A Hybrid Heuristic Algorithm for the Dial-a-Ride Problem

André Luyde S. Souza[1](✉), Jonatas B. C. Chagas[1,2], Puca H. V. Penna[1], and Marcone J. F. Souza[1]

[1] Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, Brazil
andreluyde@iceb.ufop.br, {puca,marcone}@ufop.edu.br

[2] Departamento de Informática, Universidade Federal de Viçosa, Viçosa, Brazil
jonatas.chagas@ufv.br

Abstract. In this paper, we propose a simple heuristic algorithm based on the Variable Neighborhood Search (VNS), which combines with the Set Covering strategy in order to solve the Dial-a-Ride Problem (DARP). In this problem, customers must be served by a heterogeneous fleet of vehicles. Each customer has a pickup and a delivery location, where each one of them has time windows that must be obeyed. All vehicles have a duration time and have to start and end their routes in a single depot, and each customer has a maximum time ride. We have tested our algorithm on the benchmark instances of literature. Experiments showed that although the algorithm is simple, it can obtain the optimal solutions for some instances and achieve solutions near the optima for the others.

Keywords: Dial-a-ride · Vehicle routing · Pickup and delivery · Variable Neighborhood Search · Randomized Variable Neighborhood Descent · Metaheuristic

1 Introduction

Lutz et al. [14] have reported the fact of the combinations of declining fertility, and increasing life expectancies resulted in the aging of the population. They also presented a likely increase in the speed of the aging population in the next decades and continuous aging of the world's population throughout the century. Due to this aging of the population, a lot of services need to be improved. Among them are transportation services. The problems involving the transportation can be classified as Vehicle Routing Problems (VRP). The VRP consists of determining a set of routes executed by a fleet of vehicles to satisfy the necessities of a set of users [22]. Many VRPs with different characteristics have been studied over the years, and one of them is the Dial-a-Ride Problem - DARP [4].

The DARP arises in the context of users transportation, a service for the people with reduced locomotion, or disabled people. Different from other VRPs,

Supported by CAPES.

R. Benmansour et al. (Eds.): ICVNS 2019, LNCS 12010, pp. 53–66, 2020.
https://doi.org/10.1007/978-3-030-44932-2_4

in the DARP, the customer's inconvenience has to take into account. The customer's nuisance can be defined as waiting time and travel time. The DARP consists of planning routes and schedules to meet a set of customers from a fleet of vehicles. Each customer has a pickup and a delivery location, where it must be collected and delivered, respectively. A maximum ride time has to be respected for each customer and duration time for each vehicle. A time window must be respected for every pickup and delivery point. The aim is to design a routing plan, i.e., a set of routes, capable of attending as many users as possible, under a set of constraints. Not different from other VRP variants, the DARP can be classified as static and dynamic. In the static case, all information about transportation request is known beforehand and used to calculate an a priori routing plan. In the dynamic case, the routing plan is designed in a real-time manner when a new request is revealed [6].

The DARP can be classified as the combination of two classical optimization problems: Vehicle Routing Problem with Pickup and Delivery (VRPPD) and Vehicle Routing Problem with Time Windows (VRPTW). The DARP differs from them as far as the human perspective is concerned, in DARP the comfort and convenience of users should be taken into account [6].

Besides, DARP is classified as an $\mathcal{NP}$-Hard problem due to its high complexity. Thus, optimal solutions cannot be obtained in a reasonable computational time. Therefore, DARP has been mainly addressed by heuristic and metaheuristic algorithms [6,10].

In this paper, we propose a VNS-based algorithm to solve the DARP. Our algorithm was combined with a Set Covering Problem (SCP) formulation, which is applied throughout the VNS algorithm in order to improve its current solution. This approach has been able to find high-quality solutions in short computational time.

The remainder of this paper is organized as follows. In Sect. 2, we present a brief literary review, contextualizing and exemplifying the DARP and its variants, as well as different ways for solving them. The formal definition is shown in Sect. 3. The proposed algorithm is detailed in Sect. 4. Section 5 presents the computational experiments and results. Conclusions are given in Sect. 6.

2 Literature Review

Several DARP variants have been studied in the past few years. We can classify them into two main branches: static and dynamic versions.

In the static version, all the information about the customer requests and vehicles is known beforehand. Cordeau and Laporte [4] considered the problem with a homogeneous fleet of vehicles and single depot. To solve it, they used a Tabu Search (TS) algorithm combined with three heuristic methods. Cordeau [5] approached the DARP by an exact branch-and-cut algorithm, which was tested on a set of instances randomly generated with a maximum of 48 requests. Jorgensen et al. [12] addressed the variant of DARP with a Genetic Algorithm (GA), which considers a heterogeneous fleet of vehicles and multiple depots they made the algorithm basing on the classical cluster-first and route-second approach. Mauri et al. [17], using the data provided by the city of Vitória–ES in Brazil, approached a static variant with a heterogeneous fleet of vehicles and multiple depots. The authors used a Simulated Annealing (SA) algorithm to solve the problem. Their results show that SA produced good quality solutions for all the instances.

Parragh [19] combined other constraints to the classic DARP and proposed a branch-and-cut algorithm to solve the problem. She added real characteristics to the problem as heterogeneity to the vehicles and users. In her tests, for the instances up to 40 requests, she found the optimal solution. Again, Parragh et al. [20] proposed a hybrid column generation and large neighborhood search algorithm to solve the problem. They improved nine of the 20 instances used in their tests in 1.1%. Currently, the best-known solutions of eight of these instances are the solutions founded by Parraagh et al. [20].

Braekers et al. [3] approached the DARP using the branch-and-cut algorithm combined with a metaheuristic named determinist annealing based in the simulated annealing metaheuristic. They formulated the problem as a single depot and multi-depot for heterogeneous DARP. They tested their approach in benchmarks created by Cordeau [5], Parragh 2011 [19], and Cordeau et al. [4]. They also extended the sets of instances creating 36 larger new instances in the same way of Parragh [19]. They match or exceed most of the best-known results found for these instances. Most of the solutions founded by the authors remain as the best-known solution for the instances. Masmoudi et al. [16] solved the same heterogeneous version of the DARP using a hybrid Genetic Algorithm. They used a constructive heuristic and efficient crossovers to guided the algorithm. They tested their approach in the instances of [5,19], and [3]. The results demonstrate the algorithm is efficient in terms of best and average solution quality.

In the dynamic version of DARP, new customer requests are added during the route. Madsen et al. [15] solve the DARP adapting an insertion algorithm called REBUS originally developed by Jaw et al. [11]. The version of the problem solved used the multiple capacities and multiple objectives. Using data given by Copenhagen Fire-Fighting Service, they tested their approach on real-life cases. In a flexible way, the algorithm permits a weighing of multiple goals such that the solution reflects the customer's preferences. Attanasio et al. [1] approached the dynamic DARP with the objective to accept as many requests as possible while satisfying the constraints of the problem. To solve the problem, they used a parallel strategy of a Tabu Search heuristic previously used by Cordeau and Laporte [4] in the static case of the problem.

In Germany, Beaudry et al. [2] proposed a two-phase heuristic to solve a dynamic DARP. The first phase is a simple insertion scheme, and the second phase is a Tabu Search algorithm that considers infeasible solutions during the search process. The problem was structured with data given by several large hospitals. They proposed additional constraints on the standard problem. For example, a different degree of urgency in the requests, the patient of the request cannot share the vehicle with another patient, among others. Experiments provided high-quality solutions, and the algorithm show capable of handling the dynamic aspect of the problem. Schilde et al. [21] proposed a dynamic DARP to solve the problem of the organization performing patient transportation in Austria. They approached the dynamic case of the DARP with a homogeneous fleet of vehicles and a single depot. To solve the problem, the authors proposed four different metaheuristics, a Variable Neighborhood Search (VNS), a Stochastic Variable Neighborhood Search (S-VNS), both dynamics, a Multiple Plan Approach (MPA), and a Multiple Scenario Approach (MSA). They tested their algorithms on a set of 12 instances based on a real road network. Results show that dynamic S-VNS strongly outperforms the others. Again in Germany, Hanne et al. [8] solve a dynamic DARP with hospital-specific constraints. They designed a computer-based planning system, Opti-TRANS, to support several concerns related to patient transport. They illustrated the system work in the daily performance of a large German hospital, and the system presents many benefits, including streamlined transportation processes and workflow.

Thus, the large number of DARP variants, here, we study the Heterogeneous DARP with single depot proposed by Parragh et al [20]. The heterogeneity is considered in the user and vehicles [20].

3 Formal Definition

The DARP can be defined as follows. Let $G = (V, E)$ be a complete graph, where V is the set of vertices and E the set of arcs. The set of vertice V is partitioned into $\{\{0, 2n + 1\}, P, D\}$, where 0 and $2n + 1$ are two copies of the depot, $P = \{1, ..., n\}$ is the set of pickup vertices and $D = \{n + 1, ..., 2n\}$ is the set of delivery vertices. For each arc $(i, j) \in E$ is associated a routing cost c_{ij} and a travel time t_{ij}.

Each customer request i consists of a pickup and delivery vertex pair $\{i, n+i\}$, where $i \in P$ and $n + i \in D$. The maximum travel time of each customer cannot exceed L. To each vertex $i \in V$ is associated a load q_i, with $q_0 = q_{2n+1} = 0$, $q_i \geq 0 \ \forall i \in P$ and $q_i = -q_{i+n} \ \forall i \in D$, and a non-negative service time τ_i. Moreover, each $i \in V$ has a time window $[e_i, l_i]$, where e_i and l_i are integers non-negative.

The set of vehicles is represented by K. Each vehicle $k \in K$ starts and ends its route in the depot. The capacity of vehicle k is Q_k and the maximal duration of route k is denoted by T_k.

Given the vertex v_i we denoted by A_i the arrival time of the vehicle in the vertex; and B_i the beginning of the service, where $B_i \geq \max\{A_i, e_i\}$, cannot start before e_i. The departure time $D_i = B_i + \tau_i$ is the time the vehicle leaves the vertex. The vehicle waiting time is defined by $W_i = B_i - A_i$. The ride time of the client is determined by $L_i = B_{n+i} - D_i$ and the total duration of the route is calculated as $B^k_{n+1} - B^k_0$, where B^k_{n+1} and B^k_0 represents the beginning of service on the depot by vehicle k, when it finishes and starts the ride, respectively.

The objective of the DARP is to find a set of routes that serve all customers such that minimizes the total routing cost.

4 Heuristic Approach

To solve the problem we based our heuristic on a VNS and a set covering procedure. Sections 4.1–4.4 define the heuristic components, while Sect. 4.5 presents the set covering model.

4.1 Solution Representation

We represent a DARP solution through a matrix of $|K|$ rows, where each row informs the route assigned to a vehicle (the k-th row refers to the route of the k-th vehicle). Each route begins and ends at the depot. The pickup and delivery customers are inserted in the route, satisfying the problem constraints.

Figure 1 shows a solution representation for an instance with five vehicles $(A, B, \cdots, E)$ and 13 customer requests. The numbers with positive sign $(1+ \quad \cdots \ 13+)$ represent the pickup locations and the ones with negative sign $(1- \quad \cdots \ 13-)$ are the delivery points. The depot is represented by node 0.

A	0	9+	4+	9-	2+	2-	4-	6+	6-	0
B	0	3+	5+	3-	1+	1-	5-	0		
C	0	7+	8+	7-	8-	0				
D	0	10+	12+	13+	10-	12-	13-	0		
E	0	11+	11-	0						

Fig. 1. Example of a solution representation.

4.2 Solution Evaluation

The solution evaluation was done based on an approach used by Cordeau and Laporte [4] and Parragh et al. [18]. Following them, we penalized the violations of load $q(s)$, duration $d(s)$, time windows tw, and user ride time $t(s)$. The load and duration are computed and penalized in the route, based on the constraints Q_k and T_k. The time windows penalization is computed as

$$tw = \sum_{i=0}^{2n} (B_i - l_i)^+$$

where $x^+ = \max\{x, 0\}$. In turn, the ride time is calculated as

$$t = \sum_{i=1}^{n} (L_i - L)^+$$

Thus, the solution evaluation was calculated as follows:

$$f(s) = c(s) + \alpha tw(s) + \beta t(s) + \delta d(s) + \gamma q(s)$$

The penalization variables for load (γ), duration (δ), time window (α), and ride time (β) violations were set to $\alpha = \beta = \delta = \gamma = 1$.

4.3 Neighborhood Structures

To explore the solution space of the DARP, six neighborhoods was implemented. These neighborhoods consist in the relocation of requests among the solution routes. For all neighborhoods, two routes, r_1 (blue line) and r_2 (black line) are randomly selected.

– Relocation – A request (pickup and delivery points) are removed from r_1 and inserted in r_2. Figure 2 shows the relocation of request $(1^+, 1^-)$ from r_1 to r_2.

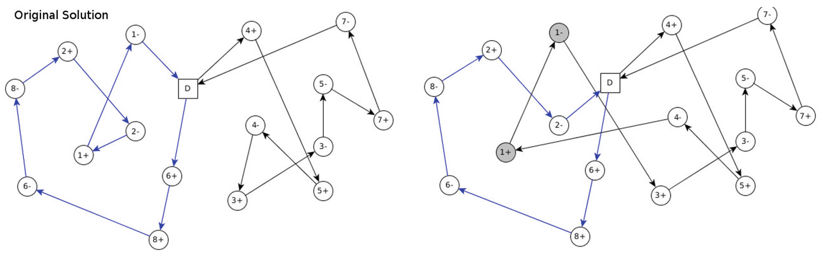

Fig. 2. Relocation. (Color figure online)

– Swap – Two requests are selected, one from r_1 and another and r_2, and exchanged them. Figure 3 shows the exchange of one request $(6^+, 6^-)$ in route r_1 with other request $(4^+, 4^-)$ in route r_2.

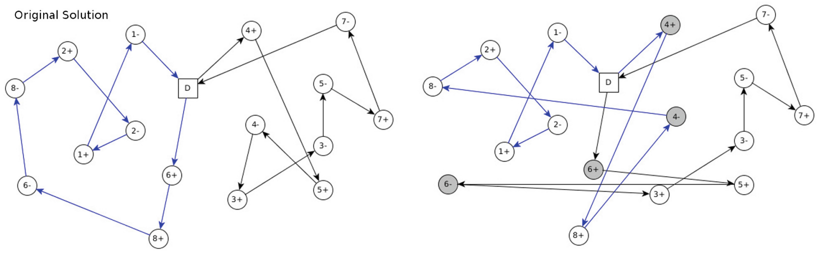

Fig. 3. Swap. (Color figure online)

– Crossover – Two points are selected, one in r_1 and other in r_2. All pickup customers before the selected point in r_1 (and their respective deliveries) are connected to all pickup customers (and their respective deliveries) that come after the selected point in r_2. The same way to the customers before r_2 point and after the r_1 point. Figure 4 shows the crossover of routes, the red line shows the selected points for each route and the relocation of requests.

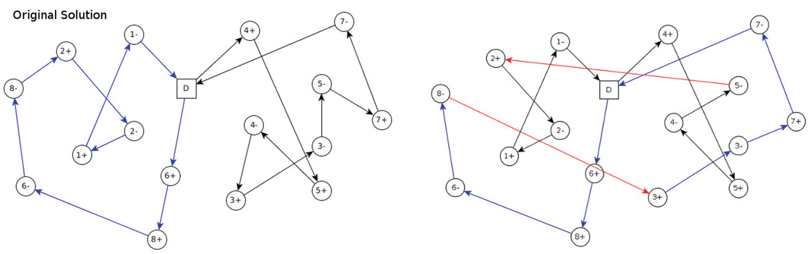

Fig. 4. Crossover. (Color figure online)

– Swap(2) – Four requests are selected, two adjacent requests from r_1 and two adjacent requests in r_2, and they are exchanged. All the four possible combination orders of exchanging the request arcs are considered. Figure 5 shows the exchange of two requests $(6^+, 6^-)$ and $(8^+, 8^-)$ in route r_1 with two requests $(4^+, 4^-)$ and $(3^+, 3^-)$ in route r_2.

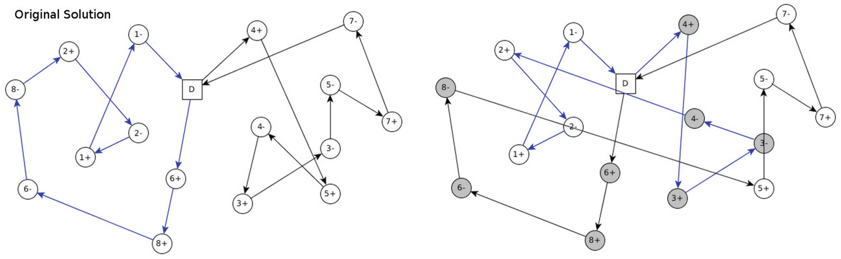

Fig. 5. Swap2. (Color figure online)

– Swap(2-1) – Three requests are selected, two adjacent requests from r_1 and one in r_2, and they are exchanged. The move examines the two possible visiting orders of transferring the r_1 requests. Figure 6 illustrates the exchange of two requests $(6^+, 6^-)$ and $(8^+, 8^-)$ in route r_1 with the request $(4^+, 4^-)$ in route r_2.

– Relocation(2) – Two adjacent requests are removed from r_1 and inserted in r_2. Figure 7 shows the relocation of requests $(6^+, 6^-)$ and $(8^+, 8^-)$ from r_1 to r_2.

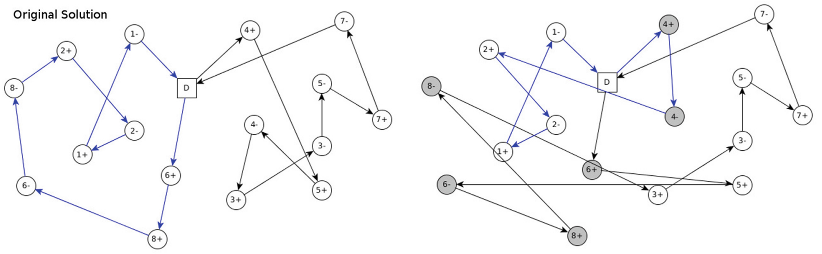

Fig. 6. Swap2. (Color figure online)

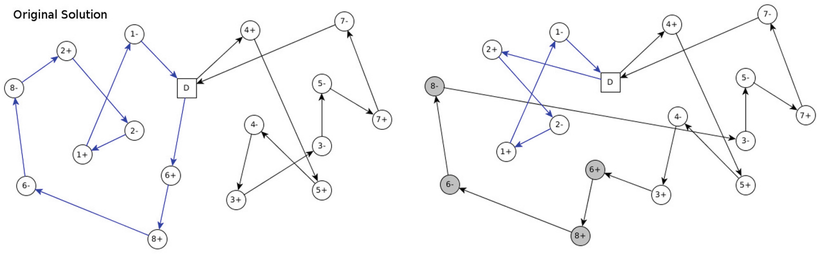

Fig. 7. Relocation. (Color figure online)

4.4 Variable Neighborhood Search

To tackle the DARP, a VNS-based algorithm [9] was proposed. For its use, three components have to be specified:

- Generate initial solution Procedure: A function that greedily selects each request taking into account the time window allocating to each chosen car with the shortest time traveled.
- Local Search Procedure: The method used as local search was the Randomized Variable Neighborhood Descent (RVND), that has its behavior like a classic Variable Neighborhood Descent, but the neighborhoods are randomly selected.
- Shaking Procedure: Let Ω be a set of neighborhood structures. The shaking procedure consists of selecting a random neighborhood belonging to Ω and then choosing a random neighbor of the current solution in this neighborhood.

All steps of our VNS algorithm applied to solve the DARP are presented in Algorithm 1. The algorithm receives as parameters an initial solution s_0 and the maximum number of iterations ($iterMax$). In the line 4 is applied a shaking in the current solution. The local search is applied in line 5, where all different feasible routes are stored in a set R. This set is used in the set covering model. When the number of iterations exceeds half of $iterMax$, the set covering procedure is applied (line 13).

Algorithm 1. VNS procedure

```
1: procedure VNS ($s_0$, $iterMax$)
2:     $iter \leftarrow 0$
3:     while $iter \leq iterMax$ do
4:         $s' \leftarrow$ Shaking($s$)
5:         $s'' \leftarrow$ LocalSearch($R, s'$)
6:         if $f(s'') < f(s)$ then
7:             $s \leftarrow s''$
8:             $iter \leftarrow 0$
9:         else
10:            $iter \leftarrow iter + 1$
11:        end if
12:        if $iter \geq iterMax/2$ then
13:            $s'' \leftarrow$ setCovering(R)
14:            if $f(s'') < f(s)$ then
15:                $s \leftarrow s''$
16:                $iter \leftarrow 0$
17:            end if
18:        end if
19:    end while
20:    return $s$
21: end procedure
```

4.5 Set Covering

Let R be a set of different feasible routes found by VNS algorithm, V the set of customers and $|K|$ the maximum number of vehicles available. A cover of V is a combination J of routes $j \in R$, where each customer $i \in V$ is covered by at least one route $j \in R$. Let c_j be the cost of each route j and the binary constant ρ_{ij} that informs if customer i is served by route j. The Set Covering problem (SCP) consists in finding a set $J \subseteq R$ such that the total cost is minimized.

In order to present the mathematical model for the SCP, let y_j be a binary variable associated with a route $j \in J$. Follow the model used in our algorithm.

$$\min \sum_{j \in R} c_j y_j$$

Subject to:

$$\sum_{j \in R} \rho_{ij} y_j \geq 1, \qquad \forall i \in V \tag{1}$$

$$\sum_{j \in R} y_j \leq |K|, \tag{2}$$

$$y_j \in \{0, 1\} \tag{3}$$

5 Computational Experiments

The developed heuristic was coded in C/C++ using the mathematical solver Gurobi 8.0.0, with all its default settings. All experiments were performed on an Intel Core i7-7700K processor with 3.60 GHz and 16 GB RAM running Ubuntu 16.04 LTS 64 bits.

5.1 Instances Description

In order to validate our approaches, the computational experiments were performed using three groups of instances (E, I, U) proposed by Parragh [19]. These instances have heterogeneity based on data given by the Austrian Red Cross. The instances were randomly generated in the square $[-10, 10]^2$ using the uniform distribution. The depot was located in the center of the plan. Each edge $(v_i, v_j) \in A$ has the cost c_{ij} and travel time t_{ij} were calculated using Euclidean distance. Each vertex i was defined a time window $[e_i, l_i]$. The pickup vertices e_i were established in a range of $[0, T-60]$, where T is the time of planning horizon and l_i was set as $e_i + 15$. The delivery vertices l_i were defined in the interval $[60, T]$ and e_i was set as $l_i - 15$. Each instance is represented by the name *ak-n*, where k is the number of vehicles used and n is the total number of requests.

In group E of instances, half of the patients are normal ones, 25% are wheelchair ones and 25% stretcher ones. 10% of the patients have one companion. The fleet of vehicles for this group is homogeneous with vehicles of type V1. Each vehicle V1 has two places to companions, one for a wheelchair patient, one for the stretcher patient and one for a normal patient. In group I of instances, 83% of requests are from normal patients, 11% are wheelchair ones, and 6% are stretcher ones. Half of the patients have one companion. The fleet of vehicles is heterogeneous with vehicles of types V1 and V2. Each vehicle V2 has one place to companions, one for a wheelchair patient, one for stretcher patient and six for the normal patients. In group U of instances, are just considered the normal patients and there are no companions. The fleet of vehicles is homogeneous with vehicles of type V3. Each vehicle V3 just has places for normal patients. Table 2 shows the heterogeneity of patients and vehicles by instance groups (Table 1).

Table 1. Patients occupation

Patient type	Place type:			
	Normal	Wheelchair	Stretcher	Companion
Normal	x	x		
Wheelchair		x		
Stretcher			x	
Companion	x	x		x

5.2 Experimental Results

For each instance, the algorithm was executed 10 times using as stop criteria $iterMax = 100$, which represents the maximum number of the iterations without improving the

Table 2. Instances information

Instance group	Probability of patient be:				Fleet of vehicles
	Normal	Wheelchair	Stretcher	Companion	
E	0.50	0.25	0.25	0.10	hom(V1)
I	0.83	0.11	0.06	0.50	het(V1, V2)
U	1.00	0.00	0.00	0.00	hom(V3)

hom = Homogeneous, het = Heterogeneous
V1: 2 Companions, 1 Normal, 1 Wheelchair, 1 Stretcher
V2: 1 Companion, 6 Normal, 1 Wheelchair, 1 Stretcher
V3: 0 Companion, 3 Normal, 0 Wheelchair, 0 Stretcher

best solution found. For running the set covering problem was used half of the $iterMax$ iterations. The parameters used in the algorithm were calibrated using the IRACE package [13]. Table 3 shows the tested configurations for the parameters. The best configurations returned by IRACE are boldfaced. The neighborhoods are represented as follow and the selected neighborhoods are presented in Sect. 4.3:

1 - Relocation - Relocation of one request of one vehicle to another.
2 - Swap - Exchange of one request of one vehicle to another.
3 - Crossover - Crossover between two vehicles.
4 - Swap(2) - Exchange of two requests of one vehicle to another.
5 - Swap(2-1) - Exchange of two requests of one vehicle with one request from another.
6 - Relocation(2) - Relocation of two requests of one vehicle to another.

Table 3. IRACE Calibration

Parameter	Values
Local search	123, 124, 125, 126, 134, 135, 136, 145, 146, 156
Neighborhood	**234**, 235, 236, 245, 246, 256, 345, 346, 356, 456
Initial penalty	1, 3, 5, 8, **10**
iterMax	50, **100**, 150

Table 4 presents the results on the instances group U, E and I. In group U, the customers and vehicle fleet were considered as homogeneous. In group E the customers were considered heterogeneous and the vehicle fleet was homogeneous. In the group I the customers and vehicle fleet are heterogeneous. In this table, the column **Instance** represents the name for each instance, this name is composed by the number of vehicles used and the number of requests. The columns **Parragh** [19] and **Braekers et al.** [3] show the results obtained by these authors in their experiments. The column **Best VNS** presents the best solution found by our algorithm and the column **AVG VNS** shows the average values in 10 executions of the VNS algorithm. For each column of the table, **sol** reports the objective function value, **gap** the difference between the solution value found and the best-known solution (BKS) value, while the column **time** shows the time spend by the algorithm in seconds. The times in this work are from different

Table 4. Results for instances U, E and I

Instances	Parragh [19]			Braekers et al. [3]			Best VNS		AVG VNS		
	Sol.	Gap%	Time	Sol.	Gap%	Time	Sol.	Gap%	Sol.	Gap%	Time
Group U											
a2−16	**294.25**	0.00	68.20	**294.25**	0.00	8.60	**294.25**	0.00	**294.25**	0.00	3.83
a2−20	**344.83**	0.00	133.80	**344.83**	0.00	20.20	**344.83**	0.00	**344.83**	0.00	15.69
a2−24	**431.12**	0.00	187.80	**431.12**	0.00	17.40	434.53	0.78	436.48	1.23	42.26
a3−18	**300.48**	0.00	45.40	**300.48**	0.00	9.40	**300.48**	0.00	302.53	0.68	2.60
a3−24	**344.83**	0.00	86.80	**344.83**	0.00	16.60	344.91	0.02	347.50	0.77	14.58
a3−30	**494.85**	0.00	105.60	**494.85**	0.00	18.80	500.51	1.13	502.52	1.53	74.02
a3−36	583.30	0.02	162.60	**583.19**	0.00	28.40	599.43	2.71	607.24	3.96	247.54
a4−16	**282.68**	0.00	26.00	**282.68**	0.00	9.80	283.10	0.15	283.10	0.15	1.04
a4−24	**375.02**	0.00	50.80	**375.02**	0.00	13.00	379.36	1.14	380.81	1.52	6.36
a4−32	486.88	0.28	86.00	**485.50**	0.00	25.60	487.31	0.37	491.70	1.26	54.24
a4−40	561.80	0.74	130.60	**557.69**	0.00	26.40	**557.69**	0.00	569.36	2.05	243.66
a4−48	673.64	0.72	253.80	**668.82**	0.00	35.40	678.98	1.50	683.13	2.09	580.92
Group E											
a2−16	**331.16**	0.00	65.60	**331.16**	0.00	9.40	**331.16**	0.00	**331.16**	0.00	4.89
a2−20	**347.03**	0.00	120.00	**347.03**	0.00	19.60	**347.03**	0.00	**347.03**	0.00	16.32
a2−24	**450.25**	0.00	160.40	**450.25**	0.00	15.80	450.38	0.02	452.92	0.65	76.08
a3−18	**300.63**	0.00	47.60	**300.63**	0.00	9.60	**300.63**	0.00	302.01	0.46	2.05
a3−24	**344.91**	0.00	76.20	**344.91**	0.00	14.60	**344.91**	0.00	347.63	0.86	17.73
a3−30	**500.58**	0.00	107.60	**500.58**	0.00	17.00	505.64	1.00	507.72	1.54	108.75
a3−36	**583.19**	0.00	161.60	**583.19**	0.00	23.60	599.43	2.71	610.46	4.47	316.76
a4−16	**285.99**	0.00	25.00	**285.99**	0.00	8.20	291.55	1.91	291.76	1.98	1.25
a4−24	**383.84**	0.00	52.60	**383.84**	0.00	12.20	386.06	0.58	289.09	1.35	8.60
a4−32	502.52	0.45	83.00	**500.24**	0.00	22.80	501.85	0.32	507.11	1.35	49.16
a4−40	585.64	0.90	121.00	**580.42**	0.00	24.20	589.29	1.51	597.89	2.92	204.60
a4−48	675.37	0.72	252.20	**670.52**	0.00	33.60	676.28	0.85	688.57	2.62	786.57
Group I											
a2−16	**294.25**	0.00	68.40	**294.25**	0.00	7.20	**294.25**	0.00	**294.25**	0.00	5.33
a2−20	**355.74**	0.00	141.80	**355.74**	0.00	17.40	360.23	1.25	362.69	1.92	14.16
a2−24	**431.12**	0.00	211.00	**431.12**	0.00	12.60	434.53	0.78	441.32	2.31	59.08
a3−18	**302.17**	0.00	47.20	**302.17**	0.00	8.40	**302.17**	0.00	303.27	0.36	3.53
a3−24	**344.83**	0.00	83.60	**344.83**	0.00	13.40	345.31	0.14	350.79	1.70	19.23
a3−30	**494.85**	0.00	106.80	**494.85**	0.00	14.80	502.77	1.57	511.75	3.30	93.29
a3−36	618.58	0.07	170.60	**618.15**	0.00	22.60	636.97	2.95	641.98	3.71	296.38
a4−16	**299.05**	0.00	27.00	**299.05**	0.00	7.20	302.87	1.26	307.23	2.66	1.09
a4−24	375.07	0.01	51.60	**375.02**	0.00	12.00	379.36	1.14	381.91	1.80	9.83
a4−32	**486.93**	0.00	88.00	**486.93**	0.00	21.00	488.74	0.37	496.25	1.88	59.96
a4−40	561.35	0.66	132.20	**557.69**	0.00	23.80	566.11	1.49	573.02	2.68	279.11
a4−48	680.43	1.45	262.40	**670.72**	0.00	30.00	684.37	1.99	695.95	3.63	749.48

computers, the computer used by Parragh [19] was an Intel Pentium D with 3.20 GHz and 4 GB RAM and the computer used by Braekers et al. [3] was an Intel Core with 2.6 GHz and 4 GB RAM, and the computer used in this work is an Intel Core I7-7700 CPU @ 3.60 GHz with 16 GB RAM. Some methods have been developed for a fair comparison, methods that measure the performance of computers in general, and can be found in work of Dongarra [7].

According to Table 4, we can see that our algorithm was able to find the BKS solution for 10 of 36 instances, where they are four in group U and E and two in group I. The VNS found solutions with gap up to 1% for 12 of 36 cases, where they are four in group U, five in group E and three in group I. In the last 14 cases our algorithm obtained solutions with gap up to 2.95%.

Regarding computation time, our algorithm on average is faster in 26 out of 32 instances compared to Parragh [19] and find the better solution just for one case (a4−40 of group U). Compared to Braekers et al. [3], the VNS algorithm on average is faster just for half of 32 instances.

6 Conclusions

In this paper, we propose a simple heuristic algorithm based on the Variable Neighborhood Search for the Dial-a-Ride Problem. This problem is a variation of the Vehicle Routing Problem, where the customer's convenience is taken into account. We considered a heterogeneous demand for customers and vehicles. Our algorithm was tested in the instances described in the literature. In three groups with a total of 36 instances. The VNS algorithm showed able to find 10 of 36 best-known solutions, and for 12 of 36 solutions, we find a solution with a gap less than 1%. For the other 14 instances, in the worst case, we find a solution with a gap of 2.95%.

Acknowledgments. The authors thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Universidade Federal de Ouro Preto (UFOP) and Universidade Federal de Viçosa (UFV) for supporting this research.

References

1. Attanasio, A., Cordeau, J.F., Ghiani, G., Laporte, G.: Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Comput. **30**, 231–236 (2004)
2. Beaudry, A., Laporte, G., Melo, T., Nickel, S.: Dynamic transportation of patients in hospitals. OR Spectr. **32**, 77–107 (2010). https://doi.org/10.1007/s00291-008-0135-6
3. Braekers, K., Caris, A., Janssens, G.K.: Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transp. Res. Part B Methodol. **67**, 166–186 (2014)
4. Cordeau, J.F., Laporte, G.: A Tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B Methodol. **37**, 579–594 (2003)
5. Cordeau, J.F.: A branch-and-cut algorithm for the dial-a-ride problem. Oper. Res. **54**, 573–586 (2006)

6. Cordeau, J.F., Laporte, G.: The dial-a-ride problem: models and algorithms. Ann. Oper. Res. **153**, 29–46 (2007). https://doi.org/10.1007/s10479-007-0170-8
7. Dongarra, J.J.: Performance of various computers using standard linear equations software. ACM SIGARCH Comput. Arch. News **20**, 22–44 (2014)
8. Hanne, T., Melo, T., Nickel, S., Melo, T., Nickel, S.: Bringing robustness to patient flow management through optimized patient transportation in hospitals. Interfaces **39**, 241–255 (2018)
9. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**, 449467 (2001)
10. Ho, S.C., Szeto, W.Y., Kuo, Y.H., Leung, J.M.Y., Petering, M., Tou, T.W.H.: A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. Part B Methodol. **111**, 395–421 (2018)
11. Jaw, J.-J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.M.: A heuristic algorithm for the multi- vehicle advance request dial-a-ride problem with time windows. Transp. Res. **20B**, 243–257 (1986)
12. Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B.: Solving the dial-a-ride problem using genetic algorithms. J. Oper. Res. Soc. **58**, 1321–1331 (2007)
13. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016)
14. Lutz, W., Sanderson, W., Scherbov, S.: The coming acceleration of global population ageing. Nature **451**, 716–719 (2008)
15. Madsen, O.B.G., Ravn, H.F., Rygaard, J.M.: A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Ann. Oper. Res. **60**, 193–208 (1995). https://doi.org/10.1007/BF02031946
16. Masmoudi, M.A., Braekers, K., Masmoudi, M., Dammak, A.: A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. Comput. Oper. Res. **81**, 1–13 (2017)
17. Mauri, G.R., Antonio, L., Lorena, N.: Customers' satisfaction in a dial-a-ride problem. IEEE Intell. Transp. Syst. Mag. **3**, 6–14 (2009)
18. Parragh, S.N., Doerner, K.F., Hartl, R.F.: Variable neighborhood search for the dial-a-ride problem. Comput. Oper. Res. **37**, 1129–1138 (2010)
19. Parragh, S.N.: Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transp. Res. Part C Emerg. Technol. **19**, 912–930 (2011)
20. Parragh, S.N., Schmid, V.: Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. **40**, 490–497 (2013)
21. Schilde, M., Doerner, K.F., Hartl, R.F.: Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. Comput. Oper. Res. **12**, 1719–1730 (2011)
22. Toth, P., Vigo, D.: The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematic and Applications, Philadelphia (2002)