Transportation Research Part C 54 (2015) 110–130

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios

Santiago Muelas [a,*], Antonio LaTorre [b], José-María Peña [b]

[a] Center for Biomedical Technology, Universidad Politécnica de Madrid, Spain
[b] Facultad de Informática, Universidad Politécnica de Madrid, Spain

ARTICLE INFO

Article history:
Received 8 August 2014
Received in revised form 15 December 2014
Accepted 11 February 2015
Available online 27 March 2015

Keywords:
Variable neighborhood search
Dial-a-ride problem
Transportation
Demand-responsive transport
Distributed algorithms

ABSTRACT

These days, transportation and logistic problems in large cities are demanding smarter transportation services that provide flexibility and adaptability. A possible solution to this arising problem is to compute the best routes for each new scenario. In this problem, known in the literature as the dial-a-ride problem, a number of passengers are transported between pickup and delivery locations trying to minimize the routing costs while respecting a set of prespecified constraints. This problem has been solved in the literature with several approaches from small to medium sized problems. However, few efforts have dealt with large scale problems very common in massive scenarios (big cities or highly-populated regions). In this study, a new distributed algorithm based on the partition of the requests space and the combination of the routes is presented and tested on a set of 24 different scenarios of a large-scale problem (up to 16,000 requests or 32,000 locations) in the city of San Francisco. The results show that, not only the distributed algorithm is able to solve large problem instances that the corresponding sequential algorithm is unable to solve in a reasonable time, but also to have an average improvement of 9% in the smaller problems. The results have been validated by means of statistical procedures proving that the distributed algorithm can be an effective way to solve high dimensional dial-a-ride problems.

1. Introduction

Nowadays, cities are demanding flexible transportation services for the design of modern mobility services, ad hoc logistic and delivering and reactive emergency transportation, to name a few. Traditional (public) transportation has shown its limitations to offer an adaptive system that could, for example, invest more resources in certain areas if a particular event could affect considerably the user transportation demands. On the other hand, the higher economic costs of classic individual transportation services (such as taxi services) make them not affordable in many cases.

Demand-responsive transport (DRT) has appeared as an alternative to traditional services by providing a flexible transport model that can adapt to the dynamic nature of the scheduling of routes based on user's demands. It uses, in general, medium-sized vehicles shared by several requests where the routes of the vehicles are dynamically computed in order to optimize the total cost, client satisfaction, pollution emissions, etc. DRT offers solutions not only for passengers mobility but also in the fields of medical transportation services and logistics (Xu and Huang, 2009).

* Corresponding author.
E-mail addresses: smuelas@fi.upm.es (S. Muelas), atorre@fi.upm.es (A. LaTorre), jmpena@fi.upm.es (J.-M. Peña).

http://dx.doi.org/10.1016/j.trc.2015.02.024
0968-090X/© 2015 Elsevier Ltd. All rights reserved.

The practical application of large-scale DRT services has the challenge to provide efficient solutions to the demands over large cities or areas. In the literature, a transportation problem with these particular characteristics, the dial-a-ride problem (DARP) has previously been studied and analyzed. In the DARP, the objective is to determine the best routing schedule for a group of vehicles to satisfy the transportation requests of a number of customers. Each request consists of a specific pickup (origin) and delivery (destination) locations along with a desired departure or arrival time as well as the number of items (passengers or packages) to be transported. The items have to be transported to their destinations but not necessarily directly since they could share a ride. The problem takes into account the customer's satisfaction, expressed in terms of additional constraints. The DARP can be proven to be NP-hard. The proof is based on the related NP-hard traveling salesman problem with time windows, into which the DARP can be transformed. Solutions for this type of problem have become increasingly popular (Cordeau et al., 2007; Horn, 2002; Xu and Huang, 2009) from the development of ambulatory services for aging people, to shuttle and merchandise transportation services for organizations. However and as far as the authors are concerned, no other study has tried to solve considerably large-scale instances (i.e. >10,000 requests).

In this study, we present a new distributed and parallel variable neighborhood search (VNS) algorithm that is able to solve high dimensional DARPs by dividing the search space into partitions to allow its exploration in parallel while exchanging the information among the computing nodes to improve the optimization of the search process. The VNS is a heuristic optimization algorithm that has been successfully used with similar problems, obtaining competitive results in all the studies (Carrabs et al., 2007; Parragh et al., 2009, 2010). In Muelas et al. (2013) we proposed a VNS algorithm for solving several demanding scenarios in a large city that an on-demand transportation company could face. Several instances from 100 to 1000 requests were studied, in which the proposed algorithm obtained the best overall results. In the study it was shown the importance of the initialization procedure when facing large-scale and demanding scenarios. In fact, for the 1000 requests instances, the quality of the solution generated by the algorithm was completely conditioned by the starting solution, being unable to obtain a feasible solution when this solution was created in a completely random way. However, as will be shown in this work, the complexity of the initialization method used, as well as some of the best initialization procedures of the literature (Diana and Dessouky, 2004; Potvin and Rousseau, 1993), makes them inappropriate to solve large-scale scenarios. Therefore, in this study we propose a new distributed approach that is able to intelligently divide the search space and explore it by means of the aforementioned VNS algorithm in parallel with the purpose of obtaining feasible and optimized solutions in a reasonable period of time.

The remainder of this paper is structured as follows. Section 2 presents an overview of the literature of the DARPs and vehicle routing problems (VRPs). Section 3 defines the main problem and evaluation function. In Section 4 the details of the new proposal are presented. Section 5 describes the experimental scenario in depth. In Section 6 the results obtained are thoroughly analyzed an discussed. Finally, Section 7 contains the concluding remarks obtained from this work.

2. Related work

The DARP belongs to a more general group of problems referred to as vehicle routing problems with pickups and deliveries (VRPPD) where goods are transported with a fleet of vehicles between pickup and delivery locations. This class of problems is divided into two subclasses depending on whether the pickup and delivery locations are paired or not:

- If the pickup and delivery locations are unpaired, each picked up item can be transported to any delivery location. Depending on the number of vehicles used, two subclasses can be identified: pickup and delivery traveling salesman problem (PDTSP) for the single vehicle case and pickup and delivery vehicle routing problem (PDVRP) for the multiple vehicles problem.
- In the opposite case, each pickup item at a specific location must be delivered to its associated delivery destination. Here, we can find the classical pickup and delivery problem (PDP) and the DARP. Both problems deal with the optimization of a number of requests in which each request specifies the number of items that must be transported from an origin to a destination. The main difference between these two problems is that the PDP is focused in transporting goods whereas the DARP deals with the transportation of passengers. This difference is usually expressed by the addition of constraints such as time windows, route duration and ride time violations.

Depending on the nature of the planning process each transportation problem can be identified as static or dynamic. In the static DARP, the objective is to define the routes that are going to attend the requests. In the dynamic problem, a solution of partial routes has been previously constructed (for example by means of a static algorithm) and new requests have to be inserted in real time. In this article the static version of the DARP has been considered.

The DARP has been extensively studied in the literature. The first publications in this area date from the late 1960s and early 1970s (Rebibo, 1974; Wilson et al., 1967, 1971). Since then, several approaches have been studied for solving this problem.

Due to the high computational demand of the exact methods, in particular on large-scale problems, several heuristic methods have been proposed for dealing with the DARP (Borndörfer et al., 1997; Cullen et al., 1981). Metaheuristics are also a common approach when dealing with the DARP (Cordeau and Laporte, 2003; D'Souza et al., 2012; Jorgensen et al., 2006; Parragh et al., 2010). In general, heuristic methods tend to run faster whereas metaheuristics usually outperform basic heuristic procedures in terms of solution quality.

Although the DARP has been greatly analyzed in the past, the authors have not found any study that tried to solve large-scale instances (>10,000 requests). In fact, most of the studies of the general VRP domain have dealt with considerably smaller problems, having, in most of the cases, solved problems instances of a few thousand requests at most (Gendreau, 2010; Li et al., 2005, 2007; Vidal et al., 2013).

When dealing with problems of this magnitude, one of the main problems that arises is the long execution time that the algorithms need to obtain a solution. Since a transportation company needs to generate the results in a small number of hours, it is of vital importance to increase the speed of the base algorithm. Parallelization is one of the common techniques that has been used in several domains to increase the performance of the algorithms. However, as Crainic mentioned in Crainic (2008) "*Very few efforts have been dedicated to developing parallel algorithms for VRP and its variants*". In fact, as far as the authors are concerned, this is the first work that proposes a parallel approach for solving the static DARP. Likewise, few efforts have been developed to parallelize the VNS algorithm. Most of the common approaches are presented in Davidović and Crainic (2012) and are mainly based on: (i) parallelizing the primary perturbators: the local search (LS) or the shaker operators or (ii) conducting several searches in parallel and selecting the best result from all of them.

With regards to parallelization in the VRP domain, Schulze characterizes parallelism in three different ways (Schulze and Fahle, 1999): a master/slave approach where the slave nodes explore the neighborhood and send their solutions to the master node which, in turn, selects the one that leads to the best solution of the whole neighborhood; a second approach in which the problem is decomposed into independent subproblems, each being solved by a different node, and where the synchronization occurs when the subproblems are merged by the master processor to obtain a solution; and a third approach, where each node is started with a different initial solution or use a different set of search parameters and periodically, the information is shared between the threads to improve the overall search of the heuristic. Some examples of these approaches are described below.

Ochi et al. (1998) propose a parallel island model genetic algorithm (GA) for a heterogenous fleet VRP. Here each node executes a GA over an independent population that contains complete solutions of the problem that are exchanged between the nodes whenever a convergence criterion is satisfied (up to a maximum number of migrations). In Berguer and Barkaoui (2004) a parallel version of a hybrid GA was developed for a VRP with time windows. The algorithm evolves concurrently two populations with different objectives where a master node manages the high-level execution of the algorithm and the slave nodes explore the neighborhood by means of different genetic operators. In Allahviranloo et al. (2014) three different parallel GA were proposed for dealing with selective VRPs under uncertainty. In the first algorithm every node evolves an independent population until it converges and then all of them are aggregated to make a final population to implement another GA on top of it. In the second strategy, at every iteration, the best chromosome of each node is copied to all other nodes and the repeated chromosomes are eliminated at the end of every iteration. The third strategy follows the same approach but does not eliminate the repeated chromosomes. From the three strategies, the third algorithm obtained on average the best results but had a major drawback in its long computation time. Finally, in Hyytiä et al. (2012) a parallel queuing approach from the framework of Markov decision processes is proposed to solve a dynamic DARP which is a different version of the static DARP where the existing routes (previously computed) are modified in real-time in response to new requests. In this work, each new petition is evaluated according to a control policy to determine the vehicle (represented by a queue) that can better satisfy it. The evaluations are computationally lightweight and therefore suitable for DARP systems dealing with hundreds of vehicles.

Since the objective of this study is to develop an algorithm that is able to cope with large-scale instances of the DARP, we have chosen a hybrid method that uses the second approach from Schulze's characterization for decomposing the problem into several subproblems and some ideas of the third approach: the periodic sharing of information to improve the overall search.

The reasons for decomposing the problem are based on the fact that the VNS algorithm that we are using contains two operators, the LS and the initialization method that, though of vital importance to improve the quality of the solutions, have a complexity of $O(n^3/m)$ where n corresponds to the number of requests and m to the number of vehicles. Thus, it is clear that, by using Schulze's second approach: decomposing the problem in several independent problems, the algorithm can obtain the greatest improvements in terms of speed ($O\left((n/k)^3/m\right)$) when compared with approaches from the first or third of Schulze's methods which only explore the neighborhood in parallel and therefore have a considerable worse complexity ($O((n^3/m)/k)$) since their aim is to improve the quality of the solution but not to reduce the computation time for obtaining a solution. It must be taken into account that most of the methods defined in Schulze's characterization were not designed for solving large scale problems where the computational time must be carefully considered when proposing an algorithm to solve them.

As will be detailed later, for our proposal we have chosen a partitional clustering approach where the set of initial requests is divided into independent subsets being each subset assigned to a computing node. The decomposition approach has been combined with the information sharing of the third approach with the idea of improving the exploration capabilities of each of the nodes to give them more possibilities to improve the routes generated by each of them. All the details of this approach are thoroughly described in Section 4.

3. Problem definition

For this work, we have used the formulation of the problem described in (Muelas et al., 2013; Parragh et al., 2010). Therefore, the static DARP is defined on a complete graph $G = (V, A)$ where $V = v_0, v_1, \ldots, v_{2n}$ is the set of all the vertices and A the set of all the arcs. For each arc (i,j) a non-negative travel time $t_{i,j}$ is considered. The transportation cost is supposed to be proportional to the travel time. Therefore, for computing the total cost of the routes, the travel times have been used. Each customer's request is made up of a pickup and a delivery vertex pair $\{i, i+n\}$ that have to be served by m vehicles with a capacity of Q. Since it is supposed that the management of the vehicles is carried out by an external company, there is no need to optimize the route from the central depot to the first pickup vertex. Therefore, it is assumed that the vehicles start the associated route at the location of the first vertex. At each pickup vertex, a number of passengers $(q_i > 0)$ associated to a request are carried in the vehicle whereas, at the associated delivery vertex, the same number of passengers leave the vehicle $(q_{i+n} = -q_i)$. The vehicle load when leaving a specific vertex is represented by y_i. Each request represents a client or a group of clients (e.g. a family, a tourist group) with the same constraints, i.e., with the same pickup time, origin and destination. If there are several passengers with some constraints in common but with a particular difference (e.g. a different pickup time), they will not be assigned to the same request but to different requests.

Since this study is focused in optimizing the problem that comes from for a public on-demand service company, all the modeled requests fall into the inbound category, i.e., they all have a tight time window on the origin $[e_i, l_i]$ where e_i is requested by the user and $l_i = e_i + P$, being P the maximum pickup time that a passenger should wait. It is important to remark that windows correspond to soft time windows since the problem allows to violate the maximum pickup time, although with an assigned penalty that increases proportionally to the magnitude of the violation, as will be shown later.

Leaving vertex i at its corresponding departure time (D_i) results in arriving at the subsequent vertex j at the arrival time $A_j = D_i + t_{i,j}$. The beginning of the departure of the following vertex D_j cannot start before the beginning of the respective time window, i.e., $D_j = max\{A_j, e_j\}$. Therefore, a vehicle could have a waiting time at vertex j of $W_j = D_j - A_j$. The difference between the requested pickup time and the computed departure time at vertex i, i.e., the user waiting time that should not exceed P, is defined by $P_i = D_i - e_i$.

The ride duration of a client, i.e. the time a client spends on board the vehicle, corresponds to $L_i = A_{i+n} - D_i$. Similarly to the pickup time constraint, there is a maximum user ride duration constraint $(Lmax_i)$ that has to be respected. To compute the maximum ride time, a relative approach is followed where the ride time of the pickup and delivery vertices of a request is multiplied by a constant $(Lmax_i = t_{i,i+n} * maxridetime_c)$. This way, the maximum ride time is set by taking into account the duration of the minimum ride time between the origin and the destination. Finally, the time window at the destination can be automatically computed by the following equations $e_{i+n} = e_i + t_{i,i+n}$ and $l_{i+n} = l_i + Lmax_i$. This notation is summarized in Table 1 and represented graphically in Fig. 1.

Table 1
Problem notation.

e_i	Beginning of time window at vertex i
l_i	End of time window at vertex i
m	Number of Vehicles
$maxridetime_c$	constant used for computing the maximum user ride time
n	Number of requests
q_i	Number of passengers picked up at vertex i
$t_{i,j}$	Travel time from vertex i to vertex j
y_i	Load when leaving vertex i
A_i	Arrival time at vertex i
D_i	Departure time from vertex i
$Lmax_i$	Maximum user ride duration of vertices $i, i+n$
L_i	Ride duration of vertices $\{i, i+n\}$
P	Maximum pickup waiting time
P_i	Difference between the requested pickup time (e_i from the pickup vertex) and the departure time
Q	Maximum capacity of the vehicles used
s	A solution (routing plan)
W_i	Vehicle waiting time at vertex i
$c(s)$	The transportation cost of the solution s
$w(s)$	The pickup time violation of the solution s
$r(s)$	The ride duration violation of the solution s
$q(s)$	The load violation of the solution s

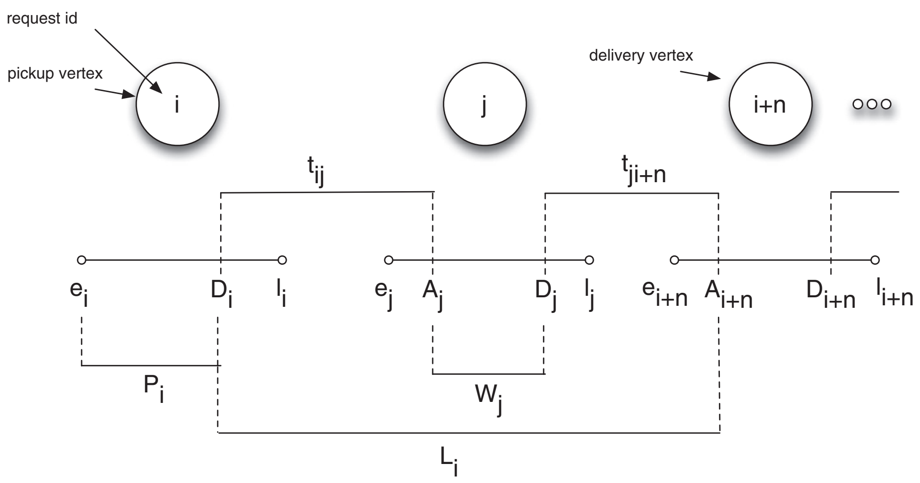

Fig. 1. Representation of the problem notation.

3.1. Evaluation function

As previously mentioned, for this work we have focused on minimizing the total cost of the routes proposed to solve the problem, i.e., $c(s) = \sum_{(i,j)\in s} t_{ij}$. The final evaluation function, described in Eq. (1), takes into account this value as well as the total violations of pickup time, ride duration and load. Pickup time violation is computed as $w(s) = \sum_{i=1}^{i=n}(D_i - l_i)^+$ where $x^+ = max\{0, x\}$. Ride duration violation is computed as $r(s) = \sum_{i=1}^{i=n}(L_i - Lmax_i)^+$ and load violation as $q(s) = \sum_{i=1}^{i=2n}(y_i - Q)^+$. The penalty terms for these violations are given by α, β and γ.

$$f(s) = c(s) + \alpha w(s) + \beta r(s) + \gamma q(s) \tag{1}$$

For the optimal computation of the A_i, W_i and D_i values we have followed the evaluation scheme used in (Cordeau and Laporte, 2003; Muelas et al., 2013; Parragh et al., 2010). With this scheme, the route duration is minimum and ride time limits are respected whenever is possible. In contrast to the algorithm of Parragh et al. (2010) the penalty terms α, β and γ are fixed and set to considerable large values so that no violations are permitted for candidate solutions. We have observed that by allowing solutions that violate any constraint during the search with high dimensional problems, the algorithm is more attracted to local optima of lower quality. This phenomena is shown in the following sections.

4. Contribution

As mentioned in the previous sections, in this paper we propose a new distributed version of the VNS algorithm presented in Muelas et al. (2013) for solving large-scale instances of a dial-a-ride problem. The main idea of this algorithm is to intelligently decompose the initial set of n requests into k independent subsets[1] so that each *computing node* can create and optimize its corresponding set of independent routes using the sequential VNS algorithm from Muelas et al. (2013). To improve the quality of the generated routes, with a frequency *freq* (i.e. each *freq* seconds of the execution), the routes are united at a *central node* and reassigned to the computing nodes based on the similarity between them in what has been called a redistribution phase. The central node is a computing node designated as such at the beginning of the execution[2] that, in addition to optimize its assigned set of requests as any other computing node, is in charge of distributing the set of routes among all the remaining computing nodes at the beginning of the execution and at each redistribution phase (determined by the *freq* parameter).

The whole algorithm has been implemented with the C++ programming language and parallelized by using the OpenMPI library. The main scheme of the distributed algorithm is represented in Algorithm 1 and each of its main parts is described in the following subsections.

[1] being k the number of available nodes.
[2] in our case, the first computing node determined by the OpenMPI library.

Algorithm 1. Distributed VNS algorithm

```
 1: while Stopping criterion is not satisfied do
 2:   Compute in parallel the distance matrix used by the k-medoids algorithm
 3:   Create a partition of $k$ clusters from the original set of requests using the k-medoids algorithm
 4:   Distribute each cluster to the corresponding computing node
 5:   Each computing node executes the sequential VNS algorithm from Muelas et al. (2013) independently using its set of requests
 6:   if execution time $mod\ freq = 0$ (redistribution phase) then
 7:     if the computing node is the central node then
 8:       Receive the routes from all the computing nodes and join them with the central node routes to obtain a global set of routes
 9:       Distribute the set of routes into $k$ different sets of routes
10:       Send each set of routes to the corresponding computing node
11:     else
12:       Send the routes to the central node
13:       Receive the new set of routes from the central node
14:     end if
15:     Continue the execution
16:   end if
17: end while
18: if the computing node is the central node then
19: Receive the routes from all the computing nodes and join them with the central node routes to obtain the final solution
20: else
21:   Send the routes to the central node
22: end if
```

4.1. Distributing the requests

The first step of the algorithm proposed in this study, and one of the key elements of its performance, is to decompose the initial set of requests into k independent subsets. Instead of conducting this partition randomly, we have used the k-medoids algorithm to create a tessellation of the requests where each partition contains requests that are similar, according to a defined metric, among them. The proposed hypothesis is that by grouping similar requests, the nodes are able to construct better routes than by conducting a uniform approach where each request has the same probability of being assigned to each of the partitions.

There are several studies in the literature that have followed the partitional approach to VRP related problems being the clustering approach one of the most popular (Dondo and Cerdá, 2007; Ganesh and Narendran, 2007; Matis and Kohani, 2011; Qi et al., 2012). Moreover, when it comes to clustering of large data sets, the k-medoids algorithm has been recommended in the past as one of the best approaches (Lucasius et al., 1993).

The k-means algorithm is one of the most frequently used algorithms within the partitional clustering methods. It has the benefits that is easy to implement and relatively fast to execute. For other distributed metaheuristics and problems, a similar approach, where closer points were assigned to the same partition, was able to significantly improve the final results (Muelas et al., 2010). Given a set of observations $(x_1, x_2, \ldots, x_n)$ where each observation corresponds to a d-dimensional numerical vector, the k-means clustering aims to partition the n observations into $k(\leqslant n)$ sets $S = S_1, S_2, \ldots, S_k$ so as to minimize the within-cluster sum of squares, i.e., its objective is to find $argmin_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$ where μ_i corresponds to the centroid, i.e., the representative of the cluster computed as the mean of all the points in S_i.

For the proposed algorithm we have selected the k-medoids algorithm that has the same philosophy as the k-means algorithm but uses the elements of the set as the representatives of the cluster instead of computing each centroid. In particular, we have used the partitioning around neighbors (PAM) algorithm one of the most common implementations of the k-medoids clustering. This algorithm is defined as follows:

Let Θ be the set of medoids, I_Θ the set of indices of the points in X that compose Θ and $I_{X-\Theta}$ the set of indices of the points that are not medoids. The quality of the clustering created by a given set Θ of medoids is obtained through the following function:

$$J(\Theta, U) = \sum_{i \in I_{X-\Theta}} \sum_{j \in I_\Theta} u_{ij} d(x_i, x_j) \tag{2}$$

and

$$u_{ij} = \begin{cases} 1 & \text{if } d(x_i, x_j) = min_{q \in I_\Theta} d(x_i, x_q) \quad i = 1, \ldots, N \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Therefore to obtain the set of medoids Θ that best represents the data set, $X, J(\Theta, U)$ needs to be minimized. For this task, PAM uses an optimization procedure described below. Before proceeding to the description of the algorithm, some definitions are necessary. Two sets of medoids Θ and Θ', each consisting of k elements, are called neighbors if they share $k-1$ elements. Thus, the number of neighbors a set $\Theta \in X$ with k elements can have is $k(N-k)$. Moreover, let Θ_{ij} represent the neighbor of Θ that results if $x_j, j \in I_{X-\Theta}$ replaces $x_i, i \in I_X$. Finally, let $\Delta J_{ij} = J(\Theta_{ij}, U_{ij}) - J(\Theta, U)$.

PAM starts with a set Θ of k medoids, which are randomly selected out of X. Then, among all $k(N-k)$ neighbors $\Theta_{ij}, i \in I_\Theta, j \in I_{X-\Theta}$, of the set Θ, we select $\Theta_{qr}, q \in I_\Theta, r \in I_{X-\Theta}$ with $\Delta J_{qr} = min_{ij} \Delta J_{ij}$. If ΔJ_{qr} is negative, then Θ is replaced by Θ_{qr} and the same procedure is repeated. Otherwise, if $\Delta J_{qr} \geqslant 0$ the procedure has reached a local minimum and terminates. Once the algorithm has finished, each $x \in X - \Theta$ is assigned to the cluster represented by the closest medoid.

The k-medoids algorithm is more robust to noise and outliers as compared to k-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared distances (Singh and Chauhan, 2011). Moreover, since it does not generate new centroids like the k-means algorithm, the distance matrix can be initially computed for each unique pair of requests and used in the following steps.

In the proposed algorithm the computation of the distance matrix is carried out in parallel to reduce the computation time. Once the distance matrix is computed, the k-medoids algorithm is executed in the central node (it could be any computing node but we have used the first node according to the MPI deployment) to obtain the clusters and send them to the remaining nodes. Then, each computing node (including the central node) starts to construct their corresponding solution by using m/k routes, where m represents the total number of vehicles or routes and k the number of computing nodes.

4.1.1. Computing the distance between two requests

One of the key elements to correctly apply the k-medoids algorithm is to define an appropriate distance measure between requests. It must be taken into account that the DARP differs from other VRPs in the fact that each request contains two positions: an origin and a destination. Therefore, to properly measure the distance between two requests a metric that takes into account these two locations needs to be determined. We have defined this new metric as the minimum distance between the four possible ways to combine the vertices of two requests, i.e., given two requests R_i and R_j with their corresponding pickup vertices: i and j and delivery vertices: $i+n$ and $j+n$, the new distance measure is computed as follows:

$$\begin{aligned} distance(R_i, R_j) = min(&d(i,j) + d(j, i+n) + d(i+n, j+n), d(i,j) + d(j, j+n) + d(j+n, i+n), d(j,i) + d(i, i+n) + d(i \\ &+ n, j+n), d(j,i) + d(i, j+n) + d(j+n, i+n)) \end{aligned} \tag{4}$$

This way, the new measure takes into account the four different positions of the requests by trying to compute the shortest route for traversing all the positions. Moreover, this measure satisfies the metric conditions, including the triangular inequality, required by the k-medoids algorithm to operate properly (Baraty et al., 2011).

To understand the behavior of the distributed assignment, Fig. 2 depicts the set of routes that are created by each node based on the initial assignment of requests. We have selected a small number of requests and a problem where

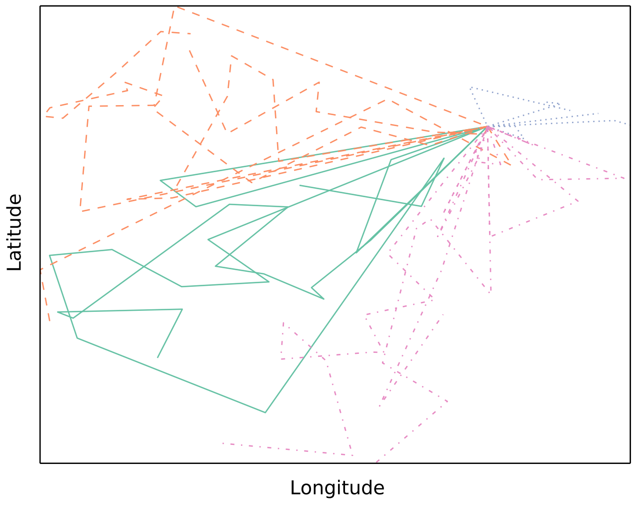

Fig. 2. Example of the constructed routes after the k-medoids distribution. The routes that belong to a node are represented with the same color and line style.

the pickup location for all the requests is fixed at a specific location to facilitate the visualization of the partition. This Figure represents the set routes that are initially constructed by each node after the distribution of requests takes place and each node construct its own set of routes. As shown in the example and explained in the following sections, the k-medoids algorithm obtains the best division in clusters of the data that it can obtain without taking into account if clusters are balanced in terms of size. This can potentially create some problems when the objective is to maximize the performance of a distributed system and will be handled with the modification proposed in the following sections.

4.2. Redistribution of routes

Without any exchange of information, each node would construct its corresponding routes, optimize them until the stop criterion is satisfied, and return them to construct the final solution. However, in order to increment the exploration ability of the overall solution, all the routes are sent to a central node for their posterior redistribution with a certain frequency. Once received, the central node creates k groups having each one m/k routes which are then assigned to each of the nodes of the system.

Similarly to the initialization phase, the k-medoids algorithm is used to create the assignation of the routes groups to nodes. However, since in this case we have to cluster routes, i.e., groups of requests instead of single requests, a new distance measure between routes has been used. There are several proximity measures between groups defined in the literature: single linkage, complete linkage, group average, ward, etc. Everitt et al. (2011) and Kaufman and Rousseeuw (2005). In Barreto et al. (2007) an extensive study was conducted for a capacitated location-routing problem, concluding that the group average measure, i.e. $d(A,B) = \frac{\sum_{I \in A; J \in B} d(I,J)}{|A||B|}$ produced the most balanced results.[3] Therefore, the k-medoids algorithm uses this measure for creating the clusters between the set of routes. After the k-medoids algorithm has finished, each cluster of routes is randomly assigned to each computing node and sent to continue its execution.

4.2.1. Balanced distribution

Analyzing the preliminary results, we observed that, in several cases, the algorithm was creating heavily unbalanced clusters where certain nodes contained most of the requests as the one depicted in Fig. 2. Since this result could bias the exploration of the algorithm, we developed another route distribution algorithm where the k-medoids step is also performed but only to obtain the best distribution of medoids. With this distribution, the requests are equitably distributed between each cluster based on their distance to each medoid so that at the end each cluster contains the closer n/k requests. This approach was also followed for the distribution of routes creating two methods for distributing the routes (besides the simple uniform random approach).

4.3. Analysis on a simple problem

Before presenting the experimental scenario and the corresponding results, we present an analysis of the behavior of the algorithm on a simple scenario of 10 requests for the carnaval problem (detailed in the following section). Here we compare the performance of 25 executions of the sequential VNS algorithm against the results of the proposed distributed algorithm with 2 nodes and migrating ten times during each execution.[4] Table 2 presents the results of the average value obtained from 25 executions for both algorithms as well as the relative difference to the global optimum, obtained by an exact procedure, and the number of times where the global optimum was reached. Fig. 3 depicts the evolution of the average value of the evaluation function for both algorithms as well as the global optimum value.

In these results, it is shown how the sequential VNS algorithm is able to find the global optimum after a few iterations in most of the executions (22 out of 25), proving the robustness of its performance and its adequacy for smaller DARPs. Due to the partition of the search space, the proposed distributed VNS algorithm has less information to perform a global optimization. Therefore, it has more difficulties to construct the initial solution and needs the migration phases to be able to improve the global search and carry on significant improvements to the solution. As already commented, the objective of the distributed VNS is not to improve the performance of the sequential VNS but to be able to optimize large scale DARPs in a reasonable time. In any case, the average difference of the solutions obtained by the distributed algorithm is only a 3% worse than the global optimum and, as will be shown in the following section, the partition of the search space has a beneficial effect in larger and more complex problems where it helps the algorithm to avoid local optima that attract the sequential VNS and, therefore, to improve considerably the quality of the final solution.

[3] which was also confirmed by the results we obtained on a small set of preliminary tests.
[4] the equivalent criterion of the experimental procedure described later.

Table 2
Mean results of 25 executions of the distributed and sequential VNS algorithms for the carnaval problem with 10 requests. The global optimum has an evaluation function value of 3896.

Alg.	Avg. eval. func. value	Rel. diff. to global opt. (%)	#Global opt. found
dVNS	4008.75	2.89	4/25
VNS-seq	3903.04	0.17	22/25

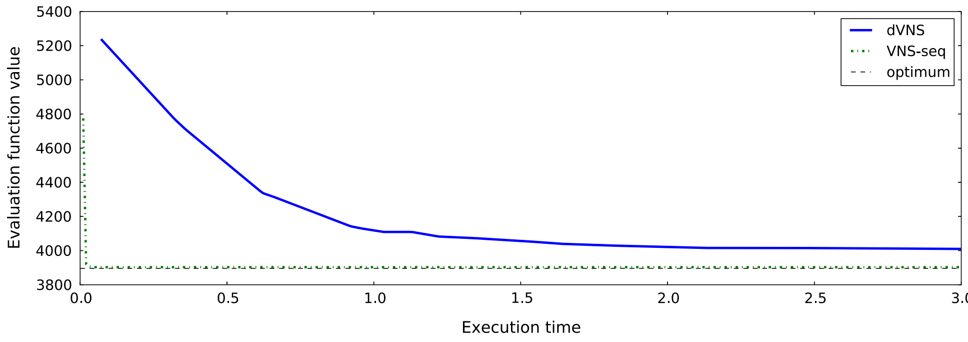

Fig. 3. Evolution of the average value of the evaluation function per execution time (in s) for the carnaval problem with 10 requests.

5. Experimentation

In this section, we describe in detail the selected problem instances as well as the tuning method for selecting the parameter values.

5.1. Problem instances

For the experimentation we have used an extended version of the same benchmark proposed in Muelas et al. (2013) to deal with high dimensional problems. In particular, four different problems, having two instances per problem, have been proposed. These problems represent different types of scenarios that have been synthetically generated taking into account real distance costs obtained for the city of San Francisco, and using the same statistical distributions proposed in Muelas et al. (2013) which are described below.

For each instance, three different large-scale scenarios have been proposed: 4000, 8000 and 16,000 requests. These dimensionalities correspond to 8000, 16,000 and 32,000 locations in the classic VRP-based problems. Therefore, a total of 24 different instances have been optimized for this study.[5] The main characteristics of each of the proposed problems are described below:

- carnaval problem (C): This problem represents the demand that could be generated during the San Francisco Carnaval Festival. In this problem two types of demands are simulated: the requests that could arise from a set of global locations to several stops around the Carnaval area and the requests that could demand a transport from the Carnaval area stops to the global locations. Two uniform distributions for each type of demand have been used: one using the time range 10:30–17:00 for the requests that go to the Carnaval stops and another one with the time range 12:00–19:00 for the requests that return from the Carnaval. To simulate the increase in the number of requests that could arise due to the popularity of some performers, two normal distributions centered at different times: 10:30 and 14:45, have been used. Furthermore, to represent the increase of the requests at the end of the Carnaval, a normal distribution centered at 18:00 has been used. The distribution of the requests has been set so that 90% of the requests are generated by the normal distributions whereas the remaining 10% are created by sampling the uniform distributions.
  To represent the global locations, a discretization based on pickup points, similar to Raghavendra et al. (1992), has been used. For this study, we have used the set of transit stops available from the San Francisco Municipal Transportation Agency (SFMTA). This way, the city has been discretized according to the real disposition of the stops used by the municipal transportation system. Eq. (5) presents the different distributions used for generating the instances of this problem. In this and in the following equations, the parameter values of the distributions represent the minutes of the pickup values.

[5] which can be downloaded from the following URL: http://laurel.datsi.fi.upm.es/~smuelas/research/benchmark/.

$$\text{Carnaval} \sim \begin{cases} U(630, 1020) & \text{Bus stops -- Carnaval stops } (5\%) \\ N(630, 5) & \text{Bus stops -- Carnaval stops } (22.5\%) \\ U(720, 1140) & \text{Carnaval stops -- Bus stops } (5\%) \\ N(885, 5) & \text{Bus stops -- Carnaval stops } (22.5\%) \\ N(1080, 5) & \text{Carnaval stops -- Bus stops } (45\%) \end{cases} \tag{5}$$

- hospitals problem (HS): This problem represents the requests that could arise from a set of hospitals located around the city from both the patients and visitors to their home addresses. Since some patients could demand to be transported in a wheelchair, the vehicles should be specially adapted to perform this task. Several well-known hospitals addresses around the city have been used for representing the origin of the requests whereas, for the destinations, the same discretized data used in the first problem has been used. As presented in Eq. (6), a uniform distribution set in the range 8:00–21:00 has been used to simulate the times of the requests.

$$\text{Hospitals} \sim U(480, 1259) \text{ Hospitals stops -- Bus stops} \tag{6}$$

- hotels problem (HT): This problem simulates the requests that could be generated from/to a set of hotels from different sets of customers. Three different types of request have been simulated for this problem: (i) a large set of requests belonging to clients attending a conference that could be located in any of the three major convention centers of the city (70% of the total), (ii) requests belonging to regular tourists that would like to visit the most popular attractions of the city (20%) and (iii) a small set of requests that would like to go to any possible location of the city (10% of the total). Two normal distributions have been used to simulate the requests that could arise from the participants of the conferences. The first one represents the distribution around the start of the conferences per day and it has been set so that 99% of the requests fall into the range 08:15–08:45. The second one, simulated also with a normal distribution, represents the requests of the return from the convention centers to the hotel, where 99% of the requests fall into the range 18:45–19:15.
  The second group has been simulated using a uniform distribution to generate random values that belong to a different time range depending on the direction of the route: 09:00–20:00 if the clients go from the hotels to the attractions or 10:00–21:00 if the clients return from the attractions to the hotels.
  Finally, for the third group, two uniform distributions have been used to represent both directions of the requests: from the hotels to any location (8:00–22:00) and from any location to the hotels (09:00–23:59). Similarly to the previous problems, to represent the set of possible origins (or destinations) that demand to go to (or return from) the Carnaval, the transit stops from the SFMTA have been used. Eq. (7) briefly represents the generation of this problem.

$$\text{Hotels} \sim \begin{cases} U(480, 1320) & \text{Hotels stops -- Bus stops } (5\%) \\ N(510, 5) & \text{Hotels stops -- Convention centers stops } (35\%) \\ U(540, 1200) & \text{Hotels stops -- Attractions stops } (10\%) \\ U(540, 1439) & \text{Bus stops -- Hotels stops } (5\%) \\ U(600, 1260) & \text{Attractions stops -- Hotels stops } (10\%) \\ N(1140, 5) & \text{Convention centers stops -- Hotels stops } (35\%) \end{cases} \tag{7}$$

- music concert problem (M): This problem represents the demand that could arise at the end of a music concert. Therefore, a large amount of requests are going to collide in the same time frame, with the same origin address and with a different destination (using the transit stops obtained from the SFMTA). This problem, as shown in Eq. (8), is simulated with a Gamma distribution with the parameters $\alpha = 2$ and $\lambda = 1$ having its values being scaled by $60/9$ and having an offset of $22 * 60$. The idea is to have the majority of the requests concentrated around the time frame 22:00–23:00 with a quick rise of the demand in the first minutes of the interval and a continuous decrease of the demand along the following minutes.

$$\text{Music Concert} \sim 1320 + \Gamma(2, 1) * 60/9 \text{ Music concert stop -- Bus stops} \tag{8}$$

For all the problems, the maximum pickup time and the maximum ride duration constraints are represented by having to attend each request in, at most, 15 min from the requested pickup time with a maximum ride duration of the 300% of the time that would take to travel directly from the origin to the destination of the request.[6] For each request, its load, i.e., the number of passengers that are picked up at a stop, has been generated randomly between one and five passengers. All the vehicles have been set to a maximum capacity of 20 passengers and its number has been adjusted based on the problem characteristics (Table 3 displays the selected values). For the HS problem, the capacity of the vehicles has been reduced to a maximum of 5 passengers since they need to be specially adapted to transport patients in a wheelchair.

[6] Being both thresholds far more restrictive than the 30 min window and the absolute value of 90 min for maximum ride duration set in other benchmarks (Cordeau and Laporte, 2003).

Table 3
Number of available vehicles.

Problem	#Requests	#Vehicles
Carnaval	4000	381
Carnaval	8000	720
Carnaval	16,000	1440
Hospitals	4000	144
Hospitals	8000	287
Hospitals	16,000	574
Hotels	4000	330
Hotels	8000	408
Hotels	16,000	816
Music	4000	417
Music	8000	834
Music	16,000	1668

5.2. Comparison with other algorithms

In order to assess the benefits of the proposed solution (dVNS from now on), four other algorithms were selected to be executed with the same problem instances: an equivalent distributed algorithm where both the distribution of the requests and the routes were assigned randomly by using a uniform distribution (dVNS-rand), the sequential VNS algorithm from Muelas et al. (2013) (VNS-seq) that the distributed algorithms use as their core algorithm for optimizing the routes, an implementation of the sequential VNS algorithm from Parragh et al. (2010) (VNS-seq-P1), and the same algorithm but with the inclusion of the same criterion of the VNS-seq algorithm for the constraint violations (not allowed in candidate solutions) called VNS-seq-P2.

5.3. Parameter values

For the experimentation, we have selected the same parameter values for the VNS algorithm of our previous study (Muelas et al., 2013) where every parameter of the algorithm was properly tuned for the same problems (but with smaller dimensions), obtaining the best results of the compared algorithms.

In this study, we have focused in tuning the parameter values of the distributed algorithm. A fractional design based on orthogonal matrices according to the Taguchi method (Taguchi et al., 2005) was chosen to conduct a study on the effect of each parameter on the response variable. This method allows the execution of a limited number of configurations and still reports significant information on the best combination of parameter values. In particular, 9 different configurations were tested for the whole set of problems defined in Section 5.1 and with the parameter values presented in Table 4 using 25 executions per configuration.

In the Taguchi method, the concept of signal-to-noise (SN) ratio is introduced for measuring the sensitivity of the quality characteristic being investigated in a controlled manner to those external influencing factors (noise factors) not under control. The aim of the experiment is to determine the *highest* possible SN ratio for the results since a high value of the SN ratio implies that the signal is much higher than the random effects of the noise factors. The SN ratio estimate for the obtained values is defined in Eq. (9) where n denotes the total number of instances and $y_1, y_2, \ldots, y_n$ the target values (the $f(s)$ values in this case).

$$SN = -10\log\left(\frac{1}{n}\sum_{t=1}^{n} y_t^2\right) \tag{9}$$

Figs. 4 and 5 display the main effects plot for the data means and SN ratio, respectively. A main effect plot is a plot of the mean response values at each level of a design parameter. This plot can be used to compare the strength of the effects of the values of the parameter. The objective is to select the values that obtain the highest SN ratio with a lower mean value.

From these graphs it can be seen that the influence of some parameter values is more determinant than others. For example, as initially suspected, the unbalanced clusters of the k-medoids algorithm hinder the performance of the distributed algorithm. However, the proposed balanced version of the k-medoids algorithm obtains the best results in both the mean values and the SN ratio. Based on these results the algorithm was configured selecting the best value for each parameter. This selection is displayed in Table 4, where the selected values are marked in bold.

5.3.1. Stopping criterion

As commented in the previous section, both sequential and distributed algorithms have been selected for analyzing their results. To compare them in equivalent conditions a maximum execution time has been set as a stopping criterion. In particular, three hours has been used, which represents a reasonable threshold that a company could have to compute a solution to these large-scale problems.

Table 4
Parameter values tested.

Parameter	Values
Number of nodes	4, **8** and 16
Requests distribution	Random, k-medoids and **k-medoids balanced**
Routes distribution	Random, k-medoids and **k-medoids balanced**
Routes exchange period	600, **1200** and 2400 s

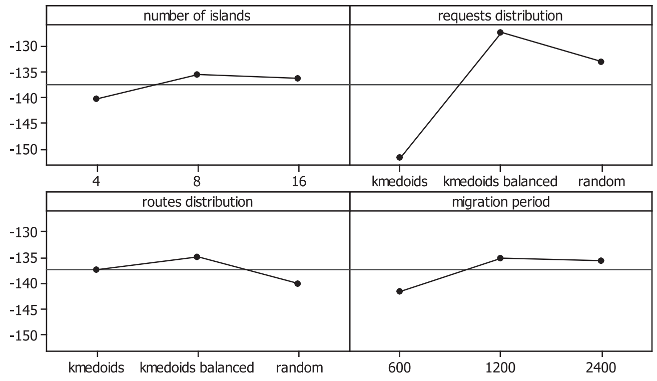

Fig. 4. Main effects plot for SN ratios.

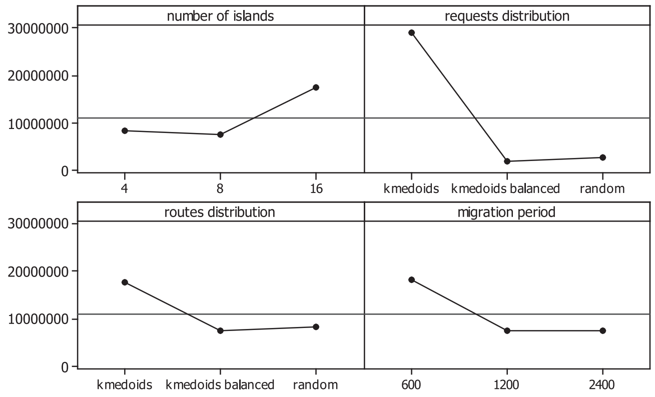

Fig. 5. Main effects plot for means.

6. Results and discussion

Tables 5–7 present, for each algorithm, the mean values (of 25 executions) obtained by each algorithm for the evaluation function. The best results for each problem are marked in bold. These results represent the overall number of seconds that the set of vehicles would need to attend all the requests of each problem instance and are computed, as detailed in Section 3.1, by taking into account the cost from the origin to the destination locations returned by a GIS system.

These tables also present the information of the sum of the times of the clients until they are picked up as well as the sum of the ride times of their trip. These values are not optimized since the DARP is originally laid out from the point of view of optimizing the cost of the routes, but it is interesting to include them to analyze the potential satisfaction of customers.

From these results it is clear that the proposed distributed algorithm has achieved the best overall results among all the problems and dimensions obtaining the best results in 22 out of 24 instances, with considerable differences in most of them, especially in the most difficult 16,000 requests problems. From the point of view of the sum of the pickup and ride times, the distributed algorithm has obtained the best results in 14 out of 24 problems for the pickup times, and 23 out of 24 problems for the ride times, proving again the beneficial behavior of the proposed algorithm.

It can be seen that all the sequential VNS algorithms are unable to obtain an initial solution in the proposed conditions for any of the 16,000 requests problems and only in half of the 8000 requests problems. Moreover, the proposed distributed algorithm has obtained considerable better results than the sequential algorithms with the 4000 requests instances. This

Table 5
Mean values (25 executions) of the algorithms under study with 4000 requests.

Alg.	Key	C-1	C-2	HS-1	HS-2
dVNS	$f(s)$	$\mathbf{5.2489 \times 10^5}$	$\mathbf{5.180 \times 10^5}$	$\mathbf{1.539 \times 10^6}$	$\mathbf{1.530 \times 10^6}$
	$\sum_i P_i$	$\mathbf{2.1490 \times 10^6}$	$\mathbf{2.167 \times 10^6}$	2.578×10^6	2.563×10^6
	$\sum_i L_i$	$\mathbf{2.3286 \times 10^6}$	$\mathbf{2.347 \times 10^6}$	$\mathbf{3.157 \times 10^6}$	$\mathbf{3.139 \times 10^6}$
dVNS-rand	$f(s)$	5.548×10^5	5.474×10^5	1.606×10^6	1.587×10^6
	$\sum_i P_i$	2.163×10^6	2.174×10^6	2.561×10^6	2.542×10^6
	$\sum_i L_i$	2.357×10^6	2.386×10^6	3.235×10^6	3.203×10^6
VNS-seq	$f(s)$	7.619×10^5	7.749×10^5	1.665×10^6	1.630×10^6
	$\sum_i P_i$	2.273×10^6	2.281×10^6	$\mathbf{2.489 \times 10^6}$	$\mathbf{2.466 \times 10^6}$
	$\sum_i L_i$	2.776×10^6	2.803×10^6	3.239×10^6	3.207×10^6
VNS-seq-P2	$f(s)$	8.408×10^5	8.538×10^5	1.743×10^6	1.705×10^6
	$\sum_i P_i$	2.247×10^6	2.246×10^6	2.655×10^6	2.607×10^6
	$\sum_i L_i$	2.854×10^6	2.865×10^6	3.224×10^6	3.185×10^6
VNS-seq-P1	$f(s)$	8.803×10^5	8.896×10^5	1.777×10^6	1.739×10^6
	$\sum_i P_i$	2.225×10^6	2.237×10^6	2.910×10^6	2.858×10^6
	$\sum_i L_i$	2.774×10^6	2.792×10^6	3.305×10^6	3.273×10^6
		HT-1	HT-2	M-1	M-2
dVNS	$f(s)$	$\mathbf{4.399 \times 10^5}$	$\mathbf{4.588 \times 10^5}$	$\mathbf{4.122 \times 10^5}$	$\mathbf{4.156 \times 10^5}$
	$\sum_i P_i$	$\mathbf{2.187 \times 10^6}$	$\mathbf{2.208 \times 10^6}$	2.163×10^6	2.175×10^6
	$\sum_i L_i$	$\mathbf{1.590 \times 10^6}$	$\mathbf{1.652 \times 10^6}$	$\mathbf{2.126 \times 10^6}$	2.124×10^6
dVNS-rand	$f(s)$	4.746×10^5	4.887×10^5	4.645×10^5	4.604×10^5
	$\sum_i P_i$	2.218×10^6	2.251×10^6	2.136×10^6	2.144×10^6
	$\sum_i L_i$	1.642×10^6	1.707×10^6	2.247×10^6	2.219×10^6
VNS-seq	$f(s)$	4.645×10^5	4.778×10^5	4.853×10^5	4.801×10^5
	$\sum_i P_i$	2.205×10^6	2.213×10^6	$\mathbf{2.043 \times 10^6}$	$\mathbf{2.039 \times 10^6}$
	$\sum_i L_i$	1.768×10^6	1.826×10^6	2.158×10^6	$\mathbf{2.124 \times 10^6}$
VNS-seq-P2	$f(s)$	4.703×10^5	4.858×10^5	5.294×10^5	5.248×10^5
	$\sum_i P_i$	2.188×10^6	2.209×10^6	2.054×10^6	2.048×10^6
	$\sum_i L_i$	1.784×10^6	1.845×10^6	2.266×10^6	2.230×10^6
VNS-seq-P1	$f(s)$	4.775×10^5	4.944×10^5	5.360×10^5	5.320×10^5
	$\sum_i P_i$	2.200×10^6	2.217×10^6	2.057×10^6	2.049×10^6
	$\sum_i L_i$	1.804×10^6	1.865×10^6	2.305×10^6	2.274×10^6

result contradicted our initial expectations, since the sequential algorithms have potentially more information to construct larger and better routes than the distributed algorithms. However, as will be shown later, even after the initialization, the solution from the distributed algorithms is, in most of the cases, better than the solution from the sequential algorithms which fall easily with large scale problems in worse local optima.

Regarding the comparison with the dVNS-rand in both the initial assignment of the requests as well as the distribution of the routes, it can be observed that the k-medoids approach has improved in all the problem scenarios the results of the simple random strategy, with only a small penalty in obtaining the initial solution as will be shown later.

To provide a quantitative vision of the magnitude of the improvements, Tables 8–10 present the relative improvements of the average performance of the evaluation function between the dVNS algorithm and the dVNS-rand and VNS-seq algorithms. It is shown that, in average terms, the dVNS algorithm has obtained an improvement of a 6% of the evaluation function against the dVNS-rand algorithm and a 9% improvement against the best sequential VNS algorithm. The best improvements when compared to the sequential algorithm have been obtained in the carnaval problem and music concert problem, where the dVNS algorithm has been able to improve by a 32% and 14% the results of the sequential algorithm.

Finally, several figures have been generated to offer a concrete view of the type of solutions that are being generated. Fig. 6 represents the locations of the stops used as the origin or destination of the requests created for the carnaval problem (as described in Section 5.1), whereas Fig. 7 depicts the heatmaps for two solutions for the same problem with 4000 and 8000 requests, respectively. These Figures provide a graphical representation of the density of routes that traverse the same streets using different degrees between green (minimum value) and red (maximum value) to represent the range of density values between them. In this figure it can easily be observed the different density levels of each street and how most of the routes are concentrated around the main stops of the carnaval and the main roads near them. This result agrees with what was expected from the problem description since all the requests go from/to any global stop to/from any of the carnaval stops. Moreover, the density of the overlapping routes is greatly reduced once the routes move away from the carnaval hub and the main roads connecting the carnaval and start to branch out to satisfy each individual request. The heatmap for the 8000 requests scenario shows how the increase in the number of requests affects the whole heat map, increasing the densities through out the whole map and especially around the area that contains the carnaval stops.

Table 6
Mean values (25 executions) of the algorithms under study with 8000 requests.

Alg.	Key	C-1	C-2	HS-1	HS-2
dVNS	$f(s)$	**1.083×10^{6}**	**1.086×10^{6}**	**3.125×10^{6}**	**3.070×10^{6}**
	$\sum_i P_i$	**4.375×10^{6}**	**4.366×10^{6}**	**5.002×10^{6}**	5.042×10^{6}
	$\sum_i L_i$	**4.804×10^{6}**	**4.806×10^{6}**	**6.352×10^{6}**	**6.280×10^{6}**
dVNS-rand	$f(s)$	1.159×10^{6}	1.164×10^{6}	3.281×10^{6}	3.251×10^{6}
	$\sum_i P_i$	4.411×10^{6}	4.383×10^{6}	5.036×10^{6}	5.046×10^{6}
	$\sum_i L_i$	4.915×10^{6}	4.911×10^{6}	6.521×10^{6}	6.456×10^{6}
VNS-seq	$f(s)$	Not finished	Not finished	3.271×10^{6}	3.236×10^{6}
	$\sum_i P_i$	Not finished	Not finished	5.031×10^{6}	**5.033×10^{6}**
	$\sum_i L_i$	Not finished	Not finished	6.512×10^{6}	6.444×10^{6}
VNS-seq-P2	$f(s)$	Not finished	Not finished	3.384×10^{6}	3.339×10^{6}
	$\sum_i P_i$	Not finished	Not finished	5.520×10^{6}	5.557×10^{6}
	$\sum_i L_i$	Not finished	Not finished	6.491×10^{6}	6.422×10^{6}
VNS-seq-P1	$f(s)$	Not finished	Not finished	3.433×10^{6}	3.386×10^{6}
	$\sum_i P_i$	Not finished	Not finished	5.921×10^{6}	5.940×10^{6}
	$\sum_i L_i$	Not finished	Not finished	6.633×10^{6}	6.558×10^{6}
		HT-1	HT-2	M-1	M-2
dVNS	$f(s)$	9.895×10^{5}	1.016×10^{6}	**8.581×10^{5}**	**8.684×10^{5}**
	$\sum_i P_i$	4.767×10^{6}	4.788×10^{6}	4.202×10^{6}	4.191×10^{6}
	$\sum_i L_i$	**3.532×10^{6}**	**3.606×10^{6}**	**4.211×10^{6}**	**4.226×10^{6}**
dVNS-rand	$f(s)$	1.055×10^{6}	1.081×10^{6}	9.530×10^{5}	9.593×10^{5}
	$\sum_i P_i$	4.816×10^{6}	4.826×10^{6}	**4.170×10^{6}**	**4.169×10^{6}**
	$\sum_i L_i$	3.701×10^{6}	3.759×10^{6}	4.443×10^{6}	4.441×10^{6}
VNS-seq	$f(s)$	**8.950×10^{5}**	**9.168×10^{5}**	Not finished	Not finished
	$\sum_i P_i$	4.735×10^{6}	4.711×10^{6}	Not finished	Not finished
	$\sum_i L_i$	3.787×10^{6}	3.859×10^{6}	Not finished	Not finished
VNS-seq-P2	$f(s)$	8.982×10^{5}	9.200×10^{5}	Not finished	Not finished
	$\sum_i P_i$	4.723×10^{6}	4.700×10^{6}	Not finished	Not finished
	$\sum_i L_i$	3.791×10^{6}	3.863×10^{6}	Not finished	Not finished
VNS-seq-P1	$f(s)$	9.039×10^{5}	9.265×10^{5}	Not finished	Not finished
	$\sum_i P_i$	**4.721×10^{6}**	**4.693×10^{6}**	Not finished	Not finished
	$\sum_i L_i$	3.819×10^{6}	3.894×10^{6}	Not finished	Not finished

6.1. Statistical analysis

After computing the average errors for each algorithm, a statistical analysis has been carried out to assess the significance of the results. First, we have compared the overall results of the algorithms in all the problems. Since the sequential VNS algorithms were unable to obtain a solution for half of the problems, only the distributed algorithms were compared in the first analysis. The non-parametrical Wilcoxon signed-rank test (García et al., 2009) was selected to evaluate if the dVNS results were better (smaller) than the dVNS-rand algorithm. The test returned a p-value of 5.96E−08 proving, as expected, the significance of the results.

In the second analysis we selected only the problems where the sequential algorithms were able to obtain a solution in order to compare the three algorithms, selecting a total of 12 problems (8 of 4000 requests and 4 of 8000 requests). Following the guidelines of García et al. (2009), the algorithms were first compared with the Friedman test to detect significant differences. A value of 27.4666 was obtained for the chi-squared statistic, which corresponds with a p-value of 1.599E−05 confirming the existence of significant differences among the algorithms. According to this test, the algorithms were ranked as shown in Table 11, including two other global measures in the comparison: the nWins value (Muelas et al., 2007) and the number of functions where each algorithm obtained the best mean results. Once again, the proposed algorithm obtained the best results in all the measures. Similarly as before, the Wilcoxon signed-rank test was used for comparing the results, adjusting the obtained p-values to take into account the Family-Wise Error Rate (FWER) when conducting multiple comparisons. The results of these tests are reported in Table 12, and show, for all of them, that there is statistical evidence to state that the proposed dVNS algorithm is significantly better than the remaining algorithms.

Between the different VNS approaches, the VNS-seq has obtained the best results, being significantly better than the other two VNS algorithms (p-value = 4.88E−04 including the FWER correction). From these two, the VNS-seq-P2 has obtained significantly better results than VNS-seq-P1 (p-value = 2.44E−04) proving the disadvantageous effect of allowing constraints violations in candidate solutions.

Table 7
Mean values (25 executions) of the algorithms under study with 16,000 requests.

Alg.	Key	C-1	C-2	HS-1	HS-2
dVNS	$f(s)$	$\mathbf{2.343\times10^{6}}$	$\mathbf{2.329\times10^{6}}$	$\mathbf{6.214\times10^{6}}$	$\mathbf{6.249\times10^{6}}$
	$\sum_i P_i$	$\mathbf{8.824\times10^{6}}$	$\mathbf{8.851\times10^{6}}$	$\mathbf{9.857\times10^{6}}$	9.860×10^{6}
	$\sum_i L_i$	$\mathbf{9.920\times10^{6}}$	$\mathbf{9.925\times10^{6}}$	$\mathbf{1.259\times10^{7}}$	$\mathbf{1.262\times10^{7}}$
dVNS-rand	$f(s)$	2.550×10^{6}	2.544×10^{6}	6.561×10^{6}	6.569×10^{6}
	$\sum_i P_i$	9.011×10^{6}	9.027×10^{6}	9.864×10^{6}	$\mathbf{9.608\times10^{6}}$
	$\sum_i L_i$	1.021×10^{7}	1.023×10^{7}	1.299×10^{7}	1.266×10^{7}
VNS-seq	$f(s)$	Not finished	Not finished	Not finished	Not finished
	$\sum_i P_i$	Not finished	Not finished	Not finished	Not finished
	$\sum_i L_i$	Not finished	Not finished	Not finished	Not finished
VNS-seq-P2	$f(s)$	Not finished	Not finished	Not finished	Not finished
	$\sum_i P_i$	Not finished	Not finished	Not finished	Not finished
	$\sum_i L_i$	Not finished	Not finished	Not finished	Not finished
VNS-seq-P1	$f(s)$	Not finished	Not finished	Not finished	Not finished
	$\sum_i P_i$	Not finished	Not finished	Not finished	Not finished
	$\sum_i L_i$	Not finished	Not finished	Not finished	Not finished
		HT-1	HT-2	M-1	M-2
dVNS	$f(s)$	$\mathbf{1.924\times10^{6}}$	$\mathbf{1.920\times10^{6}}$	$\mathbf{1.805\times10^{6}}$	$\mathbf{1.767\times10^{6}}$
	$\sum_i P_i$	$\mathbf{9.407\times10^{6}}$	$\mathbf{9.432\times10^{6}}$	8.183×10^{6}	$\mathbf{8.220\times10^{6}}$
	$\sum_i L_i$	$\mathbf{7.162\times10^{6}}$	$\mathbf{7.144\times10^{6}}$	$\mathbf{8.380\times10^{6}}$	$\mathbf{8.321\times10^{6}}$
dVNS-rand	$f(s)$	2.005×10^{6}	1.988×10^{6}	1.874×10^{6}	1.884×10^{6}
	$\sum_i P_i$	9.603×10^{6}	9.564×10^{6}	$\mathbf{8.173\times10^{6}}$	8.229×10^{6}
	$\sum_i L_i$	7.515×10^{6}	7.471×10^{6}	8.547×10^{6}	8.632×10^{6}
VNS-seq	$f(s)$	Not finished	Not finished	Not finished	Not finished
	$\sum_i P_i$	Not finished	Not finished	Not finished	Not finished
	$\sum_i L_i$	Not finished	Not finished	Not finished	Not finished
VNS-seq-P2	$f(s)$	Not finished	Not finished	Not finished	Not finished
	$\sum_i P_i$	Not finished	Not finished	Not finished	Not finished
	$\sum_i L_i$	Not finished	Not finished	Not finished	Not finished
VNS-seq-P1	$f(s)$	Not finished	Not finished	Not finished	Not finished
	$\sum_i P_i$	Not finished	Not finished	Not finished	Not finished
	$\sum_i L_i$	Not finished	Not finished	Not finished	Not finished

Table 8
Relative improvements of the mean values (25 executions) of the evaluation function with 4000 requests.

dVNS vs.	C-1 (%)	C-2 (%)	HS-1 (%)	HS-2 (%)
VNS-seq	31.10	33.15	7.5	6.13
dVNS-rand	5.4	5.37	4.2	3.5
	HT-1 (%)	HT-2 (%)	M-1 (%)	M-2 (%)
VNS-seq	5.29	4	15.06	13.43
dVNS-rand	7.31	4	11.25	9.7

Table 9
Relative improvements of the mean values (25 executions) of the evaluation function with 8000 requests.

dVNS vs.	C-1	C-2	HS-1	HS-2
VNS-seq	Not finished	Not finished	4.46%	5.1%
dVNS-rand	6.5%	6.5%	4.7%	5.5%
	HT-1	HT-2	M-1	M-2
VNS-seq	−10.5%	−10%	Not finished	Not finished
dVNS-rand	6.2%	6%	9.9%	9.4%

6.2. Analysis of the evolution

In the previous section we conducted a statistical analysis from the point of view of the evaluation of the final values. In this section we analyze the evolution of these values for the different algorithms and problems.

Table 10
Relative improvements of the mean values (25 executions) of the evaluation function with 16,000 requests.

dVNS vs.	C-1	C-2	HS-1	HS-2
VNS-seq	Not finished	Not finished	Not finished	Not finished
dVNS-rand	8.11%	8.4%	5.2%	4.8%
	HT-1	HT-2	M-1	M-2
VNS-seq	Not finished	Not finished	Not finished	Not finished
dVNS-rand	4%	3.4%	3.6%	6.2%

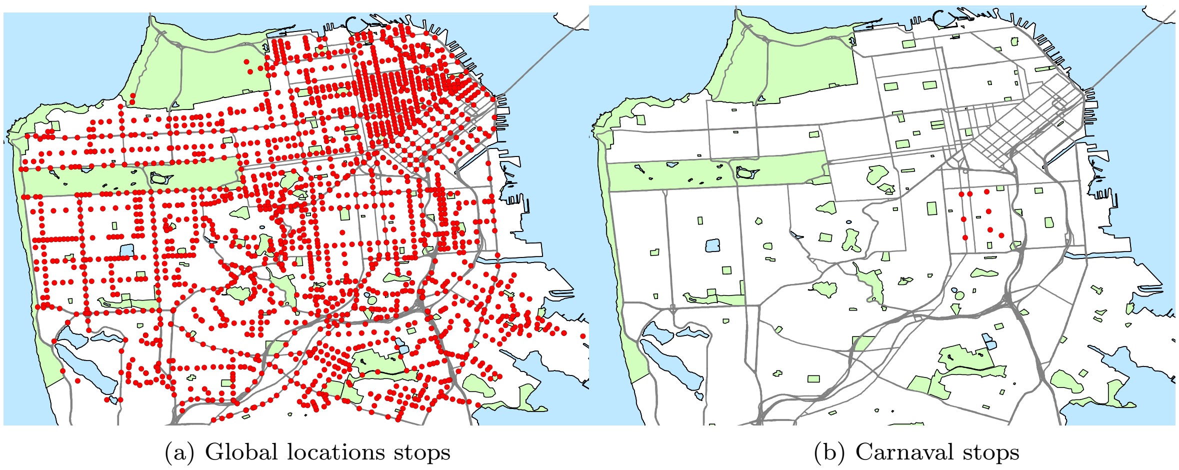
(a) Global locations stops (b) Carnaval stops

Fig. 6. Map of the locations used to generate the requests in the carnaval problem.

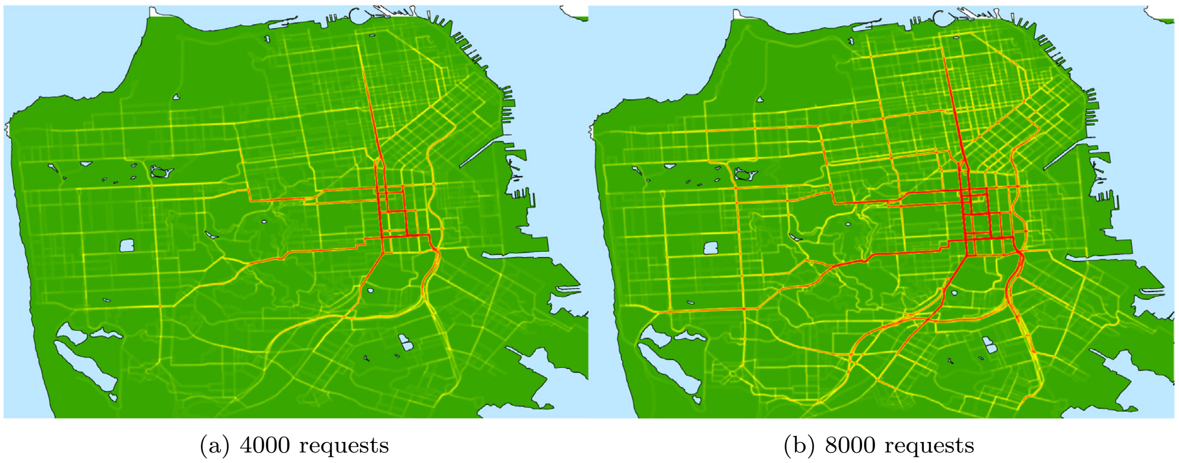
(a) 4000 requests (b) 8000 requests

Fig. 7. Heatmaps representations of a solution for the carnaval problem.

Table 11
Average ranking, number of best results and nWins value.

	Ranking	#Best	nWins
dVNS	1.5	10	4
VNS-seq	2.33	2	1
dVNS-rand	3	0	0
VNS-seq-P2	3.5	0	−1
VNS-seq-P1	4.66	0	−4

Table 12
Statistical validation (dVNS is the control algorithm).

dVNS vs.	Wilcoxon *p*-value
dVNS-rand	2.44×10^{-04}[a]
VNS-seq	1.34×10^{-02}[a]
VNS-seq-P2	4.64×10^{-03}[a]
VNS-seq-P1	4.64×10^{-03}[a]
Wilcox *p*-value with FWER: dVNS vs. dVNS-rand, VNS-seq, VNS-seq-P2, VNS-seq-P1	2.28×10^{-02}[a]

[a] Means that there are statistical differences with significance level $\alpha = 0.05$.

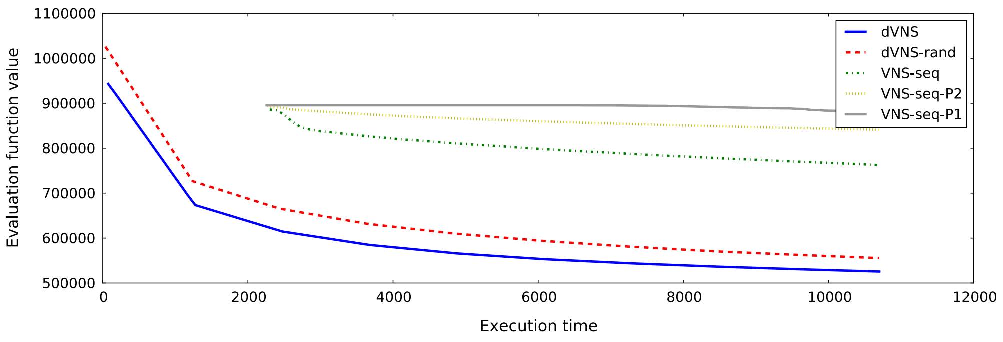

Fig. 8. Evolution of the average value of the evaluation function per execution time (in s) for C-1 4000 requests.

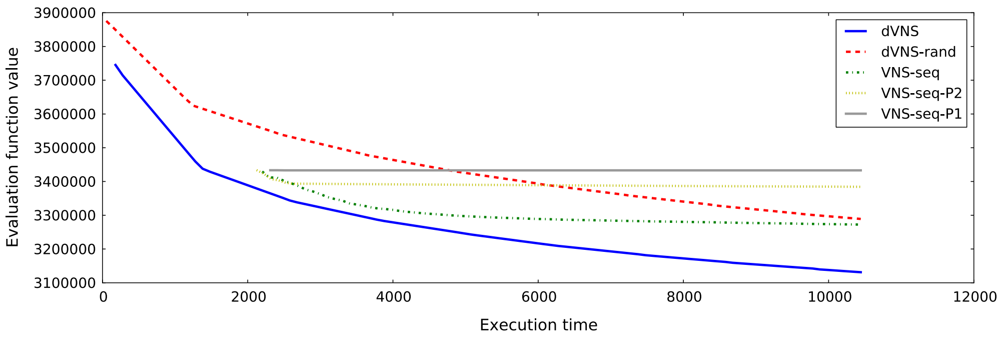

Fig. 9. Evolution of the average value of the evaluation function per execution time (in s) for HS-1 8000 requests.

Figs. 8, 9 and 11 present the evolution of the average value of the evaluation function per execution time for three different problems with 4000, 8000 and 16,000 requests, respectively. To avoid including the 24 different figures we have selected these three as representatives of the whole set since the remaining problems present a similar behavior. Besides this representative set of graphs, Fig. 10 has also been included since it is the only scenario where the sequential algorithms are able to improve the results of the distributed algorithm.

With these results it is clear that, although the k-medoids strategy delays the generation of the initial solution (when compared to the random approach), it allows the algorithm to start with a better solution. Even when the starting solution is of lower quality (as shown in Fig. 11), it has a greater potential to reach better solutions as demonstrated by the slope of the curve, which allows it to improve the solution of the random strategy at the first quarter of the execution.

When compared with the performance of the sequential VNS, both distributed algorithms seem to have a better convergence behavior with the potential of improving even further the solution if a longer stopping criterion was allowed, whereas the sequential algorithms seem to have almost converged with little margin for improving the solution. It can also be noticed the significant delay of the sequential algorithms to construct the initial solution as it is further extended in the following section. Only in the HT problem and 8000 requests the algorithms have been able to obtain better results than the distributed approach, although their initial solutions were generated just before the stopping criterion is reached. Analyzing the different alternatives for the sequential algorithms, it is clear that the selection of shakers and parameters of the VNS-seq, allows it to reach better solutions. Furthermore, it has been demonstrated the harmful effects of allowing the violation of constraints during the search, reducing substantially the potential improvements of the initial solution.

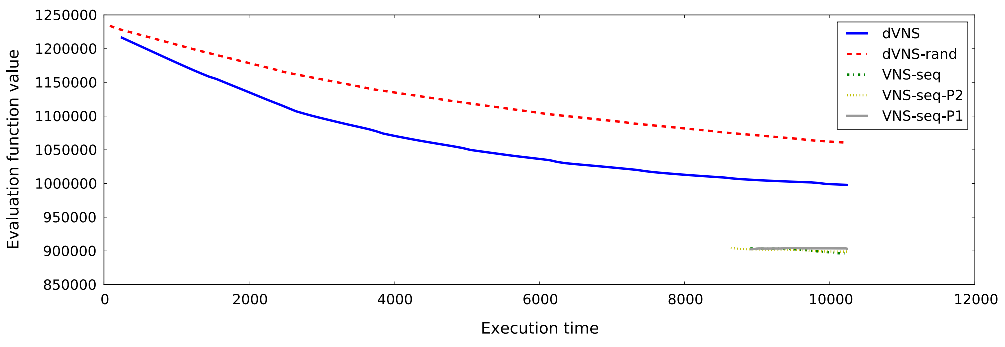

Fig. 10. Evolution of the average value of the evaluation function per execution time (in s) for HT-1 8000 requests.

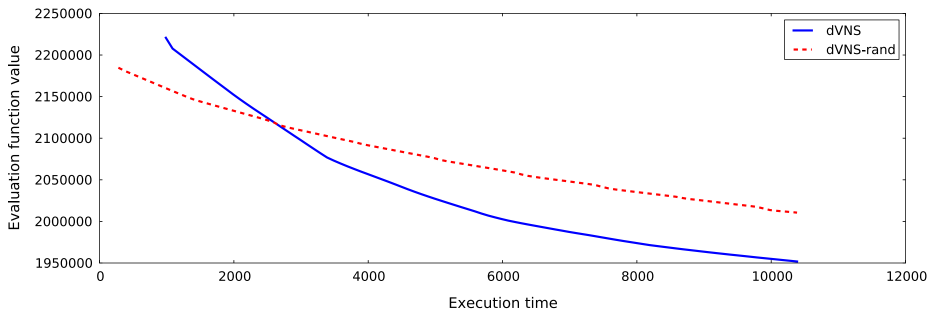

Fig. 11. Evolution of the average value of the evaluation function per execution time (in s) for HT-1 16,000 requests.

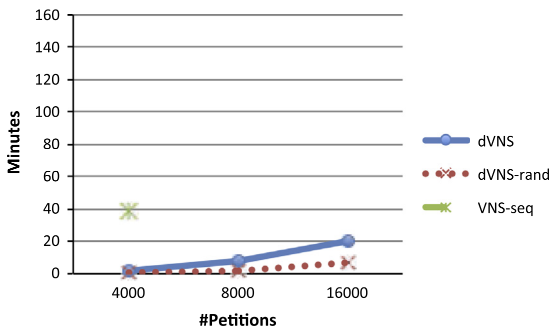

Fig. 12. Computational average running time to obtain the first solution for the carnaval problem.

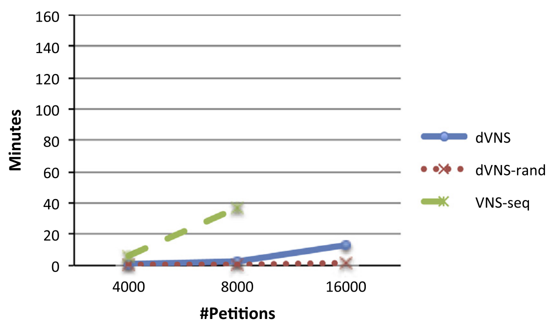

Fig. 13. Computational average running time to obtain the first solution for the hospitals problem.

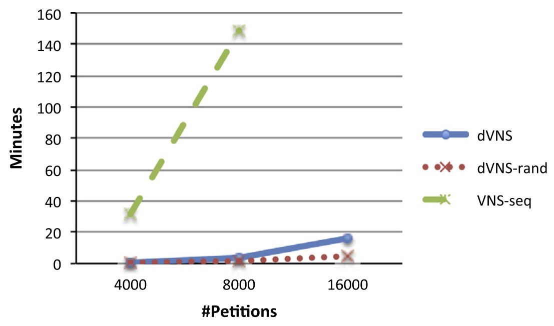

Fig. 14. Computational average running time to obtain the first solution for the hotels problem.

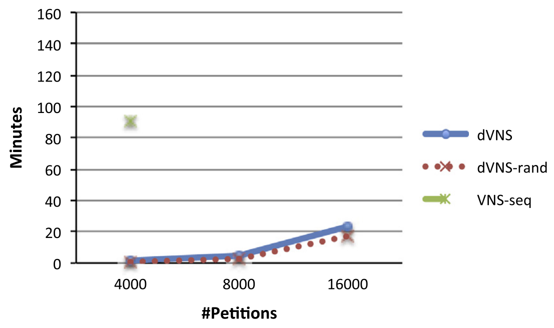

Fig. 15. Computational average running time to obtain the first solution for the music concert problem.

6.3. Scalability analysis

In this Section we analyze the scalability behavior of the algorithms. Since the stopping criterion is based on a maximum execution time, we have analyzed the computational execution time required by each algorithm to obtain an initial solution. Figs. 12–15 depict these values for each of the analyzed algorithms and problems.[7]

In general, both distributed algorithms exhibit an excellent behavior when compared against the sequential VNS. Clearly, the distribution of the requests has a significant impact in the execution time of the algorithms allowing them to obtain an initial solution in approximately twenty minutes in the worst case scenario (16,000), being well below of the stopping criterion and even less than the required execution time of the sequential algorithm for most of the 4000 problems.

As expected, the k-medoids strategy hinders the scalability performance of the dVNS algorithm when compared to the random strategy. However, even in the worst scenarios, this difference is relatively small ($\sim$10 min) and, both the quality of its solution and the potential to obtain better solutions, as was shown in the previous sections, minimizes this small overload.

7. Conclusions

In this work, a new distributed VNS algorithm has been presented for the optimization of large-scale DARP problems that a potential DRT transportation company could offer. The proposal uses a clustering algorithm to intelligently distribute the requests between the nodes in order to exploit the proximity relationships of the requests in parallel and achieve optimized solutions in a reasonable period of time. Moreover, by means of exchanging the routes and combining them using a similar strategy, the algorithm is able to improve the solution when compared with a simple random strategy.

The algorithm has been tested over a broad set of synthetic problems that have been generated with a GIS software and statistical procedures to simulate real demanding scenarios in the city of San Francisco. These problems have been designed in order to be challenging to traditional approaches reaching up to 16,000 requests (32,000 locations for classic VRP problems) in the most complex scenarios. The costs for computing the travel time between the different stops has been obtained by means of a popular GIS software which takes into account the different characteristics of the streets (maximum speed, crossroads, acceleration, etc.) to provide a reliable estimate of the cost of the travel time. However, we have used a fixed-cost matrix which does not take into account congestion-dependent travel time. For future work, traffic data should be incorporated into the computation of the travel times to provide more realistic results.

[7] For the VNS-seq algorithm only the problems for which the algorithm finishes before the stopping criterion are represented. Also, since there are no differences in the initialization procedure of the sequential algorithms, only the VNS-seq algorithm is displayed.

The performance proposal has been thoroughly analyzed and compared with an equivalent random distributed approach and several sequential VNS algorithms validating the results by means of statistical tests. The results from these procedures have proved that the proposed algorithm has obtained the best results. Not only the algorithm has achieved significantly better results, but has considerably reduced the execution time required to obtain an initial solution offering an interesting approach when dealing with the scenarios that a large city could demand.

Acknowledgments

This work was financed by the Spanish Ministry of Science through the NAVAN project and supported by the Cajal Blue Brain Project. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Supercomputing and Visualization Center of Madrid (CeSViMa). A. LaTorre gratefully acknowledges the support of the Spanish Ministry of Science and Innovation (MICINN) for its funding throughout the Juan de la Cierva program. We also acknowledge Óscar Pérez Rayas for his valuable help with the GIS software.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.trc.2015.02.024.

References

Allahviranloo, M., Chow, J.Y.J., Recker, W.W., 2014. Selective vehicle routing problems under uncertainty without recourse. Transp. Res. Part E 62, 68–88.

Baraty, S., Simovici, D., Zara, C., 2011. The impact of triangular inequality violations on medoid-based clustering. In: Kryszkiewicz, Marzena, Rybinski, Henryk, Skowron, Andrzej, Raś, Zbigniew W. (Eds.), Foundations of Intelligent Systems (Lecture Notes in Computer Science). Springer, Berlin Heidelberg, pp. 280–289.

Barreto, S., Ferreira, C., Paixão, J., Santos, B.S., 2007. Using clustering analysis in a capacitated location-routing problem. Eur. J. Oper. Res. 179 (3), 968–977.

Berguer, J., Barkaoui, M., 2004. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Comp. Operat. Res. (31), 2037–2053

Borndörfer, R., Grötschel, M., Klostermeier, M., Küttner, C., 1997. Telebus Berlin: Vehicle Scheduling in a Dial-a-ride System sc 97-23, Technical report. Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Carrabs, F., Cordeau, J.-F., Laporte, G., 2007. Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. INFORMS J. Comput. 19 (4), 618–632.

Cordeau, J.F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transport. Res. Part B: Methodol. 37 (6), 579–594.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., Savelsbergh, M.W.P., 2007. Transportation on demand. Handb. Operat. Res. Manage. Sci. 14, 429–466.

Crainic, T.G., 2008. Parallel solution methods for vehicle routing problems. In: Golden, Bruce, Raghavan, S., Wasil, Edward, Sharda, Ramesh, Voß, Stefan (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, Operations Research/Computer Science Interfaces Series, vol. 43. Springer, US, pp. 171–198.

Cullen, F.H., Jarvis, J.J., Ratliff, H.D., 1981. Set partitioning based heuristics for interactive routing. Networks 11, 125–143.

Davidović, Tatjana, Crainic, Teodor Gabriel, 2012. MPI parallelization of variable neighborhood search. Electron. Notes Disc. Math. 39, 241–248.

Diana, M., Dessouky, M.M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transport. Res. Part B: Methodol. 38 (6), 539–557.

Dondo, R., Cerdá, J., 2007. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. Eur. J. Oper. Res. 176 (3), 1478–1507.

D'Souza, C., Omkar, S.N., Senthilnath, J., 2012. Pickup and delivery problem using metaheuristics techniques. Expert Syst. Appl. 39 (1), 328–334.

Everitt, B.S., Landau, S., Morven, L., Stahl, D., 2011. Cluster Analysis. Wiley.

Ganesh, K., Narendran, T.T., 2007. CLOVES: a cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. Eur. J. Oper. Res. 178 (3), 699–717.

García, S., Molina, D., Lozano, M., Herrera, F., 2009. A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the CEC2005 special session on real parameter optimization. J. Heurist. (15), 617–644

Gendreau, M., 2010. Solving Large-Scale Vehicle Routing Problems with Time Windows: The State-of-the-Art, Technical report. CIRRELT, Montreal, Québec, Canada.

Horn, M.E.T., 2002. Fleet scheduling and dispatching for demand-responsive passenger services. Transport. Res. Part C: Emerg. Technol. 10 (1), 35–63.

Hyytiä, E., Penttinen, A., Sulonen, R., 2012. Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives. Comp. Operat. Res. (39), 3021–3030

Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2006. Solving the dial-a-ride problem using genetic algorithms. J. Operat. Res. Soc. 58 (10), 1321–1331.

Kaufman, L., Rousseeuw, 2005. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley-Interscience.

Li, F., Golden, B., Wasil, E., 2005. Very large-scale vehicle routing: new test problems, algorithms, and results. Comp. Operat. Res. 32 (5), 1165–1179.

Li, F., Golden, B., Wasil, E., 2007. The open vehicle routing problem: algorithms, large-scale test problems, and computational results. Comp. Operat. Res. 34 (10), 2918–2930.

Lucasius, C.B., D Dane, A., Kateman, G., 1993. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. Anal. Chim. Acta 3 (282), 647–669.

Matis, P., Kohani, M., 2011. Very large street routing problem with mixed transportation mode. CEJOR 19 (3), 359–369.

Muelas, S., LaTorre, A., Peña, J.-M., 2013. A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. Expert Syst. Appl. 40 (14), 5516–5531.

Muelas, S., Peña, J.M., LaTorre, A., Robles, V., 2010. A new initialization procedure for the distributed estimation of distribution algorithms. Soft Comput. – A Fus. Found., Methodol. Appl. 15 (4), 713–720.

Muelas, S., Peña, J.M., Robles, V., LaTorre, A., DeMiguel, P., 2007. Machine learning to analyze migration parameters in parallel genetic algorithms. In: Corchado, Emilio, Corchado, Juan M., Abraham, Ajith (Eds.), Innovations in Hybrid Intelligent Systems, Advances in Soft Computing, vol. 44. Springer, pp. 199–206.

Ochi, L.S., Vianna, D.S., Drummond, L.M.A., Victor, A.O., 1998. A parallel algorithm for the vehicle routing problem with heterogeneous fleet. Fut. Gener. Comp. Syst. (14), 285–292

Parragh, Sophie N., Doerner, Karl F., Hartl, Richard F., 2010. Variable neighborhood search for the dial-a-ride problem. Comp. Operat. Res. 37 (6), 1129–1138.

Parragh, Sophie N., Doerner, Karl F., Hartl, Richard F., Gandibleux, Xavier, 2009. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks 54 (4), 227–242.

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur. J. Oper. Res. 66 (3), 331–340.

Qi, M., Lin, W.-H., Li, N., Miao, L., 2012. A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows. Transport. Res. Part E: Logist. Transport. Rev. 48 (1), 248–257.

Raghavendra, A., Krishnakumar, T.S., Muralidhar, R., Sarvanan, D., Raghavendra, B.G., 1992. A practical heuristic for a large scale vehicle routing problem. Eur. J. Oper. Res. 57 (1), 32–38.

Rebibo, K., 1974. A computer controlled dial-a-ride system. traffic control and transportation systems. In: Proceedings of 2nd IFAC/IFIP/IFORS Symposium Monte Carlo. North- Holland, Amsterdam.

Schulze, J., Fahle, T., 1999. A parallel algorithm for the vehicle routing problem with time window constraints, combinatorial optimization: recent advances in theory and praxis. Ann. Oper. Res. 86, 585–607.

Singh, S.S., Chauhan, N.C., 2011. K-means v/s K-medoids: a comparative study. In: National Conference on Recent Trends in Engineering and Technology (NCRTET). Anand, India.

Taguchi, G., Chowdhury, S., Wu, Y., 2005. Taguchi's Quality Engineering Handbook. John Wiley.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comp. Operat. Res. 40 (1), 489-475.

Wilson, H., Sussman J., Wang, H., Higonnet. B., 1971. Scheduling algorithms for dial-a-ride system usl tr-70-13, Technical report. Urban Systms Laboratory, MIT, Cambridge, MA.

Wilson, H., Weissberg, H., 1967. Advanced dial-a-ride algorithms research project: Final report r76-20. Technical report, Department of Civil Engineering. MIT, Cambridge, MA.

Xu, J., Huang, Z., 2009. An intelligent model for urban demand-responsive transport system control. J. Softw. 4 (7), 766–776.