Expert Systems with Applications 40 (2013) 5516–5531

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city

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

[a] Computer Architecture Department, Facultad de Informática, Universidad Politécnica de Madrid, Spain
[b] Instituto Cajal, Consejo Superior de Investigaciones Científicas, Madrid, Spain

ARTICLE INFO

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

ABSTRACT

On-demand transportation is becoming a new necessary service for modern (public and private) mobility and logistics providers. Large cities are demanding more and more share transportation services with flexible routes, resulting from user dynamic demands. In this study a new algorithm is proposed for solving the problem of computing the best routes that a public transportation company could offer to satisfy a number of customer requests. In this problem, known in the literature as the dial-a-ride problem, a number of passengers has to be transported between pickup and delivery locations trying to minimize the routing costs while respecting a set of pre-specified constraints (maximum pickup time, maximum ride duration and maximum load per vehicle). For optimizing this problem, a new variable neighborhood search has been developed and tested on a set of 24 different scenarios of a large-scale dial-a-ride problem in the city of San Francisco. The results have been compared against two state-of-the-art algorithms of the literature and validated by means of statistical procedures proving that the new algorithm has obtained the best overall results.

1. Introduction

The concept of flexible transportation services has become a hot topic in the design of modern city mobility. Traditional public transportation has shown its limitations to satisfy the changes in growing cities and particularly unable to adapt to particular events that affects significantly to user's transportation demands. On the other hand, individual transportation services (such as taxi or limo services) have higher economic costs not affordable in many cases. Demand-responsive transport (DRT) has recently appeared to provide a transport model with flexible scheduling of routes, based on dynamic user's demands using medium size vehicles shared by several clients. DRT provides solutions not only for passengers mobility demands but also in the fields of logistics and medical (non-emergency) transportation services (Xu & Huang, 2009). Moreover, the adoption of DRT transportation models has many beneficial side effects in pollution reduction and traffic congestion.

The practical application of large scale DRT services has the challenge to provide efficient solutions to large number of users demands over large city areas. In the literature, a transportation problem with these particular characteristics, the dial-a-ride problem (DARP) has previously been studied. The DARP is an example of a transportation problem in which the objective is to determine the best routing schedule for a set of vehicles in order to satisfy the transportation requests for a number of customers. A request consists of a specified pickup (origin) and delivery location (destination) along with a desired departure or arrival time as well as the number of passengers to be transported. Each customer has to be transported to his destination but not necessarily directly (they can share a ride). Furthermore, the problem takes into account the passenger satisfaction, expressed it in terms of additional constraints such as the maximum ride time of the users or the maximum waiting time at the pickup locations. In order to apply DRT transportation solutions in a large city, it is required to be able to solve large-scale DARP problems. However, 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.

In the recent years, these DARP systems have become increasingly popular (Cordeau, Laporte, Potvin, & Savelsbergh, 2007, chap. 7) due to a number of reasons; with the trend towards the development of ambulatory health care services for aging people, more and more people rely on door-to-door transportation systems provided by local authorities. Shuttle services have also gained in popularity between organizations and, recently, taxi companies have started to offer a sharing service for their customers. Several on-demand courier services and merchandise transportation have equivalent requirements.

* Corresponding author. Tel.: +34 914524900x1763.
E-mail addresses: smuelas@fi.upm.es (S. Muelas), atorre@fi.upm.es (A. LaTorre), jmpena@fi.upm.es (J.-M. Peña).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.04.015

This study presents the results of the work we developed for a transport company interested in providing an on-demand transportation service, taking passengers at their requested locations and times while sharing the trip with passengers that have similar demands. The objective of the company is to optimize the cost of the proposed routes while preserving a reasonable quality in the service offered to its customers.

For this task, a variable neighborhood search (VNS) algorithm has been developed and adapted for the proposed problem. This metaheuristic has been successfully used with similar problems, obtaining competitive results in all the studies (Carrabs, Cordeau, & Laporte, 2007; Parragh, Doerner, Hartl, & Gandibleux, 2009, 2010). For analyzing the results a benchmark containing several real-life-based scenarios with underlying complex request patterns has been developed, comparing the results of the proposed algorithm against several state-of-the-art algorithms of the literature.

The remainder of this article is organized as follows: Section 2 presents an overview of the literature with vehicle routing problems. Sections 3 and 4 define the problem and the evaluation function. Section 5 details the proposed algorithm. In Section 6 the experimental scenario is described in depth. Section 7 presents and comments on the results obtained and lists the most relevant facts from this analysis. Finally, Section 8 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 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 like the time window, 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 problems, 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 class has been extensively studied in the literature. The first publications in this area were published in the late 1960s and early 1970s (Rebibo, 1974; Wilson & Weissberg, 1967; Wilson, Wang, & Higonnet, 1971). Since then, several approaches have been studied for solving this problem.

Regarding the exact methods, two of the most successful approaches can be found in Cordeau (2006) and Ropke, Cordeau, and Laporte (2007). Both studies used a branch-and-cut algorithm for solving a static DARP. In Cordeau (2006), an algorithm based on a 3-index mixed-integer problem formulation was proposed for solving to optimality a DARP of 36 requests. In Ropke et al. (2007), two new 2-index based formulations and additional valid inequalities were used for solving to optimality an instance of 194 nodes.

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. One of the first approaches was analyzed in Cullen, Jarvis, and Ratliff (1981). This study proposed an interactive algorithm based on a set partitioning formulation solved by means of column generation although user-related constraints were not explicitly considered. In Borndörfer, Grötschel, Klostermeier, and Küttner (1997) a set partitioning approach consisting of two steps was proposed. The first clustering step identifies segments of possible vehicle tours such that more than one person is transported at a time. In the second step, the selected orders are chained to yield possible routes respecting all side constraints. The clustering step can be solved optimally whereas the routing subproblem was solved approximately by a branch and bound algorithm. Customer ride times were implicitly considered by using time windows.

Metaheuristics are also a common approach when dealing with the DARP (D'Souza, Omkar, & Senthilnath, 2012). In general, heuristic methods tend to run faster whereas metaheuristics usually outperform basic heuristic procedures in terms of solution quality. Cordeau and Laporte proposed in Cordeau and Laporte (2003) a data set of 20 instances with sizes between 24 and 144 requests. For solving the instances, they proposed a tabu search (TS) algorithm in which at each step, all the possible neighbors created by moving one request to another route are considered. The solution moves to the best neighbor unless the move itself is forbidden in a Tabu memory that contains recents moves that lead to worse solutions (used to avoid cycling). In this work, solutions were evaluated using an evaluation function that takes into account the total cost of the routes and penalizes this value if one of the constraints (maximum pickup time, maximum ride duration, maximum load exceeded and maximum route duration) is not satisfied. In Jorgensen, Larsen, and Bergvinsdottir (2006), a genetic algorithm (GA) was presented for solving the DARP. The algorithm is based on the classical cluster-first (assigning customers to vehicles), route-second approach (solving independent routing problems using a routing heuristic). A different evaluation function was selected, namely a weighted combination of routing costs, total route duration, user ride time, user waiting time, and penalties for violations of route duration, time window and ride time. The authors compared their results to the TS of Cordeau and Laporte (2003) showing that the TS obtained better results for route duration whereas the GA improved the pickup time and ride duration. Finally, a VNS aimed at minimizing the total routing costs while respecting some constraints was presented in Parragh, Doerner, and Hartl (2010) (VNSP from here on). Three classes of neighborhoods were used by the algorithm: one based on swapping requests, a second one that uses an ejection chain approach and the last one that swaps natural sequences (sequences in which the vehicle load at the end is zero). This algorithm compared itself against both the TS and the GA previously described improving their results on the selected benchmark.

In this Section we have offered a brief review of the main approaches used with the DARP. For a complete review of the literature we refer the reader to Cordeau et al. (2007) and Parragh, Doerner, and Hartl (2008).

3. Problem definition

For this work, the formulation of the problem has been based on the studies conducted in Cordeau and Laporte (2003) and Parragh et al. (2010) although some modifications have been added in order to correctly represent the proposed scenario. Therefore, the static DARP is defined on a complete directed 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 ($q_i > 0$) 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.

Since this study is focused in optimizing the problems 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. This implies a minor modification to the original model proposed in Cordeau and Laporte (2003) and Parragh et al. (2010), where their objective was to model a transportation service for the disabled people who cannot use regular public transportation systems and who need to be able to specify either the pickup or the delivery time (depending if they are going or coming from the hospital). The requests represented in our problem are modeled according to the usual requests for a common public transportation system, like, for example, a typical taxi or shuttle service, where the customers specify when they would like to be picked up.

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, 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. However, instead of using an absolute approach for computing the maximum ride time, we have used a relative approach that computes the maximum ride time value taking into account the ride time of the pickup and delivery vertices of a request and multiplying this value by a constant ($Lmax_i = t_{i,i+n} * maxridetime_c$). This way, the constraint represents better the possible dissatisfaction of the user. 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 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

4. Evaluation function

As previously mentioned, for this work we have focused on minimizing the total cost of the routes proposed to solve the solution, 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.

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

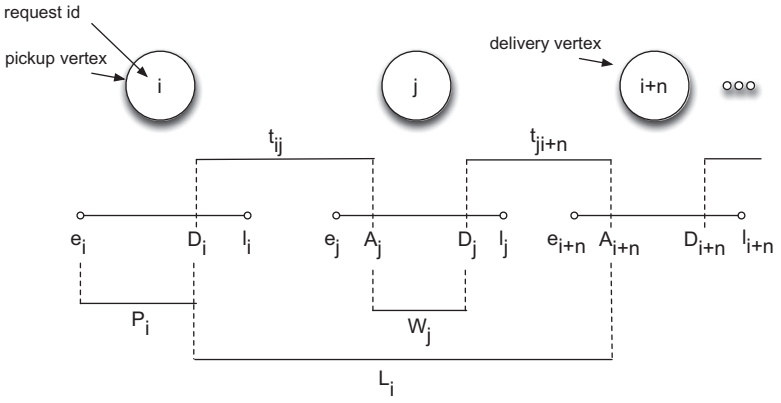

Fig. 1. Representation of the problem notation.

From an implementation point of view, the aforementioned definition of the evaluation function can be transformed to $f(s) = \sum_{route=1}^{m} f_{route}(s)$ where $f_{route}(s)$ corresponds to the evaluation function defined in Eq. (1) but considering only the vertices that belong to a route.

4.1. Forward time slack and solution evaluation

For the optimal computation of the A_i,W_i and D_i values we have followed the evaluation scheme used in Parragh et al. (2010) and Cordeau and Laporte (2003) and briefly described in Algorithm 1. With this scheme, the route duration is minimum and ride time limits are respected whenever is possible.

Consider a particular route $k = (v_1, \ldots, v_i, \ldots, v_q)$. It is clear that setting $D_i = max\{e_i, A_i\}$ for $i = 1, \ldots, q$ is optimal because the vehicle leaves the depot as early as possible. However, it must be taken into account that a solution that is infeasible due to the ride duration constraints, can, in fact, be feasible (or reduce its violation values) if the departure at some vertices is properly delayed, especially when the time window associated with a vertex is wide. This idea was used by Savelsbergh (1992) to define the forward time slack F_i of a vertex v_i and was adapted to the static DARP by Cordeau and Laporte (2003) in the evaluation scheme mentioned before.

The forward time slack at a certain vertex v_i is the minimum slack of all the vertices that go from vertex v_i to the last vertex of the route. Having an ordered route $k = (v_1, \ldots, v_i, \ldots, v_q)$, the forward time slack F_i of vertex v_i is defined as

$$F_i = \min_{i \leqslant j \leqslant q} \left\{ l_j - \left(D_i + \sum_{i \leqslant p < j} t_{p,p+1} \right) \right\} \tag{2}$$

Considering the fact that

$$D_j = D_i + \sum_{i \leqslant p < j} t_{p,p+1} + \sum_{i < p \leqslant j} W_p \tag{3}$$

Eq. (2) can be rewritten as:

$$F_i = \min_{i \leqslant j \leqslant q} \left\{ l_j - \left(D_j - \sum_{i < p \leqslant j} W_p \right) \right\} = \min_{i \leqslant j \leqslant q} \left\{ \sum_{i < p \leqslant j} W_p + (l_j - D_j) \right\} \tag{4}$$

Therefore, the forward time slack represents the largest increase in the departure time at vertex v_i that will not cause any time window violation. As it will be seen later, the proposed algorithm allows the existence of infeasible solutions. Consequently, the term $(l_j - D_j)$ should be replaced with $(l_j - D_j)^+$ where $x^+ = max\{0, x\}$ so that if a violation of the time window occurs, this violation does not get incremented.

Moreover, when delaying the departure time at vertex v_i, attention must be paid to avoid the possible ride time violation that could happen for a request whose origin vertex is before v_i and whose destination vertex is at or after v_i. As a result, Eq. (4) becomes:

$$F_i = \min_{i \leqslant j \leqslant q} \left\{ \sum_{i < p \leqslant j} W_p + min\{(l_j - D_j)^+, (Lmax_j - R_j)^+\} \right\} \tag{5}$$

where v_q denotes the last vertex on the route and R_j the ride time of the user whose destination is $j \in n+1, \ldots, 2n$ given that v_{j-n} is visited before v_i on the route and $R_j = 0$ for all other j.

Note that delaying the departure time from a vertex v_i by $\sum_{i<p<q} W_p$ does not affect the arrival time A_q at the end of the route whereas delaying it more would simply increase A_q by as much. As a result, the departure from a node should only be delayed by at most $min\{F_i, \sum_{i<p<q} W_p\}$.

The forward time slack concept lead Cordeau and Laporte (2003) and Parragh et al. (2010) to an evaluation scheme that computes the A_i,W_i,D_i values for each vertex on the route and then, tries to delay the departure time at the first vertex in order to reduce the ride time of the affected vertices (the first ones and the subsequent ones that come before the associated delivery vertex v_{1+n}). Then, if a violation of the ride duration of a request is detected, every vertex that is an origin is delayed until the detected violation is resolved. The whole evaluation process is described in Algorithm 1.

Algorithm 1. Evaluation scheme

1: Set $D_1 = e_1$
2: Compute A_i,W_i,D_i and y_i for each vertex i on the route.
3: **if** some $D_i > l_i$ or $y_i > Q$ **then**
4: GOTO step 15
5: **end if**
6: Compute F_1
7: Set $D_1 = e_1 + min\left\{F_1, \sum_{1<p<q} W_p\right\}$
8: Update A_i,W_i and D_i for each vertex v_i on the route
9: Compute L_i for each request on the route. If all $L_i \leqslant Lmax_i$ GOTO step 15
10: For every vertex j that is an origin:
11: Compute F_j
12: Set $W_j = W_j + min\{F_j, \sum_{j<p<q} W_p\}$ and $D_j = A_j + W_j$
13: Update A_i,W_i and D_i for each vertex i that comes after j in the route
14: Update L_i for each request i whose destination is after j. If all $L_i \leqslant Lmax_i$ of requests whose destination lie after j GOTO step 15
15: Compute changes in violations of vehicle load, duration, time window and ride time constraints.

5. Description of the algorithm

As mentioned in Section 1, the proposed static DARP has been solved by means of a VNS-based algorithm. The general idea of this algorithm is to start with an initial solution (being it also the first incumbent solution s). Then, in every iteration, a neighborhood class (or shaker method) is used to generate a random solution s' in the neighborhood defined by the method $N_k(s)$ whose neighborhood size is defined by k. In the next step, a local search (LS) algorithm is applied to s', yielding s''. If s'' is better than s, it replaces s and k is set to the first possible neighborhood. If s'' is worse, s is not replaced, incrementing k so that subsequent iterations use the next possible neighborhood. Whenever the maximum number of neighborhoods k_{max} is reached, the search continues with the first neighborhood. This whole process is repeated until a stopping criterion is satisfied.

In this work, we have implemented a generalized version of the modifications proposed by Parragh et al. (2010) to the general VNS scheme. Therefore, deteriorating solutions may be accepted as incumbent with a certain probability. Moreover, intermediate infeasible solutions can also become incumbent solutions. As a result, it is necessary to keep track of the best feasible solution s_{best} found along the optimization process. Algorithm 2 presents the main algorithm steps whereas each design element is described in further detail in the following sections.

Algorithm 2. VNS algorithm

1: generate s_{init}
2: $s = s_{init}$; $k = 1$
3: $t = 0$
4: $\delta = random(mindelta,maxdelta)$
5: $\alpha = \beta = \gamma = initpenalization$
6: **while** the stopping criterion is not satisfied **do**
7: //shaking
8: randomly compute s' with $N_k(s)$
9: // local search
10: **if** $c(s') < lsvalue1 \cdot c(s)$ or $p_{rand} < lsprobvalue$ **then**
11: Use the local search method over s' to create s''
12: **else**
13: $s'' = s'$
14: **end if**
15: // Move or not
16: **if** t is 0 and s_{best} is feasible **then**
17: $t = t_{init} = \frac{t_{initratio} * f(s_{best})}{ln\left(\frac{1}{t_{initprob}}\right)}$
18: $t_{stepratio} = t/\#maxevals$
19: **end if**
20: $p_{SA} = e^{-\left(\frac{f(s'') - f(s_{best})}{t}\right)}$
21: **if** $f(s'') < f(s)$ or $p_{rand} < p_{SA}$ **then**
22: **if** $c(s'') \geqslant lsvalue2 \cdot c(s)$ **then**
23: Use the local search method to s''
24: $s = s''; k = 0$
25: // Update penalty parameters
26: **for** each associated penalty term *penterm* in α,β and γ **do**
27: **if** s violates the corresponding constraint of *penterm* (max pickup time, max ride duration or max load) **then**
28: $penterm = penterm*(1 + \delta)$
29: **else**
30: $penterm = penterm/(1 + \delta)$
31: **end if**
32: $\delta = random(mindelta,maxdelta)$
33: **end for**
34: **end if**
35: **end if**
36: **if** s'' is feasible and better than s_{best} **then**
37: $s_{best} = s''$
38: **end if**
39: $k = (k \ mod \ k_{\max}) + 1$
40: $t = t_{init} - (t_{stepratio} * \#evalcalls)$
41: **end while**
42: **return** s_{best}

5.1. Initialization

For the initialization, a different approach than those followed in Cordeau and Laporte (2003) and Parragh et al. (2010) has been used. As it will be seen in the following sections, due to the difficulty of the proposed problems, it is crucial to start with a feasible (or close to feasible) solution in the high dimensional problems in order to be able to improve it along the optimization process.

The original initialization processes of Cordeau and Laporte (2003) and Parragh et al. (2010) did not take into account any of the possible violations and tried to exploit the information of the spatial relationships (Parragh et al., 2010) or use a completely random approach (Cordeau & Laporte, 2003). Here, we propose a method that incrementally builds a solution by selecting, at each step, the request that obtains the best evaluation function value. To avoid the construction of solutions that do not satisfy the constraints, each infeasible insertion is heavily penalized. The whole initialization process is described as follows:

- First, all the requests are sorted according to their pickup requested times.
- Then, all routes are initialized with one request each, using the first m requests of the list.
- The following request of the list is evaluated on all the routes, inserting it in the route that obtains the best evaluation value and that does not violate any constraint. If no route is found that does not violate any constraint, the request is inserted in the route that minimizes the violations values. Each request is inserted as follows: First, the pickup vertex of the respective request is inserted at its best position of all the possible positions that are compatible with the time window values. Then, the delivery vertex is inserted at its best position in accordance with the pickup one. This process is repeated in order until all requests of the list have been inserted into one route.
- Once all the requests have been inserted, each route undergoes the LS search procedure described in Section 5.3

5.2. Neighborhood classes

Seven different neighborhood classes (or shakers) have been proposed for the algorithm. Two of them were previously defined in Parragh et al. (2010) whereas the remaining five have been defined specifically for this work. Most of the shakers can be parametrized by a size value which determines the maximum number of requests (or routes) that can be modified at each application of the shaker. They are detailed below:

Swap neighborhood (S): This shaker, proposed in Parragh et al. (2010), exchanges a number of requests between two routes. First, two different routes are chosen randomly. Then, on each route, a sequence to be swapped is randomly selected: first, the starting vertex for each sequence and then the length. The maximum sequence length is referred to as the size of this neighborhood. Note that for each vertex within each selected sequence, the corresponding origin or destination vertex has to be selected as well even if it is not part of the sequence. Finally, all the requests forming the respective sequences are deleted from their routes and inserted, one-by-one, into the other route. The insertion of each vertex of the sequence follows the same procedure that was described in the initialization process (Section 5.1).

Chain neighborhood (C): The second neighborhood class, also defined in Parragh et al. (2010), applies the ejection chain idea. In this shaker, first, a sequence of vertices, randomly selected as in the swap neighborhood, is moved to a second route. Then, a random length l value is selected (being the maximum value the size of the shaker). From the second route, the sequence from all possible sequences of the selected route (of length l) that improves the most the evaluation function, is moved to a third route (also, randomly selected). This step is repeated until the maximum number of moved sequences (specified by the size of the shaker) is reached. Therefore, the neighborhood size specifies both the maximum number of sequences moved as well as the maximum length of a sequence to be selected. All insertions are done one-by-one following the same procedure described in the initialization process.

Greedy worst origin move neighborhood (GWOM): In this neighborhood class, a sequence of vertices is moved from a random route to a different one. The characteristic aspect of this shaker is that it selects the worst possible sequence of vertices (according to the evaluation function) of the size defined by the neighborhood class, and moves it to the best possible route (for conducting the insertion). For this task, it first computes all the possible sequences

(from the selected route) and computes, one-by-one, the resulting evaluation value of removing them from the route. Then, it selects the sequence that reduces the most the evaluation value, and inserts it in the best possible route from the remaining ones, i.e., in the route which obtains the best final value after conducting the insertion. Similarly to the previous shakers, the insertions are conducted following the same procedure described in the initialization process.

Greedy best destination move neighborhood (GBDM): This neighborhood class follows a similar strategy to the previous one but uses an opposite approach for selecting the sequence to be moved: instead of selecting (from all the possible sequences of specific size), the worst sequence of a selected route, it selects the sequence that obtains the best evaluation value when inserted in the destination route. For selecting the destination route, a tournament roulette selection method (a well-known selection method in GAs) has been applied. Thus, two candidate routes are first selected with a roulette method that uses the inverse of the route evaluation function, i.e., $\frac{1}{f_{route}(s)}$. From these routes, the one with the best route evaluation value is finally selected. This way, the routes with the best (or lower) route evaluation values will have a higher probability of being selected for the destination route. Similarly, an inverse selection process is applied for selecting the origin route, i.e., the routes with the worst values will have a higher probability for being selected as origin routes. The idea for this selection is to move the worst sequences of the worst routes to the best routes where they could potentially have more margin for carrying out the insertion.

Greedy origin swap (GOS): The fifth neighborhood class borrows some of the ideas of the previous GWOM and GBDM shakers and extends them for a swap operation. First, it selects two routes using the same criterion of the GBDM shaker. Then, for each route, it selects the best and worst sequence (of the size specified by the shaker) to be removed. With these four sequences (two per route) it tries all the four possible swaps of sequences, selecting, at the end, the swap operation that obtains the best evaluation value.

Greedy destination swap (GDS): This neighborhood class extends the previous shaker by changing the selection criterion for the best and worst sequences: instead of selecting the best and worst sequences to be *removed* for each route, it selects the best and worst sequences to be *inserted* in the other route. Similarly to the previous shaker, it tests all the four possible swaps, selecting the swap operation that obtains the best result in terms of evaluation value.

All natural sequences combinations neighborhood (ANSC): The final neighborhood class is based on the idea of natural sequences developed for the Zero split neighborhood class described in Parragh et al. (2010). A natural sequence is a sequence of vertices in which the vehicle load at the end is zero. Without considering the trivial sequence of the complete route, it was discovered that routes quite often contain more than only one sequence of this type. Fig. 2 represents this concept graphically. In the original Zero split neighborhood class, a random natural sequence was removed from a route and inserted (following the same insertion procedure described in the initialization phase) into a different randomly selected route. In this new shaker, each natural sequence is treated as a single unit in order to try all the possible swaps of natural sequences between all combinations of pairs of routes. For example, if route i and route j contains two natural sequences respectively, the evaluation of the pair (i,j) would imply the computation of the evaluation value of four possible swaps of natural sequences. Since each natural sequence is treated as a block, the whole sequence is extracted and inserted (as a unit) in the different route, reducing, considerably, the number of evaluation function calls per swap. In our preliminary experiments, this approach obtained significantly better results than the Zero split neighborhood, requiring, for each call, a significantly fewer number of evaluation function calls. Finally, contrary to the other shakers, this shaker has no size parameter since it always conducts the same operations.

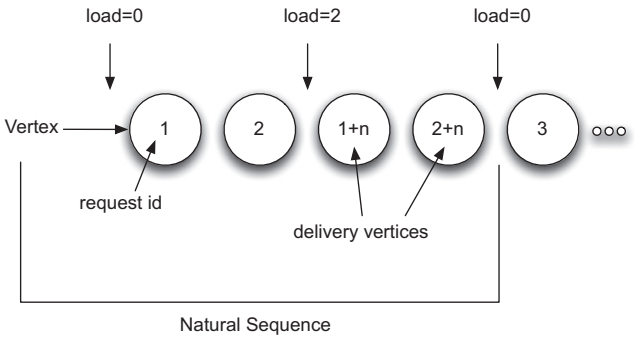

Fig. 2. An example of a natural sequence.

The aforementioned shakers can be applied in any order and with different neighborhood sizes (except for the ANSC shaker). Any sequence could be developed[1] but, to follow the philosophy of the VNS algorithm, the shakers that carry out less perturbations should be applied first. For example, the sequence: S1-C1-S2 would execute first the swap neighborhood class with a size of one unit, followed by the chain neighborhood of size 1 and ending the sequence with the swap neighborhood class of size 2.

5.3. Local search

Whereas the preceding neighborhood classes focus their efforts in conducting inter-tour perturbations, the LS conducts a greedy approach based on intra-tour modifications to obtain the best sequence of vertices for each route. This LS, based on the algorithm proposed in Parragh et al. (2010), is applied to every route as follows: first, it removes the pickup vertex, and its corresponding delivery vertex. Then, it inserts the pickup vertex at the first possible position (according to the time window values). Thereafter, the delivery vertex is inserted at the first possible position with respect to the recently inserted pickup vertex. If this insertion improves the evaluation value, the LS continues with the following pickup vertex on the route. Otherwise, the delivery vertex is inserted at the next available position. This process is repeated until there is either an improvement, or no other insertion position for the delivery vertex is found. In this case, the pickup vertex is moved to the next position, repeating the last two steps. If no improvement is found along this process, the pair of vertices is kept at its original position and the whole process is repeated with the following pickup vertex of the route. Once a route has been optimized, the described algorithm is applied to the next route until all the routes have been adjusted.

5.3.1. Local search frequency

Since the LS consumes a considerable number of function evaluations, instead of conducting the LS step after every shaking step (as the canonical VNS algorithm recommends), the LS is only used with promising solutions. Note that most of the shakers use a greedy insertion algorithm that includes in itself some kind of local search so it seems reasonable to avoid unnecessary calls to this function.

A promising solution is a solution that could potentially become a new incumbent solution. Since the objective of our approach is to minimize the total cost $c(s)$, a promising solution has been characterized by $c(s') < lsvalue1 \cdot c(s)$, where $lsvalue1$ estimates the range

[1] As it will be seen in the experimentation, several sequences have been actually tested in this work.

of values for a solution to be considered promising. To introduce another element of diversification, every solution has a *lsprobvalue* probability chance to be subject to a LS improvement phase. Moreover, Parragh et al. (2010), introduced another point for using the LS: every solution s'' which meets the acceptance criteria (described in detail in Section 5.4), can undergo a LS process if its $c(s)$ value is only worse than a percentage (defined by *lsvalue*2) of the current incumbent solution. Here, the idea is to promote the diversification since a solution that has not been locally optimized may, in some cases, provide better options regarding the removal and reinsertion of a request in the subsequent iteration than a locally optimized one.

5.4. Acceptance criterion

The algorithm proposes an acceptance criterion using a simulated annealing type approach for deciding whether the incumbent solution should move to the new solution s'' or not. In the beginning, the new solution can only replace the actual solution if its evaluation value is better. Once the first feasible solution is found, deteriorating solutions may become incumbent solutions with probability $e^{-((f(s'')-f(s_{best}))/t)}$. t is linearly decreased at each step based on the initial temperature (t_{init}) computed when the first feasible solution is found. In Parragh et al. (2010), the initial temperature was set such that if $f(s'')/f(s_{best}) - 1 = 0.005$, s'' is accepted with a probability of 0.2. In this work, we have generalized these values in order to conduct several tests with alternative values. The variables $t_{initratio}$ and $t_{initprob}$ represent the 0.005 and 0.2 values in Parragh et al. (2010) whereas *#maxevals* represents the maximum number of calls to the evaluation function and *#evalcalls* the actual number of calls.

5.5. Update of the penalization terms

Finally, the penalization terms for the maximum pickup time, maximum ride duration and maximum load violations (α, β and γ) are dynamically adjusted every time an incumbent solution is identified. In case the incumbent solution violates a constraint, its corresponding term is increased by the product factor $(1+\delta)$. Otherwise, it is decremented by dividing it by $(1+\delta)$ as specified in Algorithm 2. Moreover, in order to reduce cycling, every time a new incumbent solution is found, this value is randomly chosen between *mindelta* and *maxdelta*.

6. Experimental scenario

In this section, the selected problem instances are described in detail. Then, the followed approach for selecting the values for the parameters as well as the tuning method is analyzed. Finally, the selection of the algorithms for the comparison of the results is justified.

6.1. Problem instances

For analyzing the results of the proposed algorithm, four different problems, having three 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 believable user demand patterns requested by the potential customers of the service provided by this company.[2]

For each instance, two different scenarios have been proposed: a medium-scaled scenario consisting of 100 different requests and a large-scaled scenario containing 1000 different requests. Therefore, a total of 24 different instances have been optimized for this study.[3] The main characteristics of each of the proposed problems are described below whereas Fig. 3 depicts these characteristics graphically:

- Carnaval problem (C): This problem represents the demand that could be generated during the San Francisco Carnaval Festival. For this problem two types of demands have been simulated: the requests that could arise from any address to several stops around the Carnaval area and the requests that could demand a transport from the Carnaval area stops to any other address. For this task, 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. Moreover, a normal distribution centered at 18:00 has been used to represent the increase of requests at the end of the Carnaval. The standard deviation has been defined so that 99% of the requests are generated in a half an hour range (centered around the aforementioned values). The distribution of the requests has been set so that 90% of the requests are generated by the normal distributions whereas the remaining 10% by the uniform distributions.
  To represent the possible locations that could be demanded, a discretization based on pickup points, similar to Raghavendra, Krishnakumar, Muralidhar, Sarvanan, and Raghavendra (1992), has been used. For this study, the set of transit stops available from the San Francisco Municipal Transportation Agency (SFMTA) has been used. This way, the city has been discretized according to the real disposition of the stops used by the municipal transportation system. Eq. (6) 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.

$$Carnaval \sim \begin{cases} U(630,1020) & \text{Bus stops} - \text{Carnaval stops}(5\%) \\ N(630,5) & \text{Bus stops} - \text{Carnaval stops}(22.5\%) \\ U(720,1140) & \text{Carnaval stops} - \text{Bus stops}(5\%) \\ N(885,5) & \text{Bus stops} - \text{Carnaval stops}(22.5\%) \\ N(1080,5) & \text{Carnaval stops} - \text{Bus stops}(45\%) \end{cases} \quad (6)$$

- 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 homes 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 (i.e. the set of transit stops obtained from the SFMTA) has been used. As presented in Eq. (7), a uniform distribution set in the range 8:00–20:00 has been used to simulate the times of the requests.

$$Hospitals \sim U(480,1259)\ \text{Hospitals stops} - \text{Bus stops} \quad (7)$$

- Hotels problem (HT): This problem simulates the requests that could be generated from/to a set of hotels from different sets of

[2] For obtaining the costs between a pair of points, a Geographic information system (GIS) with the San Francisco cartography has been used.

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

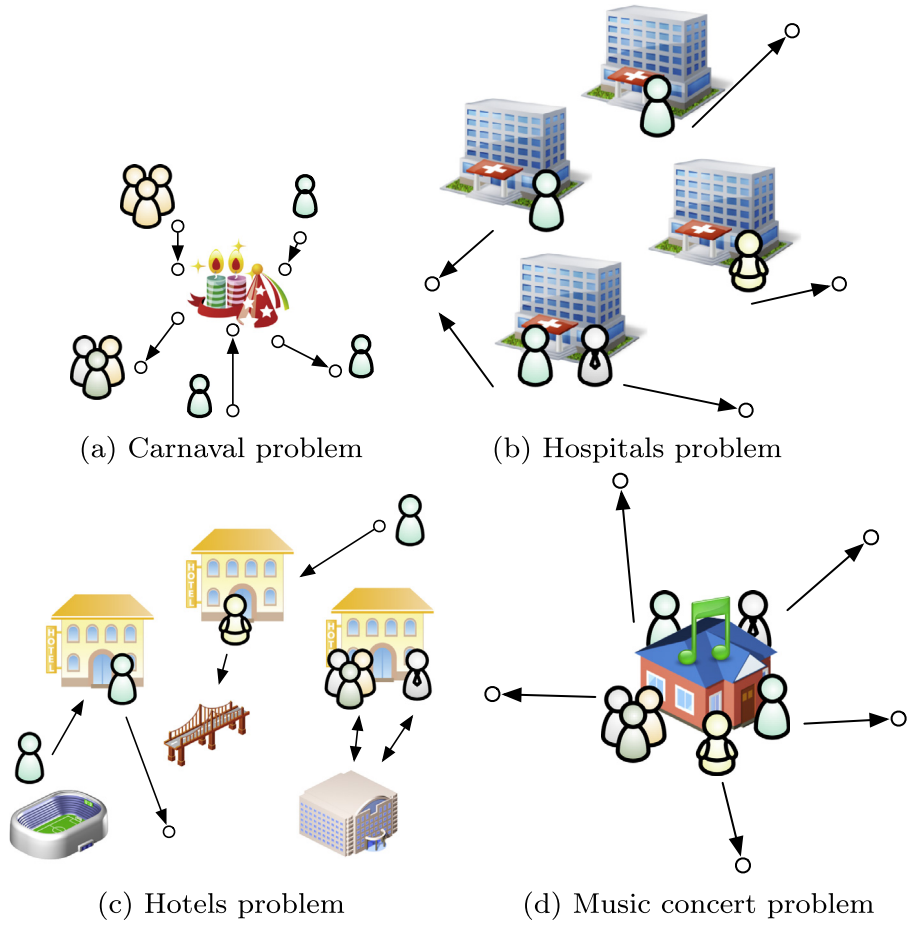

Fig. 3. Depiction of the proposed problems.

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 (09:00–23:00). 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. (8) briefly represents the generation of this problem

$$Hotels \sim \begin{cases} U(480,1320) & \text{Hotels stops} - \text{Bus stops}(5\%) \\ N(510,5) & \text{Hotels stops} - \text{Convention centers stops}(35\%) \\ U(540,1200) & \text{Hotels stops} - \text{Attractions stops}(10\%) \\ U(540,1439) & \text{Bus stops} - \text{Hotels stops}(5\%) \\ U(600,1260) & \text{Attractions stops} - \text{Hotels stops}(10\%) \\ N(1140,5) & \text{Convention centers stops} - \text{Hotels stops}(35\%) \end{cases} \quad (8)$$

- Music concert problem (M): In this problem, the demand that could arise from the end of a music concert has been represented. Therefore, and in contrast to the other problems, a large amount of requests are going to collide in the same time frame, with the same origin address and with a different destination (represented by the set of addresses of the transit stops obtained from the SFMTA). This problem, as shown in Eq. (9), has been 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.

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

Table 2
Number of available vehicles.

Problem	#Requests	Instance	#Vehicles
Carnaval	100	i1	9
Carnaval	100	i2	9
Carnaval	100	i3	9
Hospitals	100	i1	8
Hospitals	100	i2	5
Hospitals	100	i3	5
Hotels	100	i1	10
Hotels	100	i2	11
Hotels	100	i3	10
Music	100	i1	11
Music	100	i2	11
Music	100	i3	12
Carnaval	1000	i1	65
Carnaval	1000	i2	65
Carnaval	1000	i3	64
Hospitals	1000	i1	42
Hospitals	1000	i2	42
Hospitals	1000	i3	42
Hotels	1000	i1	37
Hotels	1000	i2	35
Hotels	1000	i3	36
Music	1000	i1	87
Music	1000	i2	89
Music	1000	i3	87

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.[4] 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 15 passengers and its number has been adjusted based on the problem characteristics (Table 2 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.

6.2. Parameter values

To properly tune the proposed algorithm, every constant that appeared on the original description of the algorithm in Parragh et al. (2010), has been parametrized and tested on a set of different values. Since no knowledge of the suitable range of values could be determined for each parameter, the selected criterion has analyzed three values for each parameter: the original value proposed in Parragh et al. (2010) as well as the corresponding half and double values. For the *maxdelta* and *mindelta* parameters, the set of values have been selected in order to avoid ranges of a single value when using both parameters.

6.2.1. Shakers schemes

To select a good sequence of neighborhood classes, a VNS algorithm containing all the shakers described in Section 5.2 with varying neighborhood sizes from one to six was executed in all the proposed problems and #requests (25 executions per problem and dimension). With the purpose of ranking the shakers the following performance measure was computed for each shaker: number of improvements to the solution divided by the number of evaluations consumed (in all the proposed executions). Table 3 presents these results along with the cumulative proportion for each position. It can be seen, for example, that the best results have been obtained by the Chain neighborhood shaker of size 1 followed by the same shaker of size two.

Table 3
Overall results of the performance of the shakers.

Shaker	#Improvements	#Evals	Improvements per feval	Cum. Value
C1	1.8680E + 03	7.6585E + 05	2.4397E − 03	0.43
C2	1.6650E + 03	2.5393E + 06	6.5568E − 03	0.54
S1	5.9900E + 02	1.4103E + 06	4.2473E − 04	0.62
GWOM2	9.5620E + 03	2.6543E + 07	3.6024E − 04	0.68
GBDM2	2.2410E + 03	7.8876E + 06	2.8411E − 04	0.73
S2	4.9500E + 02	2.0424E + 06	2.4236E − 04	0.78
GOS2	9.5600E + 02	5.1655E + 06	1.8507E − 04	0.81
S3	3.1000E + 02	1.6972E + 06	1.8265E − 04	0.84
ANSC	5.0760E + 03	2.8147E + 07	1.8033E − 04	0.87
S4	2.7300E + 02	2.0647E + 06	1.3222E − 04	0.90
C3	8.1000E + 02	7.0325E + 06	1.1151E − 04	0.92
S5	2.1600E + 02	2.0878E + 06	1.0345E − 04	0.93
S6	2.3200E + 02	2.3801E + 06	9.7473E − 05	0.95
GWOM4	2.8260E + 03	4.3062E + 07	6.5626E − 05	0.96
C4	6.5500E + 02	1.0311E + 07	6.3525E − 05	0.97
GDS2	9.9800E + 02	1.7390E + 07	5.7387E − 05	0.98
GOS4	3.1200E + 02	1.3553E + 07	2.3021E − 05	0.99
GWOM6	1.9200E + 03	1.0384E + 08	1.8489E − 05	0.99
C5	3.5800E + 02	4.9404E + 07	7.2464E − 06	0.99
GBDM4	6.9600E + 02	1.1360E + 08	6.1266E − 06	0.99
C6	2.1900E + 02	6.2820E + 07	3.4861E − 06	0.99
GOS6	1.7600E + 02	6.1836E + 07	2.8462E − 06	1.00
GDS4	1.8100E + 02	1.7243E + 08	1.0497E − 06	1.00
GBDM6	3.7700E + 02	1.3176E + 09	2.8612E − 07	1.00
GDS6	7.1000E + 01	1.5615E + 09	4.5469E − 08	1.00

Based on these results, three different schemes were proposed: (i) the complete set of shakers used in these experiments, (ii) the subset of shakers that accumulate the 90% of the total value of the proposed measure, i.e., S1-C1-S2-C2-GWOM2-GBDM2-GOS2-S3-S4-ANSC and (iii) the same set of shakers of the previous case but sorted according to the proposed measure instead of following the VNS philosophy where the shakers are sorted according to its neighborhood size, i.e., C1-C2-S1-GWOM2-GBDM2-S2-GOS2-S3-ANSC-S4.

6.2.2. Stopping criterion

Each algorithm has been executed until a fixed number of route evaluations is consumed. Since different algorithms have been compared in the following sections, the selection of the number of iterations as a stopping criterion would have created unfair comparisons. Furthermore, the number of calls to the evaluation function is also an inappropriate criterion due to the fact that some perturbation methods (e.g. shakers) tend to modify more routes per execution than others, having more routes to recompute their window, duration and load values. This criterion approximates reasonably well the expected execution time since the time spent in the route evaluation function represents the 95% of the overall execution time of the proposed algorithms.

In particular, for all the tests carried out in this work, the stopping criterion has been set to $20000 \times \#requests$ calls to the route evaluation function. Furthermore, due to the stochasticity behavior of the algorithms, 25 executions have been conducted per algorithm.

6.3. Parameter tuning

For the experimentation a fractional design based on orthogonal matrices according to the Taguchi method (Taguchi, Chowdhury, & Wu, 2005) was chosen in order 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

[4] Which is far less than the absolute threshold of 90 minutes that was set in the benchmark of Cordeau and Laporte (2003).

Table 4
Parameters values tested. The final selected values by the Taguchi method are marked in bold.

Parameter	Values
tinitratio	0.0025, 0.005 and **0.01**
tinitprob	0.1, **0.2** and 0.4
lsvalue1	**1.01**, 1.02 and 1.04
lsprobvalue	0.005, **0.01** and 0.02
lsvalue2	1.025, **1.05** and 1.1
min delta	**0**, 0.05 and 0.1
max delta	0.1, 0.2 and **0.4**
shakersScheme	first, second and **third** as defined in Section 6.2.1

values. In particular, 27 different configurations were tested for the whole set of problems defined in Section 6.1 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. (10) 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).

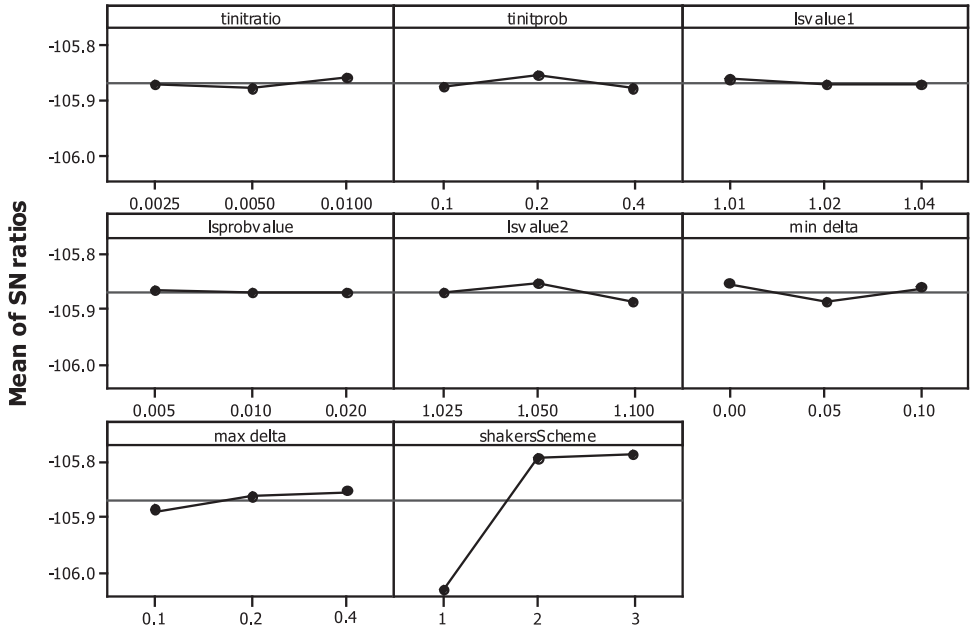

Fig. 4. Main effects plot for SN ratios.

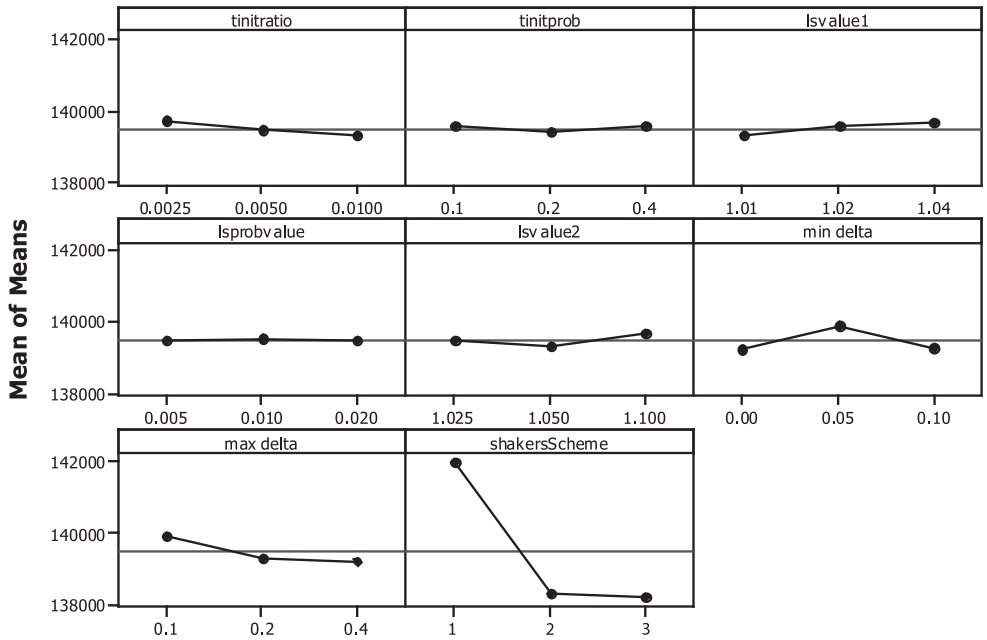

Fig. 5. Main effects plot for means.

Table 5
Algorithms selected for conducting the tests.

ID	Algorithm	Initialization
VNSN	VNS (New proposal)	New
VNSP-1	VNS (Parragh et al.)	Original
VNSP-2	VNS (Parragh et al.)	New
TS-1	TS (Cordeau and Laporte)	Original
TS-2	TS (Cordeau and Laporte)	New

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

Figs. 4 and 5 and 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 parameters is more determinant than others. For example, the values used for the *lsprobvalue* parameter have obtained very similar results whereas in the *shakersScheme* parameter, the selection of the first scheme (the one that uses all the shakers) has dramatically affected the performance of the algorithm. It is also worth to mention that in the *shakersScheme* parameter, the best results have been obtained by the scheme that does not follow the VNS philosophy but instead tries to apply each shaker based on the results of a performance measure. 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.

6.4. Comparison with other algorithms

Two of the best algorithms from the literature, the TS of Cordeau and Laporte (2003) and the VNS of Parragh et al. (2010), proposed for a similar DARP problem have been implemented and tested in the same benchmark (TS-1 and VNSP-1). Furthermore, in view of the strong influence in the final performance of the initialization method, these algorithms have also been executed with the proposed initialization process (TS-2 and VNSP-2). The final selection of algorithms for conducting the tests is displayed in Table 5.

7. Results and discussion

The algorithms proposed in Table 5 were executed for a set of 24 different problems (three different instances on each of the four different scenarios with 100 requests and another 12 instances with 1000 requests) and for 25 executions each. Tables 6 and 7 present, for each algorithm, the mean values (of the 25 executions) obtained by each algorithm for the evaluation function as well as the sum of pickup and ride times. If an algorithm is unable to obtain a feasible solution, this fact is represented in the table with the abbreviation inf.

Table 6
Mean values in 100-D.

Alg.	Key	C-1	C-2	C-3	HS-1	HS-2	HS-3
VNSN	$f(s)$	**2.3903E + 04**	**2.4090E + 04**	2.5939E + 04	**5.3184E + 04**	5.0890E + 04	**5.1471E + 04**
	$\sum_i P_i$	4.3844E + 04	4.1711E + 04	4.4225E + 04	4.3600E + 04	4.8762E + 04	5.0409E + 04
	$\sum_i L_i$	6.6912E + 04	6.7841E + 04	6.6318E + 04	7.6167E + 04	7.3580E + 04	7.7660E + 04
VNSP-1	$f(s)$	2.7237E + 04	2.6828E + 04	2.8484E + 04	5.5577E + 04	5.2986E + 04	5.3307E + 04
	$\sum_i P_i$	4.5881E + 04	4.3725E + 04	4.2897E + 04	4.6331E + 04	4.9810E + 04	5.1786E + 04
	$\sum_i L_i$	6.9184E + 04	7.0068E + 04	6.7902E + 04	7.6575E + 04	**7.2252E + 04**	**7.7637E + 04**
VNSP-2	$f(s)$	3.0904E + 04	3.0131E + 04	3.0834E + 04	5.8256E + 04	5.5746E + 04	5.7945E + 04
	$\sum_i P_i$	4.4573E + 04	4.3909E + 04	4.4431E + 04	**4.1716E + 04**	4.7611E + 04	**4.7681E + 04**
	$\sum_i L_i$	6.8114E + 04	7.0596E + 04	6.7847E + 04	7.4889E + 04	7.2821E + 04	7.8663E + 04
TS-1	$f(s)$	2.8475E + 04	2.8829E + 04	2.9797E + 04	5.4447E + 04	5.1404E + 04	5.1998E + 04
	$\sum_i P_i$	**4.2842E + 04**	4.1614E + 04	**4.1118E + 04**	4.5473E + 04	**4.6631E + 04**	5.1736E + 04
	$\sum_i L_i$	6.6862E + 04	6.8192E + 04	**6.4115E + 04**	7.4974E + 04	7.3016E + 04	7.8011E + 04
TS-2	$f(s)$	2.4645E + 04	2.4294E + 04	**2.5766E + 04**	5.4014E + 04	**5.0752E + 04**	5.1971E + 04
	$\sum_i P_i$	4.5718E + 04	**4.1001E + 04**	4.4558E + 04	4.4215E + 04	4.9206E + 04	5.0650E + 04
	$\sum_i L_i$	**6.5988E + 04**	**6.7379E + 04**	6.6016E + 04	**7.4054E + 04**	7.2761E + 04	7.9037E + 04
Alg.	Key	HT-1	HT-2	HT-3	M-1	M-2	M-3
VNSN	$f(s)$	2.0143E + 04	1.7197E + 04	2.0530E + 04	**1.5189E + 04**	**1.5554E + 04**	**1.5312E + 04**
	$\sum_i P_i$	4.1446E + 04	4.0474E + 04	4.3860E + 04	4.4022E + 04	**4.4294E + 04**	4.4424E + 04
	$\sum_i L_i$	4.5710E + 04	3.7416E + 04	4.7402E + 04	**6.6627E + 04**	**6.7854E + 04**	**7.2651E + 04**
VNSP-1	$f(s)$	2.2284E + 04	1.9374E + 04	2.2705E + 04	inf.	inf.	inf.
	$\sum_i P_i$	4.2266E + 04	4.1927E + 04	4.5100E + 04	inf.	inf.	inf.
	$\sum_i L_i$	4.6946E + 04	4.0197E + 04	4.9102E + 04	inf.	inf.	inf.
VNSP-2	$f(s)$	2.2446E + 04	1.8677E + 04	2.3091E + 04	3.0427E + 04	3.0856E + 04	4.5990E + 04
	$\sum_i P_i$	4.1036E + 04	**3.8765E + 04**	**4.2992E + 04**	5.3231E + 04	5.3240E + 04	6.9151E + 04
	$\sum_i L_i$	4.6778E + 04	3.8478E + 04	4.9700E + 04	8.4988E + 04	8.4915E + 04	1.0712E + 05
TS-1	$f(s)$	2.3749E + 04	2.0653E + 04	2.3597E + 04	inf.	inf.	inf.
	$\sum_i P_i$	**4.0811E + 04**	3.9239E + 04	4.4732E + 04	inf.	inf.	inf.
	$\sum_i L_i$	**4.4469E + 04**	3.7693E + 04	**4.6268E + 04**	inf.	inf.	inf.
TS-2	$f(s)$	**1.9739E + 04**	**1.6805E + 04**	**2.0348E + 04**	1.5842E + 04	3.5557E + 04	1.6579E + 04
	$\sum_i P_i$	4.1345E + 04	4.0103E + 04	4.4001E + 04	**4.2425E + 04**	6.0646E + 04	**4.3324E + 04**
	$\sum_i L_i$	4.5216E + 04	**3.6739E + 04**	4.6715E + 04	6.7850E + 04	8.8362E + 04	7.4562E + 04

inf. means an infeasible solution.

Table 7
Mean values in 1000-D.

Alg.	Key	C-1	C-2	C-3	HS-1	HS-2	HS-3
VNSN	$f(s)$	**2.0327E + 05**	**1.9937E + 05**	**2.0487E + 05**	**4.3751E + 05**	**4.4327E + 05**	**4.3996E + 05**
	$\sum_i P_i$	**4.4881E + 05**	**4.5210E + 05**	**4.5899E + 05**	4.7771E + 05	4.7455E + 05	4.7827E + 05
	$\sum_i L_i$	**7.1993E + 05**	**6.8218E + 05**	**7.0915E + 05**	**8.3767E + 05**	**8.3023E + 05**	**8.2386E + 05**
VNSP-1	$f(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
VNSP-2	$f(s)$	2.4803E + 05	2.3735E + 05	2.4917E + 05	4.8579E + 05	4.9113E + 05	4.9101E + 05
	$\sum_i P_i$	4.6515E + 05	4.6147E + 05	4.7197E + 05	4.7471E + 05	4.7294E + 05	4.7992E + 05
	$\sum_i L_i$	7.7419E + 05	7.2788E + 05	7.6100E + 05	8.4671E + 05	8.3910E + 05	8.3358E + 05
TS-1	$f(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
TS-2	$f(s)$	2.4757E + 05	2.3804E + 05	2.4959E + 05	4.7844E + 05	4.8569E + 05	4.8307E + 05
	$\sum_i P_i$	4.7037E + 05	4.6646E + 05	4.7507E + 05	**4.6901E + 05**	**4.6303E + 05**	**4.6867E + 05**
	$\sum_i L_i$	7.7477E + 05	7.3110E + 05	7.6200E + 05	8.4516E + 05	8.3501E + 05	8.2879E + 05
Alg.	Key	HT-1	HT-2	HT-3	M-1	M-2	M-3
VNSN	$f(s)$	**1.7217E + 05**	**1.7497E + 05**	**1.7843E + 05**	**1.6091E + 05**	**1.6368E + 05**	**1.6703E + 05**
	$\sum_i P_i$	4.7062E + 05	**4.6206E + 05**	**4.6996E + 05**	**4.2898E + 05**	**4.2199E + 05**	**4.2825E + 05**
	$\sum_i L_i$	**4.9709E + 05**	**4.9605E + 05**	**4.8777E + 05**	**6.6531E + 05**	**6.5665E + 05**	**6.6757E + 05**
VNSP-1	$f(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
VNSP-2	$f(s)$	1.7307E + 05	1.7627E + 05	2.1717E + 05	1.7367E + 05	1.7765E + 05	1.8742E + 05
	$\sum_i P_i$	4.7247E + 05	4.6521E + 05	5.0852E + 05	4.3317E + 05	4.3327E + 05	4.4084E + 05
	$\sum_i L_i$	5.0354E + 05	5.0282E + 05	5.3184E + 05	7.1554E + 05	7.1767E + 05	7.3010E + 05
TS-1	$f(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
TS-2	$f(s)$	1.7334E + 05	1.7590E + 05	1.9063E + 05	1.7158E + 05	1.7852E + 05	1.9373E + 05
	$\sum_i P_i$	**4.6699E + 05**	4.6918E + 05	4.7856E + 05	4.3199E + 05	4.3379E + 05	4.5044E + 05
	$\sum_i L_i$	5.0307E + 05	5.0467E + 05	5.0839E + 05	7.1565E + 05	7.2003E + 05	7.3063E + 05

inf. means an infeasible solution.

Table 8
Average ranking of the mean values.

	Ranking
VNSN	1.20
TS-2	2.04
VNSP-2	2.75

Table 9
Statistical validation for the mean values (VNSN is the control algorithm).

VNSN vs.	Wilcoxon p-value
VNSP-2	$5.96E-08^{\surd}$
TS-2	$1.23E-05^{\surd}$
Wilcoxon p-value with FWER: VNSN vs. VNSP-2, TS-2	$1.24-05^{\surd}$

$\surd$ means that there are statistical differences with significance level $\alpha = 0.05$.

First, the ability to generate feasible solutions was analyzed. From the aforementioned tables, it can be seen that the proposed initialization method is the only method that is able to obtain a 100% value of feasible solutions for both 100 and 1000 requests. Furthermore, due to the special characteristics of the problems, the original initialization method of the VNSP-1 algorithm is unable to find a feasible solution for the Music concert problem. This initialization method constructs the initial set of routes based on the spatial relationships of the requests. If there are several requests with similar origin and destination, this method has the disadvantage that tends to group all the related requests in a single route, creating considerably large routes that are, in general, harder to optimize (and consume more evaluations per local search call) than the solutions obtained by a random approach (as happens in the TS-1 algorithm). This effect is dramatically enlarged in the 1000 requests problems, where the initialization methods of the VNSP-1 and TS-1 algorithms are unable to obtain a single feasible solution. Due to the high dimensionality of these problems, it is crucial to start with a feasible (or almost feasible solution). Otherwise, the algorithm is not capable of constructing a satisfactory solution.

The second analysis consisted in comparing the overall results obtained in all the problems. As previously mentioned, Tables 6 and 7 display the results of the evaluation function as well as the sum of the pickup ($\sum_i P_i$) and ride time ($\sum_i L_i$) values for 100 and 1000 requests, respectively. For each algorithm and problem, the best results for each measure are marked in bold. Although the evaluation function is proposed to optimize the cost of the route (while satisfying the proposed constraints), the pickup and ride time values have also been included in the table in order to have a better insight of the service provided to the clients.

The first thing that can be observed from these results, is that for the HS problem, the algorithms have obtained higher values than in the other problems. This results seems logical since the requests in the HS problem are more sparsed in time than in the other proposed problems and, therefore, it is harder to group the clients in the same route in order to avoid traversing the same path several times. The next pattern that can be observed is that, in general, all the solutions from the proposed algorithms have higher L_i values than their corresponding P_i values and that this behavior is independent of the initialization method, algorithm and evaluation

Table 10
Best values in 100-D.

Alg.	Key	C-1	C-2	C-3	HS-1	HS-2	HS-3
VNSN	$f(s)$	**2.2845E + 04**	**2.2546E + 04**	2.4676E + 04	**5.2510E + 04**	**4.9403E + 04**	**4.9880E + 04**
	$\sum_i P_i$	3.6970E + 04	3.6792E + 04	3.6045E + 04	3.7334E + 04	3.7922E + 04	4.1838E + 04
	$\sum_i L_i$	**6.2046E + 04**	**6.2446E + 04**	6.1149E + 04	7.0998E + 04	7.0005E + 04	7.4423E + 04
VNSP-1	$f(s)$	2.5644E + 04	2.4701E + 04	2.7119E + 04	5.4070E + 04	5.1234E + 04	5.1669E + 04
	$\sum_i P_i$	3.8660E + 04	3.5707E + 04	3.5675E + 04	2.9928E + 04	4.0885E + 04	4.0596E + 04
	$\sum_i L_i$	6.2173E + 04	6.6236E + 04	6.3904E + 04	7.1061E + 04	6.7443E + 04	7.4083E + 04
VNSP-2	$f(s)$	2.4887E + 04	2.5606E + 04	2.6089E + 04	5.3807E + 04	5.1466E + 04	5.3378E + 04
	$\sum_i P_i$	**3.6086E + 04**	3.5232E + 04	3.6213E + 04	**2.7505E + 04**	3.5285E + 04	4.2035E + 04
	$\sum_i L_i$	6.4934E + 04	6.7000E + 04	6.4101E + 04	**6.8840E + 04**	6.7857E + 04	7.2927E + 04
TS-1	$f(s)$	2.7014E + 04	2.6837E + 04	2.7969E + 04	5.2946E + 04	4.9618E + 04	5.0580E + 04
	$\sum_i P_i$	3.8294E + 04	**3.3972E + 04**	**3.2245E + 04**	3.7858E + 04	**3.5275E + 04**	4.5865E + 04
	$\sum_i L_i$	6.2538E + 04	6.3931E + 04	**5.9628E + 04**	7.1107E + 04	**6.7414E + 04**	7.4071E + 04
TS-2	$f(s)$	2.3491E + 04	2.2704E + 04	**2.4644E + 04**	5.3169E + 04	4.9553E + 04	5.0728E + 04
	$\sum_i P_i$	3.8406E + 04	3.6230E + 04	3.8184E + 04	3.3495E + 04	4.0840E + 04	**3.7685E + 04**
	$\sum_i L_i$	6.2881E + 04	6.2963E + 04	5.9881E + 04	6.9401E + 04	6.8505E + 04	**7.1460E + 04**
Alg.	Key	HT-1	HT-2	HT-3	M-1	M-2	M-3
VNSN	$f(s)$	**1.9085E + 04**	1.6386E + 04	1.9527E + 04	**1.3871E + 04**	**1.4114E + 04**	**1.3762E + 04**
	$\sum_i P_i$	3.5261E + 04	3.4095E + 04	3.8756E + 04	4.0933E + 04	3.9249E + 04	3.9124E + 04
	$\sum_i L_i$	4.1958E + 04	**3.3552E + 04**	4.2344E + 04	6.1266E + 04	**5.8337E + 04**	6.8238E + 04
VNSP-1	$f(s)$	2.1279E + 04	1.8352E + 04	2.1315E + 04	inf.	inf.	inf.
	$\sum_i P_i$	3.7372E + 04	3.6120E + 04	3.6786E + 04	inf.	inf.	inf.
	$\sum_i L_i$	4.3724E + 04	3.7317E + 04	4.5035E + 04	inf.	inf.	inf.
VNSP-2	$f(s)$	2.0480E + 04	1.7712E + 04	2.1340E + 04	1.5780E + 04	1.7444E + 04	1.4268E + 04
	$\sum_i P_i$	3.4611E + 04	**3.3338E + 04**	**3.6470E + 04**	**3.4046E + 04**	3.6527E + 04	**3.3678E + 04**
	$\sum_i L_i$	4.3755E + 04	3.6309E + 04	4.6864E + 04	6.4345E + 04	6.6202E + 04	7.1488E + 04
TS-1	$f(s)$	2.2224E + 04	1.8680E + 04	2.2079E + 04	inf.	inf.	inf.
	$\sum_i P_i$	3.3815E + 04	3.5405E + 04	3.7924E + 04	inf.	inf.	inf.
	$\sum_i L_i$	**4.0340E + 04**	3.4414E + 04	**4.1487E + 04**	inf.	inf.	inf.
TS-2	$f(s)$	1.9098E + 04	**1.6348E + 04**	**1.8926E + 04**	1.4903E + 04	1.4650E + 04	1.4861E + 04
	$\sum_i P_i$	**3.3460E + 04**	3.4881E + 04	3.7675E + 04	3.6860E + 04	**3.4614E + 04**	3.7471E + 04
	$\sum_i L_i$	4.2522E + 04	3.4473E + 04	4.3789E + 04	**5.9984E + 04**	6.3075E + 04	**6.6607E + 04**

inf. means an infeasible solution.

function. Therefore, the clients tend to spend more time during their ride times than waiting at the pickup stops.

Regarding the performance of each algorithm, it is clear that the proposed algorithm has obtained the best overall results, especially with the difficult 1000 requests problems where the best results were found in 12 out of 12 problems. As expected, the TS-1 algorithm (Cordeau & Laporte, 2003) has obtained the worst results of the comparison, mainly, due to the difference in the initialization function.

It is worth pointing out the surprising performance of the TS-2 algorithm in 100-D where, with merely the change of the initialization procedure, it has obtained the best mean results in 5 out 12 problems. With the original VNS the addition of the new initialization mechanism has not been so determinant in the results obtained but has allowed it to find feasible solutions in all the problems where the VNSP-1 algorithm was unable to do so.

In order to provide a proper statistical validation of the results, the distribution of all the results was first compared with the Friedman test to detect significant differences among the algorithms. The VNSP-1 and TS-1 algorithms were not included in this study since not all of their results were feasible. A value of 28.58 was obtained for the chi-squared statistic, which corresponds with a $p - value$ of $6.21E - 07$, confirming the existence of significant differences between the algorithms. According to this test, the algorithms were ranked as shown in Table 8, where, once again, the proposed algorithm obtained the best results. Then, 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 9, and show, for all of them, that there is statistical evidence to state that the proposed VNSN algorithm is significantly better than the remaining algorithms.

The previous analysis gave us an insight of the central tendency behavior of each algorithm. Since it is a common usage to conduct several executions and select the best routing solution among all the results, an interesting complementary study is to analyze the performance of the best results that each algorithm returned. Tables 10 and 11 display the best results. In these tables, the proposed algorithm has achieved similar better results, with 9 and 11 (out of 12 problems) best results in 100 and 1000 requests, respectively. Similarly, these results were validated following the previous procedure. The Friedman test returned a $p - value$ of $1.11E - 07$ and the associated ranks are displayed in Table 12. As shown in Table 13, the Wilcoxon tests also confirmed the significance of the results. Therefore, it is confirmed that the conducted modifications to the VNS algorithm have obtained not only the best results from a central point of view but also from an absolute point of view.

7.1. Improving the shakers scheme

Once it was proved the beneficial effects of the proposals, we tried to improve even more the VNS algorithm by looking for new ways for combining the shakers. As it can be seen in Table 3, there are some shakers, such as GWOM-4, that have a high number of improvements but that were not selected with the previous criterion due to their high number of evaluations consumed. On the other hand, there were some shakers that were being placed at

Table 11
Best values in 1000-D.

Alg.	Key	C-1	C-2	C-3	HS-1	HS-2	HS-3
VNSN	$f(s)$	**1.9590E + 05**	**1.9456E + 05**	**1.9744E + 05**	**4.2821E + 05**	**4.3324E + 05**	**4.2772E + 05**
	$\sum_i P_i$	**4.2853E + 05**	**4.2723E + 05**	**4.3569E + 05**	**4.4453E + 05**	**4.3919E + 05**	**4.3104E + 05**
	$\sum_i L_i$	**6.9961E + 05**	**6.6210E + 05**	**6.8203E + 05**	**8.2224E + 05**	**8.1139E + 05**	**8.0999E + 05**
VNSP-1	$c(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
VNSP-2	$f(s)$	2.4242E + 05	2.3305E + 05	2.4396E + 05	4.7996E + 05	4.8067E + 05	4.7901E + 05
	$\sum_i P_i$	4.3484E + 05	4.2750E + 05	4.4551E + 05	4.5194E + 05	4.5344E + 05	4.5527E + 05
	$\sum_i L_i$	7.3863E + 05	7.1159E + 05	7.2597E + 05	8.3409E + 05	8.2593E + 05	8.1391E + 05
TS-1	$f(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
TS-2	$f(s)$	2.4174E + 05	2.3248E + 05	2.4131E + 05	4.6567E + 05	4.7653E + 05	4.7401E + 05
	$\sum_i P_i$	4.5000E + 05	4.4574E + 05	4.5667E + 05	4.4677E + 05	4.4464E + 05	4.3470E + 05
	$\sum_i L_i$	7.6526E + 05	7.0802E + 05	7.3636E + 05	8.3019E + 05	8.1950E + 05	8.1312E + 05
Alg.	Key	HT-1	HT-2	HT-3	M-1	M-2	M-3
VNSN	$f(s)$	**1.6719E + 05**	**1.7014E + 05**	1.7437E + 05	**1.5153E + 05**	**1.5655E + 05**	**1.5935E + 05**
	$\sum_i P_i$	4.5342E + 05	**4.4073E + 05**	**4.4832E + 05**	4.1149E + 05	**4.0567E + 05**	**4.0534E + 05**
	$\sum_i L_i$	**4.7891E + 05**	**4.8018E + 05**	**4.7212E + 05**	**6.2439E + 05**	**6.2211E + 05**	**6.3563E + 05**
VNSP-1	$c(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
VNSP-2	$f(s)$	**1.6719E + 05**	1.7220E + 05	1.7527E + 05	1.6649E + 05	1.7068E + 05	1.7300E + 05
	$\sum_i P_i$	4.5424E + 05	4.4273E + 05	4.5265E + 05	4.1578E + 05	4.1799E + 05	4.1512E + 05
	$\sum_i L_i$	4.9588E + 05	4.9036E + 05	4.8255E + 05	6.9072E + 05	6.9023E + 05	6.9617E + 05
TS-1	$f(s)$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i P_i$	inf.	inf.	inf.	inf.	inf.	inf.
	$\sum_i L_i$	inf.	inf.	inf.	inf.	inf.	inf.
TS-2	$f(s)$	1.6784E + 05	1.7053E + 05	**1.7366E + 05**	1.6448E + 05	1.7240E + 05	1.7552E + 05
	$\sum_i P_i$	**4.5081E + 05**	4.5057E + 05	4.5006E + 05	**4.0748E + 05**	4.0962E + 05	4.2134E + 05
	$\sum_i L_i$	4.8720E + 05	4.9225E + 05	4.8497E + 05	6.8988E + 05	6.8017E + 05	6.9033E + 05

inf. means an infeasible solution.

Table 12
Average ranking of the best values.

	Ranking
VNSN	1.18
TS-2	2.00
VNSP-2	2.81

Table 13
Statistical validation for the bests values (VNSN is the control algorithm).

VNSN vs.	Wilcoxon p-value
VNSP-2	$1.19E-07^{\surd}$
TS-2	$5.38E-05^{\surd}$
Wilcoxon p-value with FWER: VNSN vs. TS-2, VNSP-2	$5.39E-05^{\surd}$

$\surd$ means that there are statistical differences with significance level $\alpha = 0.05$.

the higher (or better) positions in the table although the number of times that they improved a solution was considerably smaller than several of the shakers placed in inferior (or worse) positions because they had a small number of evaluations consumed. Since the shakers with a high number of improvements could be beneficial for finding better solutions, it was decided to take into account the number of improvements as part of the criterion for selecting the set of shakers. The new criterion, defined in Eq. (11), combines both the percentage normalization of the number of improvements per #evaluations consumed as well as the percentage normalization of the number of improvements.

$$newmeasure_i = \frac{1}{2} * \left(\frac{\#improvements_i/\#evals_i}{\sum_i \#improvements_i/\#evals_i} + \frac{\#improvements_i}{\sum_i \#improvements_i} \right) \quad (11)$$

The results of the performance of the shakers according to new criterion are shown in Table 14, where it can be seen that some shakers that were discarded with the previous criterion, such as GWOM-4, are now placed at the top positions of the table. Based on this criterion, two new shakers schemes were defined: (i) the subset of shakers that accumulats the 90% of the total value of the measure and sorted based on this value, i.e., C1-GWOM2-ANSC-C2-GBDM2-GWOM4-S1-GOS2-GWOM6-S2-C3-S2-GDS2 and (ii) a slightly reduced version where only shakers that accumulate the 80% of the total value have been selected, i.e, C1-GWOM2-ANSC-C2-GBDM2-GWOM4-S1-GOS2.

Based on the new schemes, two new VNS algorithms were created using the same set of parameter values of VNSN and naming them as VNSN-2 for the VNS using the first of the new schemes and VNSN-3 for the second. These algorithms were tested with the same set of problems and compared against the same algorithms of the previous Section. Tables 15 and 16 present the results of the new algorithms for 100 and 1000 requests. As with the previous tables, the best results among all the algorithms proposed in the paper are marked in bold.

From these results it can be seen that the new algorithms have obtained outstanding results in both 100 and 1000 dimensions having the VNSN-2 algorithm obtained the best results in 6 out of 12 problems in 100 dimensions and having VNSN-3 obtained the best results in all the problems in 1000 dimensions. It seems that the larger set of shakers of VNSN-2 has been more beneficial to explore new solutions with the 100 requests problems. On the

Table 14
Overall results of the performance of the shakers combining both the #improvements per evaluation as well as the number of improvements.

Shaker	#Improvements per feval	#Improvements	New Measure	Cum. Value
C1	2.4397E − 03	1.8680E + 03	0.48	0.244
GWOM2	3.6024E − 04	9.5620E + 03	0.35	0.420
ANSC	1.8033E − 04	5.0760E + 03	0.18	0.513
C2	6.5568E − 03	1.6650E + 03	0.16	0.596
GBDM2	2.8411E − 04	2.2410E + 03	0.11	0.655
GWOM4	6.5626E − 05	2.8260E + 03	0.09	0.704
S1	4.2473E − 04	5.9900E + 02	0.09	0.750
GOS2	1.8507E − 04	9.5600E + 02	0.06	0.781
GWOM6	1.8489E − 05	1.9200E + 03	0.06	0.812
S2	2.4236E − 04	4.9500E + 02	0.05	0.841
C3	1.1151E − 04	8.1000E + 02	0.04	0.863
S3	1.8265E − 04	3.1000E + 02	0.04	0.884
GDS2	5.7387E − 05	9.9800E + 02	0.04	0.904
S4	1.3222E − 04	2.7300E + 02	0.03	0.920
C4	6.3525E − 05	6.5500E + 02	0.03	0.935
S5	1.0345E − 04	2.1600E + 02	0.02	0.948
S6	9.7473E − 05	2.3200E + 02	0.02	0.960
GBDM4	6.1266E − 06	6.9600E + 02	0.02	0.971
GOS4	2.3021E − 05	3.1200E + 02	0.01	0.978
C5	7.2464E − 06	3.5800E + 02	0.01	0.984
GBDM6	2.8612E − 07	3.7700E + 02	0.01	0.990
C6	3.4861E − 06	2.1900E + 02	0.00	0.993
GOS6	2.8462E − 06	1.7600E + 02	0.00	0.996
GDS4	1.0497E − 06	1.8100E + 02	0.00	0.999
GDS6	4.5469E − 08	7.1000E + 01	0.00	1.000

Table 15
Mean values of the new algorithms in 100-D.

Alg.	Key	C-1	C-2	C-3	HS-1	HS-2	HS-3
VNSN-2	$f(s)$	**2.3769E + 04**	2.3394E + 04	**2.5402E + 04**	5.3312E + 04	5.0776E + 04	**5.1436E + 04**
	$\sum_i P_i$	4.4352E + 04	4.0996E + 04	4.5201E + 04	4.4459E + 04	4.9610E + 04	5.1673E + 04
	$\sum_i L_i$	6.6828E + 04	6.8226E + 04	6.6689E + 04	7.5112E + 04	7.3705E + 04	7.7196E + 04
VNSN-3	$f(s)$	2.3819E + 04	**2.3354E + 04**	2.5726E + 04	5.3194E + 04	5.1128E + 04	5.1444E + 04
	$\sum_i P_i$	4.3962E + 04	4.2510E + 04	4.3933E + 04	**4.2135E + 04**	5.0240E + 04	5.0482E + 04
	$\sum_i L_i$	6.6827E + 04	6.7895E + 04	6.6758E + 04	7.5101E + 04	7.4637E + 04	**7.7112E + 04**
Alg.	Key	HT-1	HT-2	HT-3	M-1	M-2	M-3
VNSN-2	$f(s)$	**1.9623E + 04**	1.7145E + 04	2.0513E + 04	**1.5164E + 04**	**1.5426E + 04**	1.5469E + 04
	$\sum_i P_i$	**4.0987E + 04**	4.0642E + 04	4.3904E + 04	4.4875E + 04	**4.3535E + 04**	4.5124E + 04
	$\sum_i L_i$	4.5907E + 04	3.7439E + 04	4.7089E + 04	6.7010E + 04	6.8194E + 04	**7.2362E + 04**
VNSN-3	$f(s)$	1.9895E + 04	1.7204E + 04	2.0615E + 04	1.5229E + 04	1.5715E + 04	1.5469E + 04
	$\sum_i P_i$	4.1436E + 04	4.0758E + 04	**4.2845E + 04**	4.4856E + 04	4.4105E + 04	4.4839E + 04
	$\sum_i L_i$	4.5758E + 0	3.7326E + 04	4.7644E + 04	6.6918E + 04	**6.7460E + 04**	7.3678E + 04

Table 16
Mean values of the new algorithms in 1000-D.

Alg.	Key	C-1	C-2	C-3	HS-1	HS-2	HS-3
VNSN-2	$f(s)$	1.9925E + 05	1.9396E + 05	1.9910E + 05	4.3797E + 05	4.4277E + 05	4.3922E + 05
	$\sum_i P_i$	4.5367E + 05	4.5354E + 05	4.5407E + 05	4.7449E + 05	4.6703E + 05	4.7581E + 05
	$\sum_i L_i$	7.2298E + 05	6.8358E + 05	7.1221E + 05	8.4207E + 05	8.3164E + 05	8.2916E + 05
VNSN-3	$f(s)$	**1.9585E + 05**	**1.9245E + 05**	**1.9690E + 05**	**4.3354E + 05**	**4.3811E + 05**	**4.3509E + 05**
	$\sum_i P_i$	4.5297E + 05	4.5508E + 05	**4.5387E + 05**	4.7936E + 05	4.6704E + 05	4.7399E + 05
	$\sum_i L_i$	**7.1943E + 05**	**6.8186E + 05**	**7.0413E + 05**	8.3986E + 05	8.3061E + 05	8.2887E + 05
Alg.	Key	HT-1	HT-2	HT-3	M-1	M-2	M-3
VNSN-2	$f(s)$	1.7157E + 05	1.7400E + 05	1.7728E + 05	1.5606E + 05	1.6129E + 05	1.6503E + 05
	$\sum_i P_i$	4.7214E + 05	**4.5944E + 05**	4.7165E + 05	**4.2508E + 05**	4.2299E + 05	**4.2757E + 05**
	$\sum_i L_i$	4.9619E + 05	**4.9153E + 05**	4.8389E + 05	6.5417E + 05	6.4982E + 05	6.6278E + 05
VNSN-3	$f(s)$	**1.7112E + 05**	**1.7338E + 05**	**1.7635E + 05**	**1.5428E + 05**	**1.5922E + 05**	**1.6248E + 05**
	$\sum_i P_i$	4.7178E + 05	4.6206E + 05	**4.6883E + 05**	4.2568E + 05	4.2544E + 05	4.2806E + 05
	$\sum_i L_i$	**4.9523E + 05**	4.9173E + 05	**4.8297E + 05**	**6.5103E + 05**	**6.4532E + 05**	**6.5037E + 05**

inf. means an infeasible solution.

Table 17
Average ranking of the mean values.

	Ranking
VNSN-2	1.83
VNSN-3	1.87
VNSN	2.79
TS-2	3.75
VNSP-2	4.75

Table 18
Statistical validation for the mean values (VNSN-2 is the control algorithm).

VNSN-2 vs.	Wilcoxon p-value
VNSN	$5.38E-05^{\surd}$
VNSP-2	$6.63E-10^{\surd}$
TS-2	$8.04E-05^{\surd}$
VNSN-3	$9.92E-01$
Wilcoxon p-value with FWER: VNSN-2 vs. VNSN,VNSP-2, TS-2	$5.54-05^{\surd}$
Wilcoxon p-value with FWER: VNSN-3 vs. VNSN,VNSP-2, TS-2	$3.78-04^{\surd}$

$\surd$ means that there are statistical differences with significance level $\alpha = 0.05$.

other hand, with 1000 requests, the subset of shakers of VNSN-3 has consumed more effectively the assigned evaluations and has been able to find better solutions in all the proposed problems.

To validate the results, the same statistical procedures of Section 7 were conducted with the data. The results are shown in Tables 17 and 18. In these tables, it can be seen that both algorithms have obtained the best results among all the algorithms that have been executed, being these differences statistically significant. VNSN-2 has obtained a better average ranking than VNSN-3 although the difference is so small that, as proved by the Wilcoxon test, there is no significant difference between the two algorithms. It seems that VNSN-2 works better with the low dimensional problems whereas VNSN-3 has obtained better results with the 1000 dimensions problems. Both approaches have demonstrated that the new criterion for constructing the shaker scheme has been quite successful in improving the results.

8. Conclusions

In this paper a new algorithm has been presented for the optimization of the service of a DRT transportation company. This algorithm, based on a VNS algorithm, has undergone several modifications in order to obtain the best results for the proposed problem. Concretely, the following proposals have been presented: a new initialization procedure centered around the objective function that tries to minimize the violation of the constraints, several neighborhood classes that use a greedy approach for conducting the perturbation of the solution and different schemes for combining the neighborhood classes that use new criteria not based on the size of the neighborhood (as has been traditionally used with VNS algorihtms). Moreover, the proposed algorithm has been generalized in several key parts to conduct a formal tuning of all of its parameters by means of the Taguchi method. The resultant 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.

Two points of view have been used for analyzing the results: one based in the central tendency of the executions and another one based on the best result of all the executions. The results have been compared against two state of the art algorithms of the literature and have been formally validated with the help of Friedman and Wilcoxon tests, proving, for both points of view, that the proposed algorithm has obtained the best results.

Acknowledgments

This work was financed by the Spanish Ministry of Science (TIN2010-21289-C02-02) and supported by the Cajal Blue Brain Project. Furthermore, the authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Supercomputing and Visualization Center of Madrid (CeSViMa) and the San Francisco Municipal Transportation Agency for the information regarding the transit stops used in this study. Finally, 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.

References

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 Journal on Computing, 19*, 618–632.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research, 54*, 573–586.

Cordeau, J. F., & Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological, 37*, 579–594.

Cordeau, J. -F., Laporte, G., Potvin, J. -Y., & Savelsbergh, M. (2007). Transportation on demand, *Handbooks in operations research and management science* (Vol. 14, pp. 429–466). Elsevier B. V.

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

D'Souza, C., Omkar, S., & Senthilnath, J. (2012). Pickup and delivery problem using metaheuristics techniques. *Expert Systems with Applications, 39*, 328–334.

Jorgensen, R. M., Larsen, J., & Bergvinsdottir, K. B. (2006). Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society, 58*, 1321–1331.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft, 58*, 81–117.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research, 37*, 1129–1138.

Parragh, S. N., Doerner, K. F., Hartl, R. F., & Gandibleux, X. (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks, 54*, 227–242.

Raghavendra, A., Krishnakumar, T., Muralidhar, R., Sarvanan, D., & Raghavendra, B. (1992). A practical heuristic for a large scale vehicle routing problem. *European Journal of Operational Research, 57*, 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*. Amsterdam: North-Holland.

Ropke, S., Cordeau, J.-F., & Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks, 49*, 258–272.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing, 4*, 146–154.

Taguchi, G., Chowdhury, S., & Wu, Y. (2005). *Taguchi's quality engineering handbook*. John Wiley.

Wilson, H. J. S., 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. *Journal of Software, 4*, 766–776.