The memory used by our implementation never exceed 1.5 Gb for the largest instance. Table 2 provides for all 6 instances of different sizes the following information and numerical values observed.

- The number of entities
- A lower bound on the number of vehicles ($\lceil \sum q_i/Q \rceil$)
- The number of clusters created
- The total length of TSP tours, before splitting
- The number of TSP obtained after splitting long tours
- The number of depots remaining after merging
- The total length of tours of the initial feasible MDRVP solution
- The total length of tours after improvement with POPMUSIC
- The number of vehicles used in MDRVP improved solution
- Total cost of MDRVP improved solution (which is equal to total length plus number of depots times depot cost)
- A lower bound of optimal total cost
- The CPU time for solving the CPMD (superclusters + clusters)
- The CPU time for POPMUSIC optimization
- The total CPU time.

Since these large instances are new, it is difficult to have an idea about the quality of solutions obtained. Indeed, good lower bounds are hard to find for theses large instances, given that classical mathematical models embeds 3-index variables. The purpose of this article is not to study lower bounds for large LRPs. A simple lower bound can be obtained by computing a spanning tree MST of minimum weight. Then, all the edges of MST that are larger than depot opening cost D are removed to obtain a forest F. A depot is opened for each connected component of F. The lower bound is given by the cost of the edges of F plus depot opening costs. Such a lower bound is tight for special instances, as shown in Fig. 3, where the lower bound is $4D - \epsilon$ and the optimal LRP solution has a value of $4D$. The lower bound is also

Table 2 Results for standard problem instances (Demand uniformly distributed between 1 and 29; $Q = 300$; $T = 290$; $D = 100{,}000$)

	Instance					
	1	2	3	4	5	6
Problem size n	17,237	46,750	113,193	115,858	260,374	1,904,711
Vehicles lower bound	862	2,332	5,670	5,790	13,024	95,238
Number of clusters	961	2,520	6,034	6,157	13,728	99,220
TSP length	100,382,746	196,926,915	295,442,154	334,179,645	782,337,809	9,713,446,886
Number of TSP tours	1,005	2,539	6,036	6,162	13,760	100,581
Number of depots	291	621	1,143	1,226	2,774	25,080
Length MDRVP	98,183,526	217,103,691	344,836,963	386,390,737	895,446,441	10,388,058,905
Length POPMUSIC	91,506,754	200,080,586	315,839,461	353,454,191	818,078,899	9,463,512,294
Number of vehicles	996	2,543	6,047	6,176	13,829	102,504
Total cost	120,606,754	262,180,586	430,139,461	476,054,191	1,095,478,899	11,971,512,294
Lower bound	67,098,741	142,180,289	217,665,536	248,768,006	568,212,071	6,613,228,345
Time clusters	7	21	63	61	172	2,497
CPU POPMUSIC	362	1,093	3,034	3,108	6,712	44,553
Total time	369	1,116	3,102	3,174	6,897	47,598

CPU times in seconds on one Intel i7 core

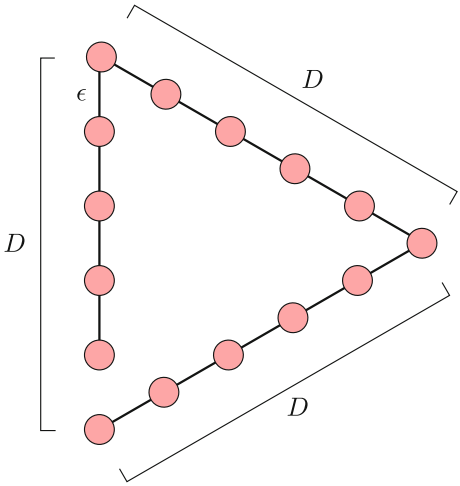

Fig. 3 A LRP instance for which a minimum spanning tree is, asymptotically, a lower bound to the optimal solution. The weight of the spanning tree is $3D - \epsilon$ while the length of an optimal LRP solution is $3D$. At least one depot must be opened, for a cost of D. So, the lower bound is $4D - \epsilon$ while optimum LRP cost is $4D$ (color figure online)

tight for instances with all entities separated by a distance larger than D. In this case, F has no edges. Without making assumptions on the distance measure between entities, the computation of a lower bound is at least in $\Omega(n^2)$. For the largest instance, the time for computing the lower bound given in Table 2 is higher than the CPU time for POPMUSIC optimization. The quality of the lower bounds is certainly very bad.

Let us mention however, that the length (line "Length POPMUSIC") of the solution for Instance 6 (corresponding to the world TSP data) is about 26 % above the length of optimal TSP tour. This length is neither an upper bound to the LRP optimum length (since there is a higher number of travel in a LRP solution due to the capacity constraints implying additional travels to the depots) nor a lower bound (since the TSP tour may have travels larger than D, the opening cost of a depot). For instance, for $D = 15{,}000$ (about 4 times the average travel length between 2 TSP cities), the total length is lower than the optimum TSP length.

In this table, we see that the initial MDRVP solution is improved by about 10% by POPMUSIC. The number of vehicles in the final solution is 6.1–15.5 % above the minimum number of vehicles. Knowing that several depots only service 1 or 2 entities (in sparse regions), an average vehicle load above 90 % means that the vehicle tours are very constrained by their capacity. Even if the upper bounds are near 2 times the value of the lower bounds, we are confident that these results are not too bad, after having made the following experiment: We took few of the largest academic VRP instances publicly available, i.e. those proposed by [6, 20] and we observed the solutions quality obtained with the following method: The entities are first clustered with our CPMD approach (trying different T capacity targets for having sets of entities not larger than vehicle capacity Q) and a TSP is solved for each cluster (+ the unique depot). For these instances with 420 and 3,000 entities, the value of the feasible VRP solutions observed were typically 8–10 % above the best solutions known, reported in [7, 20]. The computational time was less than 0.4 s for the instance with 420 entities and about 10 s for the instances with 3,000

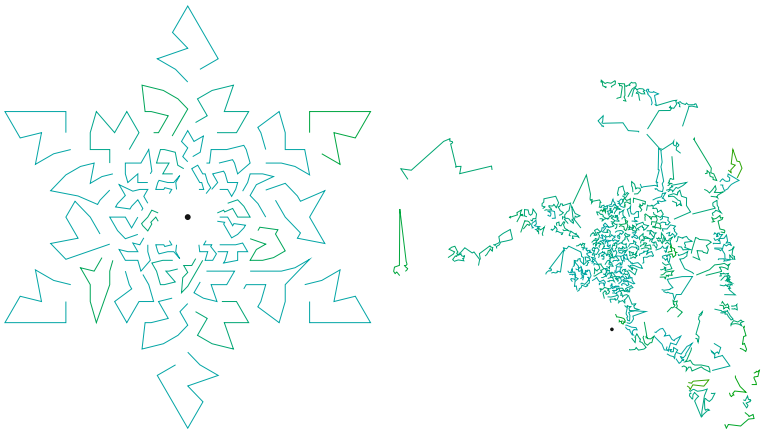

Fig. 4 Solutions of 2 of the largest VRP instances publicly available, obtained just by solving a TSP on the clusters produced by our CPMD procedure. No further improvement of the tours with a local search. The computational time is 0.37 s for (**a**) and 10.4 s for (**b**). The tour length is about 8 % above best solutions known for both instances (color figure online)

entities. For these instances, the best solutions known are 117 and 1,010 % above the lower bound proposed above. Figure 4 illustrates the solutions obtained with this experiment.

Sensitivity with respect to problem size

Figure 5 provides the computing times of the most important steps of our algorithm as a function of the problem size. In this figure, we remark that:

- The building of superclusters and decomposing them into clusters requires a computational effort growing slightly less than the O ($n^{3/2}$) expected.
- The placement of depots by merging TSP tours requires a computational effort in O ($n^{3/2}$).
- The improvement of all subproblems grows linearly with n. This step requires the largest computational effort, even for the largest instance. As observed for centroid clustering [15] and map labelling [1], the number of subproblems optimized in POPMUSIC seems to grows linearly with problem size.

Sensitivity with respect to problem types

Figure 6 provides the average vehicle load for various instances. The vehicle load is the ratio between the sum of quantities on a tour and the vehicle capacity. We see in this figure that the load is systematically above 0.85. For VRP instances, such a load is relatively high. For the LRP the load mainly depends on the sparsity of the entities. The smallest and the largest instances are characterized by large zones with

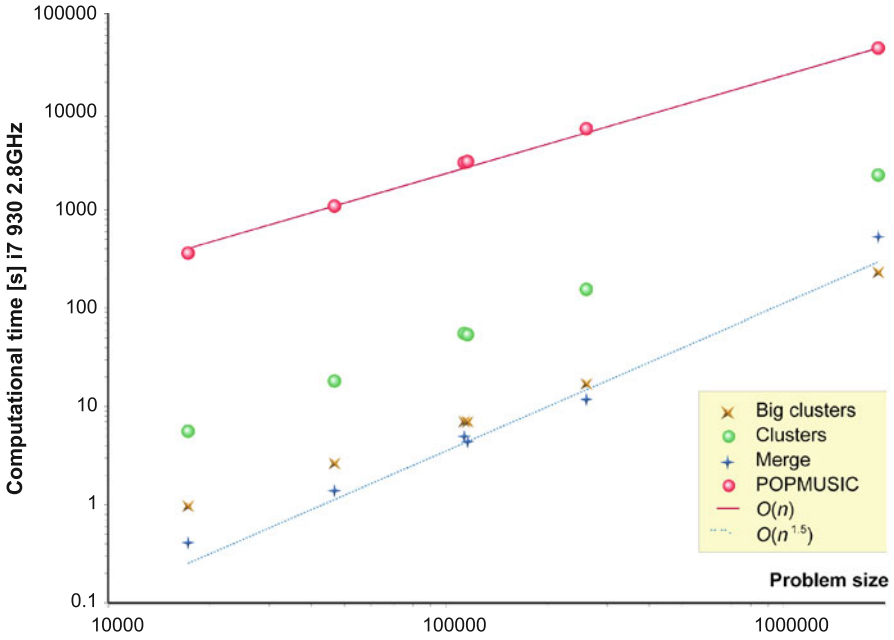

Fig. 5 Evolution of computational effort as a function of problem size, for various parts of the algorithm (color figure online)

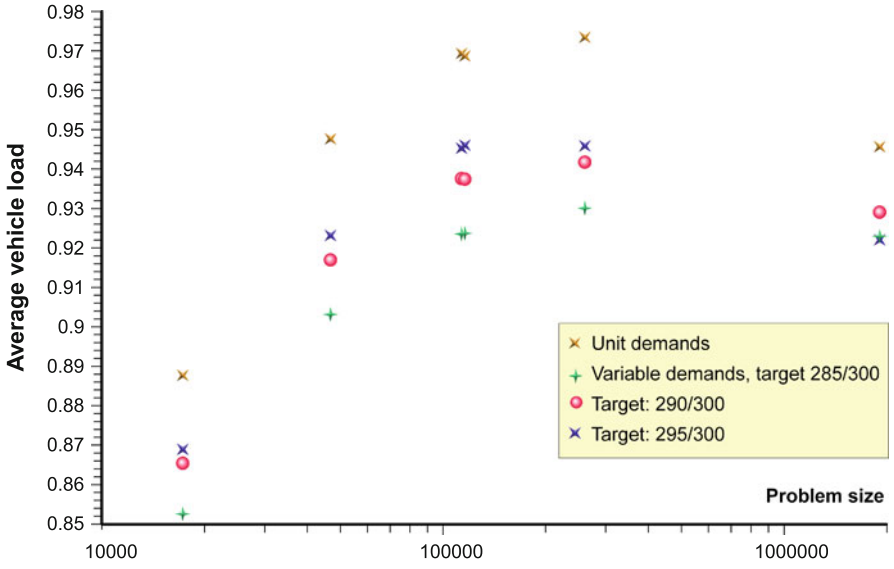

Fig. 6 Sensitivity of final vehicle load with respect to target cluster volume (color figure online)

very few entities, encouraging the opening of depots for single entities. For unit demand instances, the vehicle load is very high, reaching 0.97 for dense instances. For variable demand instances, the vehicle load depends on the target quantity used

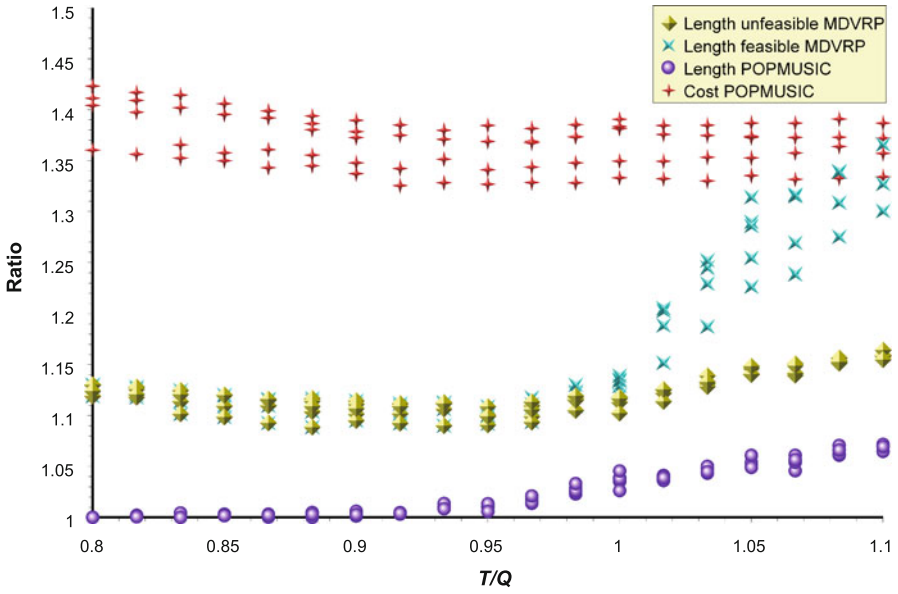

Fig. 7 Influence of target clusters volume on the length and cost of final solution. All problem instances with variable demands and $Q = 300$. Lengths and costs have been divided by the best LRP length observed for each instance (color figure online)

for creating clusters. Again, for dense instances, vehicle loads are almost reaching the target.

Figure 7 analyses the influence of solution length and cost with respect to target volume. For variable demand instances, this figure plots the ratio of the total cost over the best LRP length found for various values of target volume. If target volume $T \leq 0.95\,Q$, there is a loss of volume in the vehicles and their number is higher, implying a higher number of trips to the depots and also a higher number of depots. This tends to increase the final cost even if POPMUSIC starts with a LRP solution with a shorter length.

If target $T \geq Q$, there are more clusters with volume above vehicle capacity, implying a higher number of TSP that are split. The clusters are also less compact. However, we see that POPMUSIC is able to improve bad solutions with a large number of TSP tours that are split and the final cost is relatively insensitive to T value. This suggests that T parameter could be removed (setting $T = Q$) for the problem treated in this article. However, T parameter could be important for other types of problems, for instance if there are fixed costs for using vehicles. For $D = 100{,}000$, we see that the cost for opening the depots corresponds to 30–40 % of the length of the tours. For the final total cost, we see that a target $0.9\,Q \leq T \leq Q$ seems to be a good compromise.

Figure 8 plots the computational time as a function of vehicle capacity Q. In this figure, we see that the increase of computational effort is proportional to Q^3, as expected. Indeed, the average number of entities per tour linearly grows with Q. Since MDRVP subproblems solved in POPMUSIC have a fixed number r of

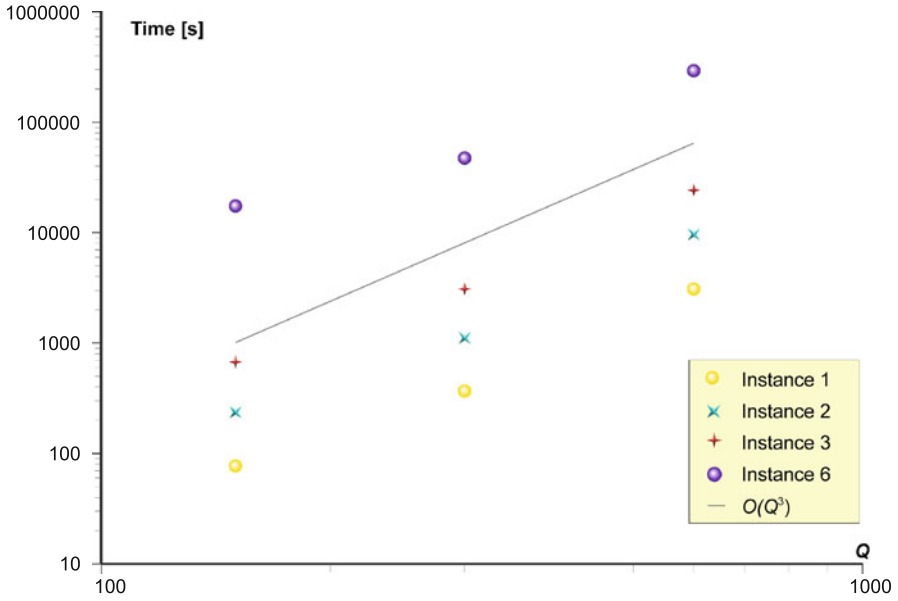

Fig. 8 Evolution of computational effort as a function of vehicle capacity (color figure online)

tours and since the taboo search has a complexity growing with the cube of the number of entities, the observation are coherent with complexity analysis.

Sensitivity with respect to depot opening cost

Since the basis of subproblem building are tours (parts in POPMUSIC terminology), our method seems to be not sensitive to depot opening cost. Indeed, the size of subproblems so defined is independent on D. The only dependency we can observe concerns the computational time needed for merging depots. The larger D is, the higher the number of depots are merged. Figure 9 plots the evolution of computational time of Step 16 of Algorithm 2 as a function of problem size for various D values. We see that the merge procedure takes a computational time growing proportionally to $n^{3/2}$ and is almost independent of D. The picture would certainly have been different for another definition of parts, including all entities attached to a given depot.

Sensitivity with respect to initial solution and improvement procedure

Figure 10 analyses the influence of using a better initial solution or using a better improvement procedure. For getting a better initial solution, we applied a POPMUSIC algorithm after having decomposed the problem into clusters, along the lines of [18]. With such a procedure, CPMD solutions are improved by about 5 %.

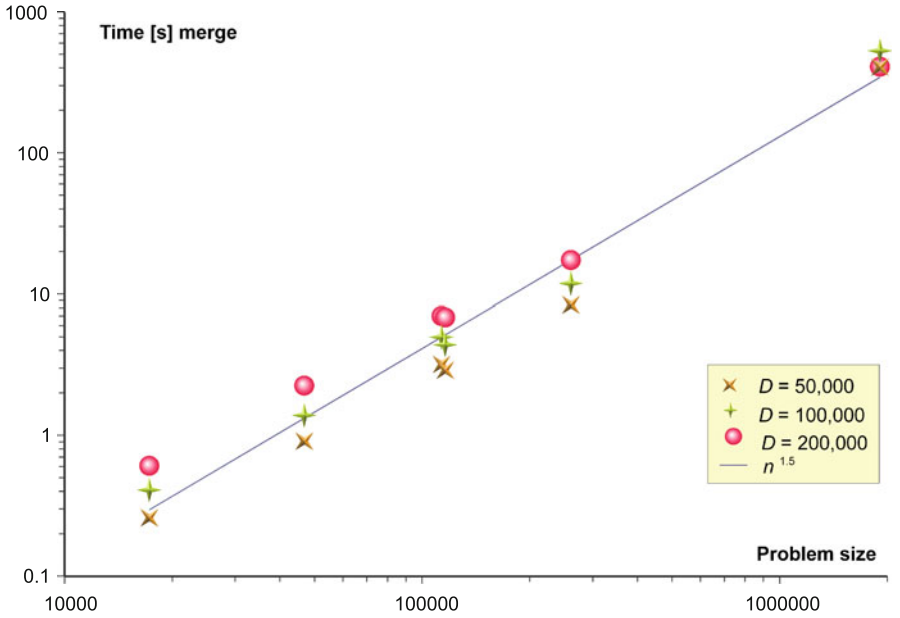

Fig. 9 Influence of computational effort for merging depots for various values of *D* (depot opening cost) (color figure online)

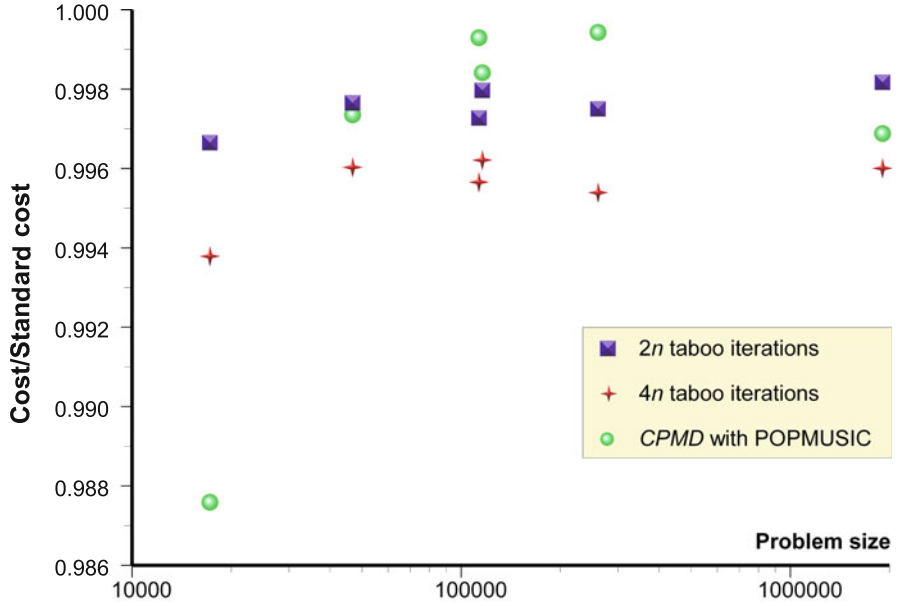

Fig. 10 Influence of initial solution quality and improvement procedure (color figure online)

Table 3 Computational time to get a feasible initial solution, in seconds on one Intel i7 core

Variant	Problem size					
	17,237	46,750	113,193	115,858	260,374	1,904,711
Standard	7	23	68	66	185	3,045
$D = 50{,}000$	7	22	62	64	182	2,984
$D = 200{,}000$	7	24	66	68	184	2,896
$Q = 150$	8	26	82	80	251	5,435
$Q = 600$	10	31	73	76	189	2,320
$Q = 10$	7	23	72	74	233	4,960
$Q = 20$	7	23	63	66	175	2,811
$Q = 40$	10	28	74	75	188	2,156
$t = 2n$	7	22	63	66	178	2,966
$t = 4n$	7	22	63	66	180	2,962
CPMD*	355	1,052	2,705	2,963	5,757	36,635
$T = 285$	7	22	64	69	179	2,895
$T = 295$	8	22	64	66	183	2,931
U: random	7	22	63	64	178	2,820

Table 4 Computational time to improve the initial solution with POPMUSIC, in seconds on one Intel i7 core

Variant	Problem size					
	17,237	46,750	113,193	115,858	260,374	1,904,711
Standard	362	1,093	3,034	3,108	6,712	44,553
$D = 50{,}000$	337	1,000	2,657	2,702	5,977	41,169
$D = 200{,}000$	415	1,257	3,371	3,376	7,289	49,584
$Q = 150$	70	211	593	576	1,356	12,051
$Q = 600$	3,090	9,646	24,212	27,417	54,824	291,605
$Q = 10$	86	250	682	700	1,563	12,514
$Q = 20$	594	1,925	5,432	5,607	11,724	74,712
$Q = 40$	4005	14,896	45,897	46,231	101,841	600,830
$t = 2n$	761	2,226	6,037	6,056	13,253	87,959
$t = 4n$	1548	4,508	11,913	12,110	27,099	176,877
CPMD*	88	331	1,102	1,076	2,799	43,211
$T = 285$	392	1,163	3,117	3,297	6,743	46,824
$T = 295$	423	1,096	2,859	3,013	6,362	45,155
U: random	281	788	1,936	2,056	4,489	32,340

For getting better solutions to sub-problems we multiplied by 2 or by 4 the number of taboo iterations in the improvement procedure (implying computational times multiplied by the same ratio). Figure 10 provides the ratio of the solution cost of the improved versions with the cost of our algorithm with standard parameters.

Table 5 Improvements obtained with POPMUSIC over initial solution, expressed in percent

Variant	Problem size					
	17,237	46,750	113,193	115,858	260,374	1,904,711
Standard	7.3	8.5	9.2	9.3	9.5	9.8
$D = 50{,}000$	6.8	7.3	7.7	7.9	8.3	8.5
$D = 200{,}000$	9.2	9.7	10.3	10.3	10.6	10.7
$Q = 150$	7.4	7.7	8.9	8.6	9.4	9.4
$Q = 600$	9.3	10.1	10.3	10.7	9.8	9.4
$Q = 10$	6	6.1	6.1	6.4	6.2	6.3
$Q = 20$	8.2	9.3	9.3	9.5	8.8	7.6
$Q = 40$	8.8	9.9	10.8	11	10.2	8.1
$t = 2n$	7.8	8.8	9.6	9.6	9.8	10
$t = 4n$	8.2	9.1	9.8	9.9	10.1	10.3
CPMD*	4.9	5.3	5.6	5.8	5.1	4.2
$T = 285$	7.7	9	9.3	9.7	9.3	10.2
$T = 295$	7.8	8.5	9.3	9.6	9.4	9.3
U: random	7.4	8.5	8.9	9	9.3	9.6

Table 6 Absolute value of final cost

Variant	Problem size					
	17,237	46,750	113,193	115,858	260,374	1,904,711
Standard	120,606,754	262,180,587	430,139,462	476,054,191	1,095,478,899	11,971,512,294
$D = 50{,}000$	105,527,279	233,571,406	380,297,149	422,298,538	970,139,834	10,751,806,821
$D = 200{,}000$	139,253,415	300,496,000	496,943,248	548,456,575	1,261,973,725	13,622,702,787
$Q = 150$	149,257,348	336,631,365	568,728,130	627,504,037	1,435,686,930	15,111,422,493
$Q = 600$	104,311,652	221,590,472	353,452,289	392,244,201	905,311,798	10,226,935,742
$Q = 10$	146,590,678	329,349,802	555,585,993	612,466,586	1,399,924,584	14,768,168,944
$Q = 20$	118,581,395	257,852,000	424,244,255	470,534,133	1,079,931,492	11,824,034,476
$Q = 40$	104,962,416	221,425,429	350,645,458	390,351,803	898,740,118	10,137,017,763
$t = 2n$	120,203,176	261,565,818	428,967,967	475,088,562	1,092,747,557	11,949,671,929
$t = 4n$	119,857,545	261,140,192	428,271,733	474,250,824	1,090,435,064	11,923,729,093
CPMD*	119,110,324	261,487,420	429,838,045	475,300,850	1,094,862,284	11,934,236,071
$T = 285$	120,374,633	261,835,995	431,088,578	476,021,788	1,094,107,559	11,921,201,625
$T = 295$	120,982,470	262,103,904	431,304,976	477,655,689	1,099,990,473	12,061,445,385
U: random	120,529,721	262,226,401	430,977,509	476,931,265	1,095,878,122	11,985,158,109

We see that the improvements are very low, generally below 0.5 %, with the exception of the smallest instance for which the improvement reaches 1.2 %. Such an improvement is mainly due to a better CPMD solution that was improved by 10 %. So, it seems that our method is relatively insensitive to initial solution, as soon as the last is of decent quality. The clustering approach is very important for

reaching an appropriate quality. Also, by allowing more iterations to the improvement procedure, the global improvements are moderate. However, we think that better solutions might be obtained by using a taboo search that relocates the depot, i.e. by using an improvement procedure that solves LRP instances rather than MDRVP as we did.

Detailed computational results

As mentioned above, by taking various windows (longitude, latitude) among the entities of the World TSP, six different entities sets have been considered. The standard instances have variable customer demand ($1 \leq q_i \leq 29$), vehicle capacity $Q = 300$ and depot opening cost $D = 100{,}000$. The standard parameter setting has a target value $T = Q - 10$ for variable demand and $T = Q$ for fixed demand. The number of taboo iterations in standard runs is $t = n$ and the set U of unoptimized part is managed as a stack.

For each entities sets, we also used other values for vehicle capacity ($Q = 10$; 20; 40 for unit demand instances and $Q = 150$; 600 for variable demand instances) and depot opening cost ($D = 50{,}000$; 200,000), creating a total of 48 test instances.

In addition, we provide computational results obtained for the standard instances by varying the target capacity parameter ($T = 285$; 295), the number of taboo iterations ($t = 2n$ and $t = 4n$), the improvement of the decomposition into clusters with POPMUSIC technique (CPMD*) and set U management (random choice).

Table 3 provides the computational times for generating the initial solution. Table 4 provides the computational times for improving the initial solution with POPMUSIC. Table 5 provides the improvement of the cost of the solution, expressed in percent, that has been obtained by applying POPMUSIC. Table 6 provides the final cost obtained after POPMUSIC run. This last table shows that POPMUSIC is relatively insensitive to the method parameters: even working much more for getting a better initial solution (CPMD*) has little impact on the final cost. Managing U set randomly tends to speed-up the method a little bit and to produce slightly worse solution, but this is not systematic.

Conclusions

We have shown in this article that POPMUSIC template can be applied to LRP instances of several order of magnitude larger than those commonly treated in the literature. POPMUSIC strategy is able to heuristically solve large instances (more than 10^6 customers) of a location-routing problem. The algorithmic complexity of POPMUSIC depends more on the generation of a decent initial solution than on the optimization of this solution with a local search. This article presents a way to generate such an initial solution by solving approximatively a CPMD. A solution to this problem can be obtained in $O(n^{3/2})$ without using geographical informations other than for computing a distance between two entities. This allows to deal with problem instances including millions of entities.

Research perspectives

There are potential improvements for our method, for instance by using a less basic improvement procedure for solving subproblems, like the iterative searches proposed in [13]. Then, the influence of the definition of proximity between two tours has to be studied, as well as the management policy of the seed-parts. A more elaborated management could allow to parallelize the method. It seems to be possible to solve problems in $O(\sqrt{n})$ using $O(n)$ processors. Then, the efficiency of POPMUSIC template should be studied for problem instances of higher dimension (e.g. adding time windows). For such instances, the definition of distance between two parts of a solution must be revisited. Finally, the approach should be tried on other problems like capacitated or two-echelon LRPs, truck and trailer routing problems with satellite depots or periodic variants of these problems [3, 5, 9, 10, 12, 17].

Acknowledgments First author was partially supported by FAPERJ (Project E-26/110.552/2010) and CAPES (Process 1715-09-7), Brazil. Both authors were partially supported by the Institute for embedded systems, HEIG-VD, project AILSI HES-SO-31022.

References

1. Alvim ACF, Taillard ÉD (2009) POPMUSIC for the point feature label placement problem. Eur J Oper Res 192(2):396–413
2. Barreto S, Ferreira C, Paixão J, Sousa Santos B (2007) Using clustering analysis in a capacitated location-routing problem. Eur J Oper Res 179(3):968–977
3. Boccia M, Crainic T, Sforza A, Sterle C (2010) A metaheuristic for a two echelon location-routing problem, Experimental algorithms. In: Paola F (ed) Lecture notes in computer science, vol 6049. Springer, Berlin, pp 288–301
4. GRS 80. http://en.wikipedia.org/wiki/grs80. Last visited on Nov 04, 2011
5. Jacobsen SK, Madsen OBG (1980) A comparative study of heuristics for a two-level routing-location problem. Eur J Oper Res 5(6):378–387
6. Li F, Golden BL, Wasil EA (2005) Very large-scale vehicle routing: new test problems, algorithms, and results. Comput OR 32:1165–1179
7. Nagata Y, Bräysy O (2008) Efficient local search limitation strategies for vehicle routing problems, EvoCOP. In: Jano I van Hemert, Carlos C (eds) Lecture notes in computer science, vol 4972. Springer, Berlin, pp 48–60
8. Nagy G, Salhi S (2007) Location-routing: issues, models and methods. Eur J Oper Res 177(2):649–672
9. Nguyen VP, Prins C, Prodhon C (2012a) A multi-start iterative local search with tabu list and path relinking for the two-echelon location routing problem. Eng Appl Artif Intell 25(1):56–71
10. Nguyen VP, Prins C, Prodhon C (2012b) Solving the two-echelon location routing problem by a hybrid GRASP x path relinking complemented by a learning process. Eur J Oper Res 216(1):113–126
11. Prins C, Prodhon C, Wolfler Calvo R (2006a) A memetic algorithm with population management (MA|PM) for the capacitated location-routing problem, EvoCOP. In: Jens G, Günther RR (eds) Lecture notes in computer science, vol 3906. Springer, Berlin, pp 183–194
12. Prins C, Prodhon C, Wolfler Calvo R (2006b) Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. 4OR 4(3):221–238
13. Prins C, Prodhon C, Ruiz AB, Soriano P, Wolfler Calvo R (2007) Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. Transp Sci 41(4):470–483
14. Taillard ÉD (1993) Parallel iterative search methods for vehicle routing problems. Networks 23:661–673
15. Taillard ÉD (2003) Heuristic methods for large centroid clustering problems. J Heuristics 9:51–73

16. Taillard ÉD, Voss S (2001) POPMUSIC: partial optimization metaheuristic under special intensification conditions. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer Academic Publishers,New York, pp 613–629
17. Villegas JG, Prins C, Prodhon C, Medaglia A, Velasco N (2010) GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. Eng Appl Artif Intell 23(5):780–794 Advances in metaheuristics for hard optimization: new trends and case studies
18. Waelti P, Mautor T, Taillard ÉD (2003) Application de méta-heuristiques au problème de la p-médiane. In: ROADEF conference, Avignon
19. World TSP. http://www.tsp.gatech.edu/world/index.html. Last visited on Nov 04, 2011
20. Zachariadis EE, Kiranoudis CT (2010) A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. Comput OR 37(12):2089–2105