CIRRELT
1971 – 2021

CIRRELT-2021-47

The Dial-a-Ride Problem with Private Fleet and Common Carrier

Cleder M. Schenekemberg
Antonio A. Chaves
Leandro C. Coelho
Thiago A. Guimarães
Gustavo G. Avelino

December 2021

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2021-014.

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

The Dial-a-Ride Problem with Private Fleet and Common Carrier

Cleder M. Schenekemberg[1], Antonio A. Chaves[1], Leandro C. Coelho[2,*], Thiago A. Guimarães[3], Gustavo G. Avelino[1]

1. Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil {cledercms@hotmail.com} {antonio.chaves@unifesp.br} {gustavo_godeny@hotmail.com}
2. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, Québec, Canada {leandro.coelho@cirrelt.ca}
3. Research Group of Technology Applied to Optimization (GTAO), Federal University of Paraná (UFPR), Curitiba, Brazil. Federal Institute of Science and Technology of Paraná (IFPR), Pinhais, Brazil {thiagoandre@ufpr.br}

Abstract. Dial-a-ride problems aim to design a least-cost door-to-door vehicle routes for transporting individual persons, subject to several service constraints like time windows, service and route durations, and ride-time. In some cases, providers cannot meet the demand and may outsource some requests. In this paper, we introduce, model, and solve the dial-a-ride problem with private fleet and common carrier (DARP-PFCC) that makes it possible to transfer the demand unmet by the provider to mobility-on-demand services and taxis. We design a branch-and-cut (B&C) algorithm to solve the DARP-PFCC exactly, and we develop a near parameter-free parallel metaheuristic to handle large instances. Our metaheuristic combines the Biased Random-key Genetic Algorithm (BRKGA) and the Q-learning (QL) method into the same framework (BRKGA-QL). Both algorithms are flexible enough to solve the classical DARP, and extensive computational experiments demonstrate the efficiency of our methods. For the DARP instances, our B&C proved optimality for 41 of the 42 instances tested in a reasonable computational time, and the BRKGA-QL found the best-known solution for these instances within a matter of seconds. In the DARP-PFCC experiments, both B&C and BRKGA-QL proved to be very efficient. We also derive some managerial analyses to assess the effects of the COVID-19 pandemic on shared transportation. Finally, we present the results for a real case study, where BRKGA-QL solves very large problem instances. The results point to the benefits of combining the private fleet and common carriers in dial-a-ride problems, both for the provider and for the users.

Keywords: dial-a-ride, private fleet, common carrier, branch-and-cut, BRKGA, machine learning.

Acknowledgements. Cleder M. Schenekemberg was supported by the São Paulo Research Foundation (FAPESP) under grant 2020/07145-8. Antonio A. Chaves was supported by FAPESP under grants 2018/15417-8 and 2016/01860-1, and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under grants 423694/2018-9 and 303736/2018-6. Leandro C. Coelho was supported by the Natural Sciences and Engineering Council of Canada under grant 2019-00094. We also thank Compute Canada for providing parallel computing facilities. We thank our contacts at the dial-a-ride operator for providing valuable insights and real data.

---

* Corresponding author: leandro.coelho@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2021

1. Introduction

Urban mobility is one of the most important contemporary challenges. Although technological development allows new transportation options, elderly and disabled people have a specific demand due to their physical and mobility needs. These users require more responsive and flexible means, mainly door-to-door, besides imposing quality of service constraints, like duration, time window, and ride-time. On the other hand, transportation providers seek to offer an efficient service, which implies designing shared-ride vehicle routes with additional operational constraints such as capacity, pairing, and precedence. Historically, this has been solved as a dial-a-ride problem (DARP) [9], and several heuristics [4, 10, 12, 22, 23, 24] and exact algorithms [3, 8, 31, 34, 35] have been proposed to solve its standard version and variants. For a deeper review and recent developments we refer to Ho et al. [19].

However, the increasing number of users over time has given rise to more significant concerns with respect to the service level offered [27]. Furthermore, the COVID-19 pandemic imposed a sharp capacity reduction in shared transportation systems to maintain a safe distance between passengers. To meet this new arrangement, providers end up resorting to more expensive means such as taxis and mobility-on-demand (MoD) services like Uber and Lyft.

In the freight transportation industry, the combination of an owned private fleet with extra vehicles is already considered within the scope of the vehicle routing problem (VRP) [11, 14]. Here, a common carrier serves as an outsourced transportation capacity that can be used on-demand. However, the human aspect in the DARP makes this fleet integration more complicated because it imposes different requirements than in a VRP. In addition, these routes transporting packages need to consider pairing, precedence, time window, and ride-time limits. These requirements are even more critical when dealing with users, especially those with restricted mobility as is the case of the elderly and the differently-abled.

In the passenger segment, the integration of modes involves, in most studies, some links with public transport. Nonetheless, the level of integration is different from the VRP with common carrier perspective since public transport can only be used on some legs, with no complete outsourcing of requests. In Posada et al. [30] an integrated DARP (IDARP) is considered where the intermediate legs of the trip are performed with regular buses, according to their timetables, while the first and last legs are performed by the private fleet. The authors compare two different models, but no dedicated algorithm was designed for the problem. Steiner and Irnich [40] propose a network planning optimization model for general users, where MoD services make the first and last legs while an existing public line performs intermediate ones. The problem takes into account a specific service-level requirement in terms of travel time. A branch-and-price algorithm was designed to solve the problem, and medium-sized instances with up to 20 bus lines were evaluated. A similar problem

was studied by Narayan et al. [26], differing in that flexible public transport can be used to perform any part of the trip. The paper captures dynamic aspects of the demand, and an agent-based simulation framework was proposed to solve a real scenario in Amsterdam, the Netherlands. More recently, Molenbruch et al. [25] propose a large neighborhood search-based algorithm to solve an IDARP with synchronization for rural areas and evaluated the method on an artificial data-set with realistic features. The algorithm also succeeded in solving related problems from the literature.

With the challenges imposed by the pandemic, the use of the maximum capacity of a vehicle is no longer ideal, given the need for the distance between passengers. In this sense, using extra vehicles from common carriers, observing the total service costs, is safer for users and drivers, besides enabling more efficient routes by combining them with the private fleet. To the best of our knowledge, no method explores the benefits of full modal integration in this new context of the problem.

In order to fill this gap, we optimize shared transportation services for door-to-door requests, satisfying operational and quality of service constraints when a provider can use a common carrier besides their own private fleet. Motivated by a real case, we introduce the dial-a-ride problem with private fleet and common carrier (DARP-PFCC), and propose a branch-and-cut (B&C) algorithm to solve it exactly. Due to its combinatorial aspect, we also design an efficient near parameter-free metaheuristic to handle more realistic instances of the DARP-PFCC. Our method is adapted from Chaves and Lorena [5] and combines a Biased Random-key Genetic Algorithm (BRKGA) with a Q-Learning (QL) algorithm (BRKGA-QL) within a new parallel framework. Since Gonçalves and Resende [15] proposed it, BRKGA has been successfully applied to a range of combinatorial optimization problems, such as container loading [16], project scheduling [18], parallel machine scheduling [6], facility layout [17], and vehicle routing [21]. However, whereas metaheuristics are strongly dependent on the problem at hand, the parameter setting is a complex and crucial step in their performance. In this sense, BRKGA-QL aims to release BRKGA from its parameter tuning phase.

The scientific contributions of this paper are:

1. We introduce a new DARP variant by combining private fleet and common carriers, and this approach is beneficial both for users who are served more quickly, safely, and comfortably, and for providers who can exploit a combined fleet to reduce their costs.
2. We design an improvement for the BRKGA-QL, exploiting the local search and the evolutionary processes in parallel.
3. We assess the performance of our algorithms, which are flexible enough to solve the standard DARP, on the classical instances of Cordeau [8] and Ropke et al. [35].
4. We propose a new instance set for the DARP-PFCC, and then analyze new operational constraints with different cost structures.

5. We solve real-case scenarios in the context of the DARP-PFCC through our metaheuristic and derive many managerial insights, both with respect to cost and service level.
6. The new instances are made publicly available.

The remainder of the paper is organized as follows. Section 2 formally describes the problem and provides a mixed-integer linear programming (MILP) formulation for the DARP-PFCC, and several valid inequalities are discussed in Section 3. The B&C algorithm is presented in Section 4, and we detail our BRKGA-QL and explain it new features in Section 5. In Section 6, we discuss the results of extensive computational experiments performed to assess the quality of the algorithms, and several managerial insights are derived based on the results. Section 7 presents our conclusions and directions for future works.

2. Problem definition and mathematical model

Based on the Ropke et al. [35], we define the DARP-PFCC as follows. Let n denote the number of transportation requests to be satisfied, and $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a directed graph. The node set is $\mathcal{N} = \{0, ..., 2n+1\}$, where 0 and $2n+1$ are the origin and destination depot nodes, while subsets $\mathcal{P} = \{1, ..., n\}$ and $\mathcal{D} = \{n+1, ..., 2n\}$ are the sets of pickup and delivery nodes, respectively. The arc set is $\mathcal{A}$, where each arc $(i, j) \in \mathcal{A}$ has a routing cost/distance c_{ij} and a travel time t_{ij}, both respecting the triangle inequality.

Each request i is known *a priori* and is associated with a ride-time limit L_i, a pickup origin $i \in \mathcal{P}$, and a delivery destination $j \in \mathcal{D}, j = n + i$. Each node $i \in \mathcal{N}$ has a demand q_i, a service duration d_i, and a time window $[e_i, l_i]$ where e_i and l_i define the earliest and the latest times to serve i. Time windows can also be imposed on delivery nodes. For the depot, time windows define the earliest and latest times a vehicle must depart $[e_0, l_0]$ and return $[e_{2n+1}, l_{2n+1}]$. The following parameters are used: $d_0 = d_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$, and $q_i \geq 0, q_i = -q_j, i \in \mathcal{P}, j \in \mathcal{D}, j = n + i$.

A limited private (owned) fleet of m homogeneous vehicles with capacity Q is available at the origin depot. As a novel approach in the DARP, transportation providers manage all requests and decide if extra vehicles from a common carrier should also serve users. We consider an unlimited and heterogeneous fleet, with enough capacity to satisfy any request directly from node i to $n + i$, i.e., a direct pickup and delivery route. When a request i is serviced by a common carrier, a fixed cost f_i and a variable cost v_i are charged, so the associated extra cost is given by $f_i + v_i(c_{i,n+i})$.

Let $\mathcal{S}$ be the set of all node subsets $S \subseteq \mathcal{N}$, in which $0 \in S$, $2n + 1 \notin S$, and at least one node $i \in \mathcal{P}$ with $i \notin S$ and $n + i \in S$. The total demand for a subset $S \subseteq \mathcal{N}$ is given by $q(S) = \sum_{i \in S} |q_i|$, so that a lower bound on the number of times vehicles must enter and leave S in order to serve all their nodes is

$\max\{1, \lceil q(S)/Q \rceil\}$. Finally, let $\mathcal{R}$ the set of infeasible paths with respect to route duration, time windows, and ride-times, and let $\mathcal{A}(R)$ represent all arcs (i,j) of $R \in \mathcal{R}$. The decision variables are:

- $X_{ij} = 1$ if a vehicle from the private fleet travels directly from i to j, with $(i,j) \in \mathcal{A}$, 0 otherwise.
- $Y_i = 1$ if node i is served by a common carrier vehicle which travels directly from $i \in \mathcal{P}$ to $n+i$, 0 otherwise. Equivalently, $Y_j, j \in \mathcal{D}, j = n+i$ are the auxiliary variables associated with Y_i.

The DARP-PFCC can be formulated by (1)−(11):

$$\min \sum_{(i,j)\in\mathcal{A}} c_{ij}X_{ij} + \sum_{i\in\mathcal{P}}[f_i + v_i(c_{i,n+i})]Y_i \tag{1}$$

subject to

$$\sum_{j\in\mathcal{N}} X_{0j} \leq m \tag{2}$$

$$\sum_{i\in\mathcal{N}} X_{i,2n+1} \leq m \tag{3}$$

$$\sum_{i\in\mathcal{N}} X_{ij} + Y_j = 1 \qquad j \in \mathcal{P}\cup\mathcal{D} \tag{4}$$

$$\sum_{j\in\mathcal{N}} X_{ij} + Y_i = 1 \qquad i \in \mathcal{P}\cup\mathcal{D} \tag{5}$$

$$Y_i = Y_{n+i} \qquad i \in \mathcal{P} \tag{6}$$

$$\sum_{i,j\in S} X_{ij} \leq |S| - 2 \qquad S \in \mathcal{S} \tag{7}$$

$$\sum_{i,j\in S} X_{ij} \leq |S| - \max\{1, \lceil |q(S)|/Q \rceil\} \qquad S \subseteq \mathcal{P}\cup\mathcal{D}, |S| \geq 2 \tag{8}$$

$$\sum_{(i,j)\in\mathcal{A}(R)} X_{ij} \leq |\mathcal{A}(R)| - 1 \qquad R \in \mathcal{R} \tag{9}$$

$$X_{ij} \in \{0,1\} \qquad (i,j) \in \mathcal{A} \tag{10}$$

$$Y_i \in \{0,1\} \qquad i \in \mathcal{P}\cup\mathcal{D}. \tag{11}$$

The objective function (1) minimizes the sum of private fleet and common carrier routing costs. Constraints (2) and (3) limit the number of private vehicles used. Constraints (4) and (5) ensure that each node is visited exactly once. Constraints (6) guarantee that if a user is picked up by a common carrier, the delivery

is performed by it, while (7) impose precedence constraints, i.e., a pickup node $i \in \mathcal{P}$ is visited before its corresponding delivery node $j = n + i, j \in \mathcal{D}$, and both are served by the same vehicle (see [36]). Rounded capacity constraints are defined by (8) and infeasible paths are eliminated with constraints (9). Finally, constraints (10) and (11) define the variables domain.

Considering the set of infeasible paths $\mathcal{R}$, Ropke et al. [35] present an improved formulation to eliminate them. For an infeasible path $R = (k_1, ..., k_r)$, constraint (9) can be strengthened based on the tournament constraints of Ascheuer et al. [1] and directly extended to the DARP-PFCC as follows:

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} X_{k_i,k_j} \leq |A(R)| - 1. \tag{12}$$

Now consider a path $R = (i, k_1, ..., k_r, n + i)$ connecting nodes i and $n + i$, and violating time window or ride-time constraints for these nodes. Because the triangular inequality is respected, R can be removed by imposing constraint (13):

$$X_{i,k_1} + \sum_{h=1}^{r-1} X_{k_h,k_{h+1}} + X_{k_r,n+i} \leq |A(R)| - 2. \tag{13}$$

Finally, if both $R = (k_1, ..., k_r)$ and its reverse path $R' = (k_r, ..., k_1)$ are infeasible, then (14) is valid:

$$\sum_{i=1}^{r-1} (X_{k_i,k_{i+1}} + X_{k_{i+1},k_i}) \leq r - 1. \tag{14}$$

3. Valid inequalities

Cordeau [8] and Ropke et al. [35] present a set of valid inequalities for the DARP that can be easily extended to the DARP-PFCC. They are subtour elimination constraints (SEC), generalized order constraints (GOC), strengthened infeasible path constraints (SIPC), and fork constraints (FC). These inequalities are not required to define a feasible solution but help provide tighter lower bounds for the problem. The following suitable notation is employed:

- $X(S) = \sum_{i,j \in S} X_{ij}$ is the sum of the arcs between all nodes in S
- $\overline{S} = S \backslash N$ is the complementary set of $S \subseteq N$
- $N(R)$ is the set of nodes of path R.

3.1. Subtour elimination constraints

Cordeau [8] employs precedence constraints for i and $n+i$ to lift classical SEC for the DARP. In this sense, the author defines $\pi(S) = \{i \in \mathcal{P} : n+i \in S\}$ and $\sigma(S) = \{n+i \in \mathcal{D} : i \in S\}$ as the predecessors and successors, respectively, for any subset $S \subseteq \mathcal{P} \cup \mathcal{D}$. Then, the following inequalities are introduced:

$$X(S) + \sum_{i \in \overline{S} \cap \sigma(S)} \sum_{j \in S} X_{ij} + \sum_{i \in \overline{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} X_{ij} \leq |S| - 1 \tag{15}$$

$$X(S) + \sum_{i \in S} \sum_{j \in \overline{S} \cap \pi(S)} X_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in \overline{S} \setminus \pi(S)} X_{ij} \leq |S| - 1. \tag{16}$$

3.2. Generalized order constraints

Ruland and Rodin [36] introduced this inequality for the pickup and delivery problem (PDP), which can also be extended for the DARP-PFCC. Let $U_1, ..., U_v \subset \mathcal{N}$ be mutually disjoint subsets and let $i_1, ..., i_v \in \mathcal{P}$ be requests, such that $0, 2n+1 \notin U_l$ and $i_l, n+i_{l+1} \in U_l$ for $l = 1, ..., v$, where $i_{v+1} = i_1$. Generalized order constraints can be formulated as follows:

$$\sum_{l=1}^{v} X(U_l) \leq \sum_{l=1}^{v} |U_l| - v - 1. \tag{17}$$

Considering the same subsets $U_1, ..., U_v \subset \mathcal{N}$, Cordeau [8] also presents two lifted forms of constraint (17) for the DARP when the graph is directed, which are presented next.

$$\sum_{l=1}^{v} X(U_l) + \sum_{l=2}^{v-1} X_{i_1, i_l} + \sum_{l=3}^{v} X_{i_1, n+i_l} \leq \sum_{l=1}^{v} |U_l| - v - 1 \tag{18}$$

$$\sum_{l=1}^{v} X(U_l) + \sum_{l=2}^{v-2} X_{n+i_1, i_l} + \sum_{l=2}^{v-1} X_{n+i_1, n+i_l} \leq \sum_{l=1}^{v} |U_l| - v - 1. \tag{19}$$

3.3. Strengthened infeasible path constraints

Let $R = (i_1, ..., i_q)$ be a feasible path satisfying time window constraints. Taking into account precedence constraints, the nodes of subset $\pi(N(R)) \setminus N(R)$ must be visited before the nodes of $N(R)$, while the nodes of subset $\sigma(N(R)) \setminus N(R)$ must be visited after those of $N(R)$. Let $\phi(S)$ be the set of all permutations of the nodes of S. If path $R' = (\phi_p, R, \phi_d)$ is infeasible for all permutations $\phi_p \in \phi(\pi(N(R)) \setminus N(R))$ and for all permutations $\phi_d \in \phi(\sigma(N(R)) \setminus N(R))$, then R is infeasible for any solution of the DARP-PFCC and can be removed by constraint (12).

3.4. Fork constraints

Fork constraints introduced by Ropke et al. [35] are also valid for the DARP-PFCC. They remove infeasible paths by identifying groups of infeasible paths that have arcs in common. Consider a feasible path $R = (k_1, ..., k_r)$, and an infeasible path $R' = (i, R, j)$, for $i \in S$ and $j \in T$, with $S, T \subset \mathcal{N}$. So the following inequality is valid:

$$\sum_{i \in S} X_{i,k_1} + \sum_{h=1}^{r-1} X_{k_h,k_{h+1}} + \sum_{j \in T} X_{k_r,j} \leq r. \tag{20}$$

Ropke et al. [35] also present two strengthened versions of constraint (20). The first one is called outfork inequality and associates with each intermediate nodes $k_2, ..., k_{r-1}$ a set of nodes that lead to infeasible paths. Let $S, T_1, ..., T_r \subset \mathcal{P} \cup \mathcal{D}$ be subsets with $j = 2, ..., r$, and $k_j \notin T_{j-1}$. If for any pair of nodes $i \in S, j \in T_h$ and for any integer value $h \leq r$, path $R' = (i, k_1, ..., k_h, j)$ is infeasible, then it can be removed. Based on this conclusion, the following inequality is valid for the DARP-PFCC:

$$\sum_{i \in S} X_{i,k_1} + \sum_{h=1}^{r-1} X_{k_h,k_{h+1}} + \sum_{h=1}^{r} \sum_{j \in T_h} X_{k_h,j} \leq r. \tag{21}$$

The second version is called infork inequality and is very similar to the previous one. This inequality is formulated considering arcs that reach path R. Here, we consider subsets $S_1, ..., S_r, T \subset \mathcal{P} \cup \mathcal{D}$ with $j = 1, ..., r-1$ and $k_j \notin S_{j+1}$. If for any integer value $h \leq r$ and any pair of nodes $i \in S_h, j \in T$, path $R' = (i, k_h, ..., k_r, j)$ is infeasible, then inequality (22) is valid for the DARP-PFCC.

$$\sum_{h=1}^{r} \sum_{i \in S_h} X_{i,k_h} + \sum_{h=1}^{r-1} X_{k_h,k_{h+1}} + \sum_{j \in T} X_{k_r,j} \leq r. \tag{22}$$

4. Branch-and-cut algorithm

The number of constraints (7)−(9) grows exponentially with the number of requests, and their full enumeration for large instances is impracticable. Alternatively, they can be dynamically generated and added whenever these constraints are violated. This branch-and-cut (B&C) scheme was previously used to solve the DARP [3, 8, 35] and we extend it to solve the DARP-PFCC. Furthermore, our algorithm extends all preprocessing techniques of Dumas et al. [13] and Cordeau [8] to the DARP-FCC. These techniques contemplate a time window tightening procedure (Section 4.1), which provides better lower bounds, and arc removal

procedures (Section 4.2), which analyze aspects of precedence, time window, demand, and user ride-times to reduce the size of the problem by removing several arcs. Section 4.3 describes the separation procedures of the many cuts and valid inequalities, and Section 4.4 presents the general framework of our B&C algorithm.

4.1. Time window tightening

As usual in several algorithms for the DARP [8, 20, 35] we consider that users can define an inbound (outbound) time window, valid for the pickup (delivery) node, but not in both. Let T be the time limit to complete the service of the last delivery. Based on Cordeau [8], inbound time windows for the DARP-PFCC can be tightened by imposing $e_{n+i} = max\{0, e_i + d_i + t_{i,n+i}\}$ and $l_{n+i} = min\{l_i + d_i + L, T\}$ on the delivery nodes. Equivalently, $e_i = max\{0, e_{n+i} - L - d_i\}$ and $l_i = min\{l_{n+i} - t_{i,n+i} - d_i, T\}$, works for the pickup nodes. Finally, time windows for the depot nodes 0 and $2n+1$ can be tightened by imposing $e_0 = e_{2n+1} = min\{e_i - t_{0,i}\}$ and $l_0 = l_{2n+1} = max\{l_i + d_i + t_{i,2n+1}\}$.

4.2. Arc removal

Considering pairing, precedence, demand, time windows, and ride-time constraints, a set of infeasible arcs $(i,j) \in \mathcal{A}$ can be removed from the search space. The following arc removal procedures were introduced by Dumas et al. [13] and Cordeau [8], and are valid for the DARP-PFCC.

- Arcs $(i,0), (2n+1,i)$, and (i,i), with $i \in \mathcal{N}$ are never used and can be removed from the model.
- *Precedence-pairing*: Following precedence and pairing relations, pickup node i must be visited prior to its associated delivery node $n+i$. Then arcs $(0,n+i), (i,2n+1)$, and $(n+i,i)$, with $i \in \mathcal{P}$ can be removed.
- *Vehicle capacity*: If $q_i + q_j > Q$, with $i \neq j \in \mathcal{P}$, then arcs $(i,j), (j,i), (i,n+j), (j,n+i), (n+i,n+j)$, and $(n+j,n+i)$ can be removed.
- *Time windows*: If $e_i + d_i + t_{i,j} > l_j$, with $i \neq j \in \mathcal{N}$, arc (i,j) can never be used and can be removed.
- *Ride-time*: If $t_{ij} + d_j + t_{j,n+i} > L$, with $i \in \mathcal{P}$ and $j \in \mathcal{N}$, then (i,j) and $(j,n+i)$ can be removed.
- *Infeasible paths*: Considering time windows and pairing constraints for $i \neq j \in \mathcal{P}$, we can remove:
  - i) arc $(i,n+j)$, if path $(j,i,n+j,n+i)$ is infeasible;
  - ii) arc $(n+i,j)$, if path $(i,n+i,j,n+j)$ is infeasible;
  - iii) arc (i,j), if paths $(i,j,n+i,n+j)$ and $(i,j,n+j,n+i)$ are both infeasible;
  - iv) arc $(n+i,n+j)$, if paths $(i,j,n+i,n+j)$ and $(j,i,n+i,n+j)$ are both infeasible.

Additional arcs can be removed based on the interaction of time window, ride-time, and vehicle capacity constraints. To this end, it is enough to identify all pairs of users who cannot be served by the same vehicle. Consider pickup nodes $i, j \in \mathcal{P}$ and their associated delivery nodes $n+i, n+j \in \mathcal{D}$. The paths respecting precedence constraints are $(i, j, n+i, n+j)$, $(i, j, n+j, n+i)$, $(j, i, n+i, n+j)$, $(j, i, n+j, n+i)$, $(i, n+i, j, n+j)$, and $(j, n+j, i, n+i)$. If all these paths are infeasible, arcs $(i, j), (i, n+j), (n+i, j), (n+i, n+j), (j, i), (j, n+i), (n+j, i)$, and $(n+j, n+i)$ can be removed.

Assessing the feasibility of a specific path $R = (i, i+1, ..., q)$ is not a trivial task, especially in the presence of ride-time constraints, as in the DARP-PFCC. We must determine the service start time B_i for each node i. However, defining $B_i = max\{e_i, A_i\}$, where A_i is the vehicle arrival time in $i \in \mathcal{P}$, is not sufficient to ensure a feasible route (if one exists).

It can often be advantageous to appropriately wait before the departure from the origin depot, after performing a delivery, or before starting the service at a pickup node, in order to minimize the route total duration and the unnecessary waiting times at the subsequent nodes (see [9]). In this context, we must compute for each node i its forward time slack F_i, which indicates the maximum delay for the start of service without violating the time windows for the subsequent nodes. Cordeau [8] presents a generalization to F_i introduced by Savelsbergh [37] for the travelling salesman problem with time windows, and it can be directly applied to the DARP-PFCC.

$$F_i = \min_{i \leq j \leq q} \left\{ \sum_{i < p \leq j} W_p + min\{l_j - B_j, L - P_j\} \right\}. \tag{23}$$

In equation (23), W_i represents the waiting time at node i and P_j is the user ride-time where the destination node is j, if $n+1 \leq j \leq 2n$, and $P_j = -\infty$, otherwise.

4.3. Separation procedures

In this section, we present the heuristic and exact procedures to check if constraints (7)−(9) and also valid inequalities presented in Section 3 are violated along the search on the branch-and-bound (B&B) tree. These procedures were proposed by Cordeau [8] and Ropke et al. [35] and can be extended to the DARP-PFCC. For presentation proposes, we consider: $\delta^-(S) = \{(i, j) \in \mathcal{A} : i \notin S, j \in S\}$, $\delta^+ = \{(i, j) \in \mathcal{A} : i \in S, j \notin S\}$, and $\delta(S) = \delta^-(S) \cup \delta^+(S)$, with $S \subseteq \mathcal{N}$.

4.3.1. Precedence constraints

Ropke et al. [35] point out that constraints (7) are equivalent to $X(\delta^-(S)) \geq 1$, for $S \in \mathcal{S}$, because $X(S) = |S| - 1 - X(\delta^-(S))$ and $0 \in S$. We can identify exactly if any equivalent constraint has been violated by solving maximum flow (MF) problems as follows. For each request i, we compute the MF from nodes i and $2n+1$ to nodes 0 and $n+i$ on a support graph $\mathcal{G}^* = (\mathcal{N}, \mathcal{A}^*)$, where the arc capacity is given by the LP solution for the X^*_{ij} variables. If $MF < 1$, a violated precedence constraint is identified, and S corresponds to one shore of the minimum cut, such that $0, n+i \in S$ and $i, 2n+1 \notin S$.

4.3.2. Capacity constraints

Inspired by Augerat et al. [2], Cordeau [8] proposed a tabu search heuristic to identify violated capacity constraints (8) for the classical DARP, and we extend it to the DARP-PFCC. The algorithm considers a random subset $S \subseteq \mathcal{P}$ or $S \subseteq \mathcal{D}$, and at each iteration, a node is either removed or added in S so that $X(\delta(S))$ is minimized and $q(S) > Q$. The algorithm controls repeated insertions and reinsertions, ensuring that a node added to (deleted from) S is declared tabu for a given number of iterations, prohibiting reverse movements. The algorithm runs for a given number of iterations (e.g., 25), allowing the identification of multiple violated constraints.

Additionally, Ropke et al. [35] designed a randomized construction heuristic to identify if constraints (9) are violated. It selects a node from $\mathcal{P} \cup \mathcal{D}$ in $\mathcal{G}^*$, and adds it to S with a probability of being selected proportional to the flow on the corresponding arc, iteratively checking for any violation.

4.3.3. Infeasible path constraints

Violated constraints (9) can be separated through a path construction heuristic $R_i = (i, k_1, k_2, ...)$ departing from each $i \in \mathcal{P}$. The algorithm adds a node k_j adjacent in $\mathcal{G}^*$ to the previous added node k_i, such that $X^*_{k_i,k_j}$ is maximized. The procedure stops when $n+i$ or $2n+1$ is reached, or when a subtour containing i is formed. We evaluate if time window and ride-time constraints are violated whenever $n+i$ is reached.

4.3.4. Subtour elimination constraints

Violated lifted inequalities (15) and (16) can also be separated through a tabu search heuristic. Cordeau [8] considers the fact that in an integer solution, $2X(S) + X(\delta(S)) = 2|S|$. Based on this, violated inequalities (15) can be identified by finding a set of nodes S, such that:

$$X(\delta(S)) - 2\left(\sum_{i \in \overline{S} \cap \sigma(S)} \sum_{j \in S} X_{ij}\right) - 2\left(\sum_{i \in \overline{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} X_{ij}\right) < 2. \tag{24}$$

From an empty set S, the procedure iteratively either adds or removes (when $|S| > 0$) a node from S, minimizing the left-hand side of (24). A similar method can separate violated valid inequalities (16).

4.3.5. Generalized order constraints

As in Ropke et al. [35], violated lifted GOC (18) and (19) can be separated by applying two simple heuristics, both only covering the case when $v = 3$ and $|U_1| = |U_2| = |U_3| = 2$. For each request i, the first heuristic identifies another request j, such that $X_{i,n+j} + X_{n+j,i} + X_{ij}$ is maximized. Then, it finds another request k that maximizes the left-hand side of (18). The second heuristic is similar and identifies a request j for each i, such that $X_{i,n+j} + X_{n+j,i} + X_{n+i,n+j}$ is maximized. Besides, it also identifies another request k which maximizes the left-hand side of (19).

4.3.6. Fork constraints

Ropke et al. [35] describe two separation procedures to identify violated FC, and we extended them to the DARP-PFCC. For $r = 1$ (see inequality (21)), a procedure identifies violated inequality (21) by first enumerating the set H of all infeasible paths containing three nodes connected in $\mathcal{G}^*$. Then, from each arc (i, j) with $X^*_{ij} > 0$, the algorithm creates a path S with all nodes h such that $(h, i, j) \in H$. Finally, set T_1 is created by adding all nodes k, with $(h, i, k) \in H$ and for $h \in S$. Equivalently, violated inequality (22) can be separated through a set T of all nodes k, with $(i, j, k) \in H$, following the identification of the set S_1 of nodes h, with $(h, j, k) \in H$.

When $r \geq 2$, violated FC can be separated through a heuristic procedure. Departing from each node $k_0 \in \mathcal{P} \cup \mathcal{D}$, this heuristic creates feasible paths $(k_0, k_1, ..., k_{l-1}, k_l)$ by adding a node k_l, such that $X^*_{k_{l-1},k_l} > 0$ in $\mathcal{G}^*$. Violated inequalities are separated based on the current path, by creating a set T of nodes j, such that $(k_0, k_1, ..., k_{l-1}, k_l, j)$ is an infeasible path. Then, node k_0 is inserted in S, jointly with node i, such that all paths $(i, k_1, ..., k_{l-1}, k_l, j)$ are infeasible for each $i \in S$ and $j \in T$. Path $(k_1, ..., k_{l-1}, k_l)$ and sets S and T describe a simple fork inequality (20). To lift it to an outfork (infork) inequality, we can add to sets $T_1, ..., T_l$ and $S_1, ..., S_l$ as many nodes as possible, as long as set definition of Section 3.4 is respected. Similar to Ropke et al. [35], in order to identify violated constraints in a reasonable computational time, we limit the path length to six vertices.

4.4. General framework of the B&C

At the root node, we generate and solve the linear programming (LP) relaxation of the model from Section 2, without constraints (7)−(9). Whenever a node is optimized, a search for violated constraints, including SEC, GOC, SIPC, and FC inequalities of Section 3, is performed. For this purpose, we use the cut separation

procedures presented in Section 4.3. These routines return violated inequalities or prove, at best, that there are no violations in the current solution of the LP relaxation. The process continues until a feasible or dominated solution is found or there are no more inequalities to add. At this point, branching occurs on a fractional variable, and two new subproblems (nodes) are generated. Then a new node is selected and optimized. The algorithm ends with no more nodes to be optimized, and the optimal solution is proven, or by reaching a time limit. In **Algorithm 1** we present the pseudocode of our B&C.

Algorithm 1 Pseudocode of the proposed B&C algorithm

```
Apply time window tightening and arc removal procedures.
At the root node, generate (1)−(6) ∪ (10)−(11).
Solve the LP relaxation of the node.
Termination check:
if there are no more nodes to evaluate then
    Stop.
else
    Select one node from the B&C tree.
end if
while the solution of the LP relaxation violates (7)−(9) or a valid inequality from Section 3 do
    Identify violated inequalities and add them to the LP relaxation.
end while
if the solution of the current LP relaxation is integer then
    Go to Termination check.
else
    Branching: branch on one of the fractional variables.
    Go to Termination check.
end if
```

5. Metaheuristic algorithm

In this section, we detail our BRKGA-QL proposed to solve the DARP-PFCC. A BRKGA explores the solution space of a problem through an indirect search in an independent r-dimensional hypercube. An encoding/decoding procedure is employed to transform solutions from this hypercube to feasible solutions for the problem at hand. To solve a specific problem, we simply need to specify algorithm parameters, how to encode and decode a solution, and how to compute its corresponding fitness value. In our algorithm, a Q-Learning mechanism is used to control the many parameters of the classical BRKGA. We present an overview of its structure (Section 5.1), followed by the description of encoding and decoding procedures (Section 5.2), local search improvement used to polish promising solutions (Section 5.3), and a novel parallel optimization scheme (Section 5.4).

5.1. BRKGA-QL overview

BRKGA is an evolutionary heuristic introduced by Gonçalves and Resende [15] that provides a general and reusable framework for solving combinatorial optimization problems. As in other evolutionary algorithms, a chromosome is a vector structure capable of adequately representing a solution for the problem at hand, and the algorithm evolves a population P over a maximum number of iterations. The initial population is composed of p random keys vectors (chromosomes), where each position of the vector stores a uniform random value from $[0, 1)$. At each iteration, the population P is ranked according to their fitness, and chromosomes are partitioned into two groups: an elite solution set $P_e \subset P$ and a non-elite solution set $P \backslash P_e$. The next generation is composed of three population substrates: *i*) an elite subset P_e, from the previous generation, *ii*) a subset of mutant solutions P_m with a uniform random value from $[0, 1)$ on each position of vectors, and *iii*) a set P_c ($|P_c| = |P| - |P_e| - |P_m|$) of solutions formed by using the *parametrized uniform crossover* (see [39]) with a parent randomly selected from P_e and another from $P \backslash P_e$ from the previous generation. In this case, parameter ρ_e represents the probability of the offspring inheriting the vector component of the elite parent. A decoder function is applied at each new generation to transform each chromosome into a solution. The population is then partitioned into elite and non-elite, and the process is repeated. In general, a time limit is imposed as a stopping criterion.

Other than ρ_e, population sizes ($|P| = p$, $|P_e| = p \times p_e$, and $|P_m| = p \times p_m$) are also BRKGA parameters and must be set before solving the problem. In general, a user configures these values in a tuning phase, evaluating the solution obtained and choosing those values which yield the best solution. However, it should be noted that once defined, these values remain fixed throughout the method's evolutionary process. Chaves et al. [7] pointed out that different values may perform better during different stages of the search. Additionally, optimal values for solving a given instance may not be appropriate for another one. Thus, determining these values can be a task as difficult as the algorithm implementation itself. To overcome this limitation, QL is employed to help find the best parameter configuration, and the tuning phase is dynamically performed along the optimization process.

QL is a reinforcement learning algorithm developed by Watkins [41], which employs a Markov process framework to model an agent learning process within an environment in terms of states, actions, and rewards. For each state s, the algorithm makes a decision a following a $Q(s, a)$ policy. The choice of the action follows the ϵ-greedy policy [41], guided by the exploration-exploitation paradigm, often choosing good actions and eventually selecting actions that lead to discovering new policies, potentially better than existing ones. The learning process updates $Q(s, a)$ from reward ρ obtained by decision a in state s, guided by the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + l_f \cdot [\rho + d_f \cdot max_{a'} Q(s',a') - Q(s,a)]. \tag{25}$$

In (25), the learning factor l_f controls how fast the algorithm learns, while the discount factor d_f is used to balance immediate and future rewards. After choosing a state-action value in current the stage, s' indicates the new state. Initially, the Q-table is generated with null values.

Our problem-dependent subroutines are inspired by Chaves and Lorena [5], who coupled QL as a control module to manage all parameters involved in the integrated BRKGA-QL. The BRKGA parameters controlled are p, p_e, p_m, and ρ_e, whereas the QL parameters are learning (l_f) and discount (d_f) rates, and the exploration-exploitation parameter ϵ. The authors consider a single state for each parameter, while actions follow recommended values from the literature, as present in Table 1. The population size p is defined by some Fibonacci values. Thus, it is possible to control population growth without adding a new rate.

Table 1: Action list of parameters [5]

Parameter	**Action List**
p	{233, 377, 610, 987, 1597, 2584}
p_e	{0.30, 0.25, 0.20, 0.15, 0.10}
p_m	{0.25, 0.20, 0.15, 0.10, 0.05}
ρ_e	{0.80, 0.75, 0.70, 0.65, 0.60, 0.55}
ϵ	{0.10, 0.15, 0.20, 0.25, 0.30}
l_f	{0.2, 0.4, 0.6, 0.8, 1.0}
d_f	{0.2, 0.4, 0.6, 0.8, 1.0}

5.2. Encoding and decoding

In the DARP-PFCC, we need to satisfy a set of n requests with a pickup and delivery pair $(i, n+i)$, either by a fleet of m private vehicles or by a common carrier. A solution can be represented by a chromosome having $n+1$ genes. Each gene i has a uniform random value from $[0, 1)$, called random-key (r_i), and $i = \{1, ..., n\}$ represents a request in the format $(i, n+i)$. In order to increase solution diversity, we have developed four different decoder functions, whose choice is dependent on the value of gene $n+1$. On the interval $[0; 0.25)$, decoder one is selected, while $[0.25; 0.50)$ selects decoder two, values within $[0.50; 0.75)$ choose decoder three, and decoder four is selected when the value of gene $n+1$ is greater than or equal to 0.75.

Decoders are specific constructive heuristics capable of producing slightly different solutions. Their choice is part of the evolutionary process, so decoders that generate better solutions are invoked more often. At the

same time, the evolutionary process alternates between decoders, diversifying the set of solutions throughout the optimization process. We present next the four decoders designed for the DARP-PFCC.

1. **Insert-end:** This decoder operates in two stages. First, it assigns each request $i \in \mathcal{P}$ to a private fleet vehicle $\nu = \lfloor (r_i)m \rfloor + 1$, where m is the fleet size. The second stage designs routes for each vehicle ν as follows: let Δ_ν be the set of requests assigned to ν, and Δ'_ν be the list of requests, sorted by r_i (from lowest to highest). Vehicle ν departs from the origin depot 0, and returns to destination depot $n+1$. Then, the first request $i \in \Delta'_\nu$ is inserted in the route, yielding the partial path $R = (0, i, n+i, 2n+1)$. After the first insertion, the next element of Δ'_ν is evaluated based on the cheapest insertion cost, considering the path from the previously inserted pickup i to destination depot $2n+1$. Let c_j and c_{j+n} be the cheapest insertion costs for the pickup and the delivery nodes of request j, respectively. Let also f_j and v_j be the fixed and variable costs to serve j by a common carrier vehicle, respectively. If $[c_j + c_{j+n}] > [f_j + v_j(c_{j,n+j})]$, or if any pairing, time window, or ride-time constraint is violated, request j is transferred to a common carrier vehicle. Inserting a request after the last pickup avoids recalculating temporal information such as service time and travel time for the entire route. We only need to consider the vicinity of the insertion and the remaining nodes of the routes.
2. **Insert-best:** This decoder is quite similar to the previous one, except that the insertion cost takes into account the whole path, from 0 to $2n+1$. Here the computational effort is more significant since all timing information must be calculated for each new request insertion evaluation.
3. **Insert-all:** This decoder is even more general than the second one since it considers all requests at once for each vehicle, and enumerates all possible sequences, i.e., it does not perform any sequential request assignment. The transfer to common carrier vehicles is also made depending on cost or infeasibility.
4. **Insert-own:** This decoder is dedicated to maximizing the usage of the private fleet. Therefore it operates similarly to **Insert-all**, but a request is transferred to a common carrier only in case of infeasibility.

Consider an example of an instance with $n = 8$ requests ($\mathcal{P} = \{1, ..., 8\}$, $\mathcal{D} = \{9, ..., 16\}$, with 0 and 17 being the origin and destination depots), and thus in a nine-element chromosome, shown in Figure 1a. We also consider a private fleet with $m = 2$ vehicles, implying the assignment rule $\nu = \lfloor (r_i) \cdot 2 \rfloor + 1$. Thus, when $r_i < 0.5$ (≥ 0.5), request i is assigned to $\nu = 1$ (2), yielding $\Delta_1 = \{1, 3, 6, 7, 8\}$ and $\Delta_2 = \{2, 4, 5\}$. The random-key of position $n+1$ indicates that decoder 1 (**insert-end**) should be applied ($r_9 = 0.19 < 0.25$).

Since the process differs only by the insertion criteria, Figure 1(b) describes the decoding for $\nu = 1$. After sorting by r_i, we have $\Delta'_1 = \{7, 3, 1, 6, 8\}$, and their corresponding delivery points $\{15, 11, 9, 14, 16\}$. We also consider $T = 120$, $Q = 3$, $L_i = 40, d_i = 3, q_i = 1, f_i = 3$, and $v_i = 2$ for each $i \in \mathcal{P}$. Table 2 reports time windows values (considering only inbound requests). For simplicity, we consider distances equal to times,

showing in Table 3 values only for the arcs evaluated in the decoding process (**insert-end**).

Considering the distances of Table 3, each insertion is presented step-by-step, based on the cheapest insertion cost. Light-blue circles indicate the last pickup point inserted (request number in red) and the last delivery point (number in black) in the route. Finally, $\mathcal{R}_{CC}$ indicates the set of requests served by the common carrier. According to the Δ'_1, request 7 is the first to be inserted. The insertion cost in the private fleet vehicle is $c_7 = c_{0,7}+c_{7,15}+c_{15,17}-c_{0,17} = 5+11+4-0 = 20$, which is smaller than the cost to serve it with the common carrier ($c'_7 = 3 + 2 \times c_{7,15} = 25$). The pickup node of the next request from Δ'_1 (node 3) can be inserted in any position after the last pickup node. The cheapest insertion cost is $c_3 = c_{7,3} + c_{3,11} + c_{11,15} - c_{7,15} = 15$, placing nodes 3 and 11 after the last pickup inserted in the route (node 7). A transfer to the common carrier is not advantageous because $c'_3 = 3+2\times c_{3,11} = 23$. On the other hand, the next request $(1, 9)$ is served by a common carrier vehicle ($\mathcal{R}_{CC} = \{1\}$), with $c'_1 = 3+2c_{1,9} = 13$. Next, pickup node 6 and its delivery node 14 are inserted in the private fleet vehicle route, with $c_6 = [c_{11,6} + c_{6,15} - c_{11,15}] + [c_{15,14} + c_{14,17} - c_{15,17}] = 19$, being the cheapest insertion cost. Departing from depot 0 at time 0, we can start the service in the last pickup node (6) at time 39. This value is greater than the opening time window of the next pickup in Δ'_1, $l_8 = 30$, forcing request $(8, 16)$ to be served by a common carrier vehicle. Thus, requests 1 and 8 are served by a common carrier, indicated by $\mathcal{R}_{CC} = \{1, 8\}$, and all other requests are served by the private fleet vehicle $v = 1$. The total cost is 96.

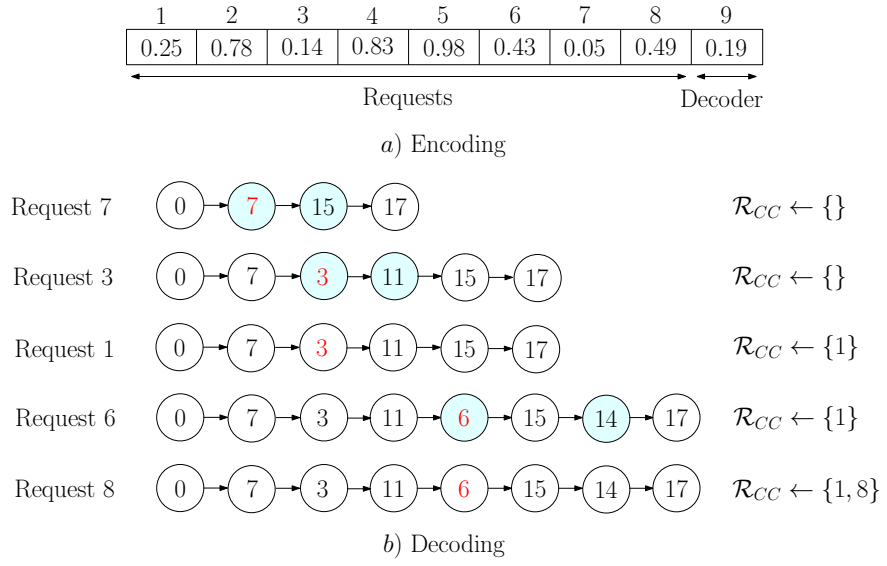

Figure 1: Encoding and decoding process

Table 2: Time windows of inbound requests

i	e_i	l_i	$n+i$	e_i	l_i
1	35	50	9	0	480
3	15	30	11	0	480
6	25	40	14	0	480
7	5	20	15	0	480
8	15	30	16	0	480

Table 3: Arcs used in the decoding procedure

i	j	$c_{ij} = t_{ij}$	i	j	$c_{ij} = t_{ij}$	i	j	$c_{ij} = t_{ij}$	i	j	$c_{ij} = t_{ij}$	i	j	$c_{ij} = t_{ij}$
0	7	5	3	17	16	8	15	14	11	15	8	15	8	19
0	17	0	6	8	12	8	16	13	11	17	15	15	9	21
1	9	5	6	11	14	8	17	15	14	8	16	15	11	17
1	11	16	6	14	17	9	11	17	14	11	14	15	14	9
1	15	13	6	15	6	9	15	16	14	15	18	15	16	16
1	17	14	6	17	15	9	17	15	14	16	15	15	17	4
3	1	18	7	3	8	11	1	13	14	17	9	16	14	15
3	6	13	7	15	11	11	6	7	15	1	14	16	15	18
3	11	10	7	17	6	11	9	15	15	3	16	16	17	19
3	15	9	8	14	19	11	14	20	15	6	15			

5.3. Local search improvement

In order to promote exploitation of promising areas of the search space, we also design a local search procedure as in Chaves et al. [7]. Our module is composed of seven operators, whose details are presented as follows.

1. **Assign a request to another vehicle:** This operator randomly selects a private fleet vehicle ν, and removes a request i served by this vehicle. Then i and $n+i$ are inserted in another private fleet vehicle ν', following the cheapest insertion rule. The insertion is performed when all pairing, precedence, time window, and ride-time constraints are preserved on the route of ν'.
2. **Swap intra-route:** The nature of the DARP-PFCC, particularly precedence and time windows, makes it unprofitable to perform exchanges between nodes close to the route endpoints. Hence, this operator performs the swap of up to four consecutive nodes within the same route. It randomly selects a private fleet vehicle ν and a node u visited by it. Let $\nu_R = (0, ..., u, v, w, z, ..., 2n+1)$ be the route of ν, and (u, v, w, z) be the four selected consecutive nodes. All 24 possible permutations are evaluated, and the most beneficial among the feasible ones is performed.
3. **Swap two requests inter-route:** This procedure randomly selects two private vehicles, ν and ν', and swaps two requests between them. Let request i served by ν, and $S_i = s_i^{\nu} + s_{n+i}^{\nu}$ be the total savings by removing the pickup and delivery nodes of i from ν. Equivalently, $S_j = s_j^{\nu'} + s_{n+j}^{\nu'}$ computes the total saving by removing request j from ν'. Insertion costs follow the cheapest insertion rule, and

$C_i = c_i^{\nu'} + c_{n+i}^{\nu'}$ is the lowest insertion cost of request i in vehicle ν', while $C_j = c_j^{\nu} + c_{n+j}^{\nu}$ represents the same for j in ν. Requests i and j are chosen so that $[(S_i + S_j) - (C_i + C_j)]$ is maximized, and all constraints are respected.

4. **Swap segments inter-route:** This operator considers a saving maximization associated with a swap inter-route as in the previous one. However, it swaps path segments between two vehicles. Path segments are natural sequences (see [28]) encompassing arcs where the vehicle load is zero.
5. **Remove a request from a private fleet vehicle:** It removes a request i from the route of a private fleet vehicle v and assigns its service to a common carrier vehicle. The most significant reduction in costs drives the request choice, that is, $[s_i^{\nu} + s_{n+i}^{\nu}] - [f_i + v_i(c_{i,n+i})]$ must be maximized.
6. **Remove a request from a common carrier vehicle:** This operator works in the opposite direction of the previous one by transferring a request from a common carrier vehicle to the private fleet. Consider a request i served by a common carrier vehicle, with service cost given by $f_i + v_i(c_{i,n+i})$. Let $C_i = c_i^{\nu} + c_{n+i}^{\nu}$ be the cost, following the cheapest insertion rule, of assigning request i to vehicle v. If $C_i < f_i + v_i(c_{i,n+i})$, a vehicle is chosen so that $(f_i + v_i(c_{i,n+i}) - C_i)$ is maximized (maximum saving), as long as no constraints are violated.
7. **Destroy** 1/3**:** This operator randomly removes up to 1/3 of the requests currently assigned to the private fleet and reinserts them in other positions into the same vehicle, or in other vehicles of the private fleet. The insertion is performed taking into account all vehicles and always observing feasibility and the cheapest insertion rule.

5.4. General framework of our parallel BRKGA-QL

At the beginning of processing, the time window tightening and arc removal procedures are applied to reduce the size of the problem (see Sections 4.1 and 4.2), and the Q-table starts with null values. An initial population P is created with p random-keys vector (chromosomes), each one representing a potential solution for the DARP-PFCC. When the population P is complete, the chosen decoder evaluates the fitness for each chromosome, identifying the best incumbent solution.

In the next step, we choose an action for each state from the Q-table, i.e., a value for each parameter, using the ϵ-greedy policy. In the evolutionary process, chromosomes are partitioned into the elite and non-elite population, copying the $|P_e|$ individuals to the next generation. Then, P_m and P_c are generated (see Section 5.1). The chosen decoder calculates the fitness for each new chromosome. If a new best solution is found in the population, a reward $\rho = 1$ is established, the Q-table is updated following the Bellman equation (25), completing the evolutionary process.

Whenever a new best solution is found (exploration) or if the best solution is not improved for 10 generations

(stagnation), we apply the Label Propagation (LP) method (Raghavan et al. [32]) to identify communities with similar chromosomes in the elite partition. The local search module is applied to the best chromosomes not yet improved from each community identified by the LP, where a Random Variable Neighborhood Descent (RVND, see [29]) manages the choice among different heuristics. Our algorithm stores the best solution, either obtained directly from the decoders or improved by local search. In order to preserve diversity in the evolutionary process, solutions obtained from the local search module are not added to the population of the BRKGA-QL. Besides, a random gene-switching move in chromosomes from P_e, which is not subject to local search improvement, is also included to increase the diversity through a perturbation operator.

In Chaves and Lorena [5], the evolutionary process only continues after processing all local searches in the identified solutions. Inspired by Schenekemberg et al. [38], who proposed a hybrid parallel framework by combining heuristic and exact algorithms to solve an inventory-routing problem, we also explore parallel computing resources, designing a non-sequential approach within the BRKGA-QL iterative process. Simultaneously to the evolutionary process, the local search module is invoked in parallel using the same RVND approach to process each candidate chromosome and polish solutions identified by the LP method. Figure 2 presents the general framework for the BRKGA-QL we propose for the DARP-PFCC.

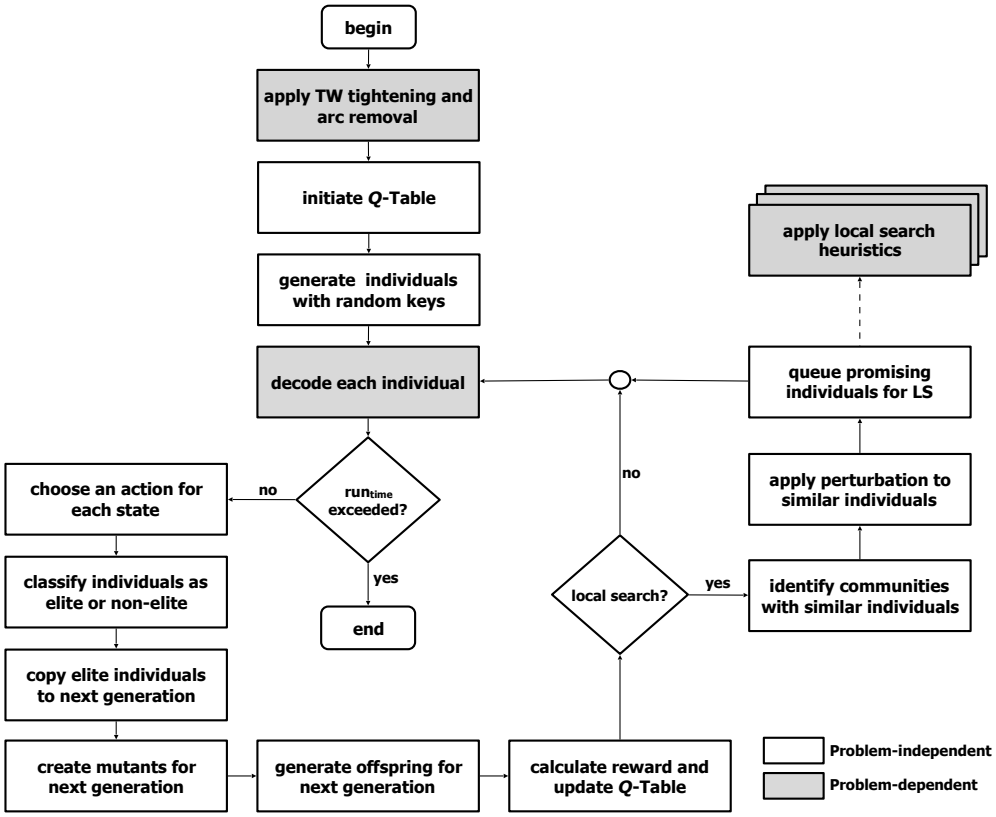

Figure 2: BRKGA-QL scheme

6. Computational experiments

In this section, we present the results of the extensive computational experiment we performed. As mentioned before, our algorithms are flexible enough to handle classical DARP instances, and the only adjustment consists in setting a big M value to the fixed and variable costs for a common carrier, f_i and v_i, respectively. We first discuss its results by solving the two sets adapted by Ropke et al. [35] from the basic set of Cordeau [8] with up to 96 requests (Section 6.1). Next, we analyze the results for the same instances adapted to the problem at hand and discuss some situations that impose restrictions on the full use of the fleet (Section 6.2). Finally, the results for a case study with up to 713 requests are presented, along with some managerial analyses (Section 6.3).

All algorithms were coded in C++, executed in computers equipped with Intel(R) Xeon(R), CPU E5-2683 v4 @ 2.10GHz. Mathematical models were solved by Gurobi Optimizer 9.1.0 with default parameters. Our BRKGA-QL was applied 30 runs (with different seeds) to each instance. The parameter run_{time} was defined as $2 \times n$ for tests with instances of the literature and $10 \times n$ for tests with instances of the case study. The instances and computational results are available at `https://www.leandro-coelho.com/darp/`. The source code for the BRKGA-QL (problem-independent) is available at `https://github.com/antoniochaves19/BRKGA-QL`.

6.1. Results for the classical DARP

These instances have up to 96 requests and 8 vehicles, divided into two sets (A and B) with 21 instances in each. In Set A, the vehicle capacity is $Q = 3$, demands are unitary ($q_i = 1$), service duration is $d_i = 3$, and maximum ride-time is $L = 30$ minutes. In Set B, the vehicle capacity is $Q = 6$, demand is randomly chosen from $\{1, ..., Q\}$, service time is proportional to the demand, i.e., $d_i = q_i$, and the maximum ride-time is $L = 45$ minutes. Both sets have tight time windows of 15 minutes. Starting from the maximum duration of the route (T), time windows $[e_i, l_i]$ are generated as follows: for an outbound user i, choose $l_{(n+i)}$ randomly in $[60, T]$ and define $e_{(n+i)} = l_{(n+i)} - 15$. For an inbound user, e_i is randomly chosen from $[0, T - 60]$ and $l_i = e_i + 15$. In all instances, depots coordinates are located at the origin $(0, 0)$, and pickup and delivery nodes coordinates are randomly chosen among $[-10, 10]^2$, following a uniform distribution. Travel time (t_{ij}) and routing cost (c_{ij}) are obtained from the Euclidean distance between i and j.

Tables 4 and 5 present the results for sets A and B, respectively. Optimal solutions are provided by Ropke et al. [35] (column Opt.). The columns UB and LB indicate the upper and lower bounds obtained by our B&C, the column Gap (%) reports the relative difference between these bounds, and the column T (s) reports the computational time in seconds. The columns UB Best and UB Avg indicate the values of the best solution

found by BRKGA-QL in 30 runs and its average. Column T (s) Best reports the average computational time of the BRKGA-QL to find the best solution. The columns Δ%UB Best and Avg. compute the relative percentage deviation defined by:

$$\Delta\%UB = \frac{(Z - Opt)}{Opt} \times 100, \tag{26}$$

where Z is UB Best or UB Avg. of our BRKGA-QL.

Our B&C found the optimal solution for all instances except b8-96. This is due to the fact that our B&C complies with the DARP-PFCC formulation, and has more variables and constraints than the classical DARP. BRKGA-QL found the optimal solution in 19 instances of set A and 20 instances of set B. The deviations of the three instances in which BRKGA-QL did not find the best-known solution was small; the worst case was 0.38% in instance a8-96. Furthermore, the deviations from the mean of the 30 runs were also very small; on average, it was 0.09% in set A and 0.03% in set B. BRKGA-QL found the optimal solution in all runs in 76% of the instances. The computational time of B&C and BRKGA-QL increases with the size of the instances. However, in larger instances, BRKGA-QL finds the best solution in approximately one minute, while B&C needs several minutes to prove optimality. The average computational time of B&C was 157 seconds for set A and 673 seconds for set B, while the average computational times of BRKGA-QL are 24 seconds and 20 seconds. We can observe that the BRKGA-QL had similar behavior in both sets, different from the B&C, which presents longer times in set B.

6.2. Results for the DARP-PFCC

In this study, we create three scenarios for the DARP-PFCC using the DARP instances. The same instances presented in Section 6.1 are used in the first scenario, allowing requests to be served by a common carrier. In the second scenario, it is possible to use a maximum of 2/3 of the capacity of the private fleet, considering social distancing in the context of the COVID-19 pandemic. In the third scenario, we consider one fewer vehicle in the fleet. Four combinations of fixed (f_i) and variable (v_i) costs (low-low, low-high, high-low, and high-high) regarding the common carrier are evaluated in each scenario, totaling 504 instances tested. In all cases, low equals 2 and high equals 3.5.

Tables 6 and 7 summarize the results for the DARP-PFCC. The columns are the same as in Tables 4 and 5, but considering the average result for the 21 evaluated instances. Detailed results for each instance are available at `https://www.leandro-coelho.com/darp/`. B&C found the optimal solution of all instances of set A. In set B the method did not find the optimal solution in 8 instances. Therefore, B&C was able to prove the optimality of 98.41% of the instances. We can observe that BRKGA-QL found solutions as good

Table 4: B&C and BRKGA-QL results for the DARP on instances of set A from Cordeau [8] and Ropke et al. [35]

Instance	Opt.	B&C				BRKGA-QL			Δ%UB	Δ%UB
		UB	LB	Gap (%)	T (s)	UB Best	UB Avg.	T (s) Best	Best	Avg.
a2-16	294.25	294.25	294.25	0.00	2.13	294.25	294.25	0.01	0.00	0.00
a2-20	344.83	344.83	344.83	0.00	3.12	344.83	344.83	0.02	0.00	0.00
a2-24	431.12	431.12	431.12	0.00	8.53	431.12	431.12	0.15	0.00	0.00
a3-24	344.83	344.83	344.83	0.00	18.31	344.83	344.83	0.26	0.00	0.00
a3-30	494.85	494.85	494.85	0.00	2.63	494.85	494.85	0.32	0.00	0.00
a3-36	583.19	583.19	583.19	0.00	8.11	583.19	583.19	0.91	0.00	0.00
a4-32	485.50	485.50	485.50	0.00	6.69	485.50	485.50	0.53	0.00	0.00
a4-40	557.69	557.69	557.69	0.00	19.82	557.69	557.69	0.82	0.00	0.00
a4-48	668.82	668.82	668.82	0.00	40.91	668.82	668.82	2.10	0.00	0.00
a5-40	498.41	498.41	498.41	0.00	14.12	498.41	498.41	0.57	0.00	0.00
a5-50	686.62	686.62	686.62	0.00	74.97	686.62	686.62	6.25	0.00	0.00
a5-60	808.42	808.42	808.42	0.00	67.20	808.42	808.42	4.29	0.00	0.00
a6-48	604.12	604.12	604.12	0.00	92.62	604.12	604.12	2.18	0.00	0.00
a6-60	819.25	819.25	819.25	0.00	71.76	819.25	819.25	24.37	0.00	0.00
a6-72	916.05	916.05	916.05	0.00	91.08	916.05	916.58	69.14	0.00	0.06
a7-56	724.04	724.04	724.04	0.00	78.03	724.04	724.04	22.43	0.00	0.00
a7-70	889.12	889.12	889.12	0.00	184.65	889.12	889.44	54.34	0.00	0.04
a7-84	1033.37	1033.37	1033.37	0.00	194.41	1034.42	1036.35	81.11	0.10	0.29
a8-64	747.46	747.46	747.46	0.00	149.94	747.46	751.34	67.45	0.00	0.52
a8-80	945.73	945.73	945.73	0.00	299.71	945.73	949.18	69.49	0.00	0.37
a8-96	1229.66	1229.66	1229.66	0.00	1884.33	1234.30	1237.76	98.00	0.38	0.66
Avg.	**671.78**	**671.78**	**671.78**	**0.00**	**157.09**	**672.05**	**672.69**	**24.04**	**0.02**	**0.09**

Table 5: B&C and BRKGA-QL results for the DARP on instances of set B from Cordeau [8] and Ropke et al. [35]

Instance	Opt.	B&C				BRKGA-QL			Δ%UB	Δ%UB
		UB	LB	Gap (%)	T (s)	UB Best	UB Avg.	T (s) Best	Best	Avg.
b2-16	309.41	309.41	309.41	0.00	7.24	309.41	309.41	0.12	0.00	0.00
b2-20	332.64	332.64	332.64	0.00	0.24	332.64	332.64	0.01	0.00	0.00
b2-24	444.71	444.71	444.71	0.00	7.89	444.71	444.71	0.25	0.00	0.00
b3-24	394.51	394.51	394.51	0.00	18.70	394.51	394.51	0.27	0.00	0.00
b3-30	531.44	531.44	531.44	0.00	7.42	531.44	531.44	0.36	0.00	0.00
b3-36	603.79	603.79	603.79	0.00	13.35	603.79	603.79	0.48	0.00	0.00
b4-32	494.82	494.82	494.82	0.00	10.65	494.82	494.82	0.34	0.00	0.00
b4-40	656.63	656.63	656.63	0.00	19.06	656.63	656.63	0.69	0.00	0.00
b4-48	673.81	673.81	673.81	0.00	164.80	673.81	673.81	2.96	0.00	0.00
b5-40	613.72	613.72	613.72	0.00	141.52	613.72	613.72	1.10	0.00	0.00
b5-50	761.40	761.40	761.40	0.00	199.06	761.40	761.40	3.74	0.00	0.00
b5-60	902.04	902.04	902.04	0.00	257.85	902.04	902.04	16.06	0.00	0.00
b6-48	714.83	714.83	714.83	0.00	11.92	714.83	714.83	0.90	0.00	0.00
b6-60	860.07	860.07	860.07	0.00	144.26	860.07	860.07	15.54	0.00	0.00
b6-72	978.47	978.47	978.47	0.00	866.18	978.47	978.47	34.05	0.00	0.00
b7-56	823.97	823.97	823.97	0.00	3398.48	823.97	823.97	21.34	0.00	0.00
b7-70	912.62	912.62	912.62	0.00	243.89	912.62	912.62	36.46	0.00	0.00
b7-84	1203.37	1203.37	1203.37	0.00	744.69	1203.37	1204.58	89.77	0.00	0.10
b8-64	839.89	839.89	839.89	0.00	487.40	839.89	840.42	50.08	0.00	0.06
b8-80	1036.34	1036.34	1036.34	0.00	190.82	1036.34	1036.64	74.45	0.00	0.03
b8-96	1185.55	1185.55	1162.46	1.95	7201.32	1187.27	1190.23	90.12	0.15	0.39
Avg.	**727.33**	**727.33**	**726.24**	**0.09**	**673.18**	**727.42**	**727.66**	**20.91**	**0.01**	**0.03**

as the best-known ones, showing an average deviation of 0.03% in set A and 0.004% in set B. A general analysis of the results shows that BRKGA-QL found the optimal solution in 451 instances proven by the B&C (91% of the total number of instances with optimal solution). Among the 8 instances in which B&C did not find the optimal solution, in two of them, the best solution found by BRKGA-QL was better than the upper bound obtained by the B&C. The BRKGA-QL was also robust, obtaining average solutions very close to the best solutions found (average gap of 0.13% in set A and 0.15% in set B). Furthermore, BRKGA-QL required 21% and 4% of the computational time spent by B&C in sets A and B, respectively.

It is interesting to note that considering the instances of set A (unit demand), it was possible to obtain solutions with total costs close to those found in the original scenario. When a vehicle is removed from the private fleet, we observe an average increase in costs of 3.06%, and when we decrease the capacity of vehicles to $2/3 \times Q$, we have an average increase in costs of only 1.15%. The instances of set B also present average costs close to the original scenario for fleet reduction (average increase of 3.66%). However, the instances with reduced capacity of this set incurred an average cost increase of 45.95%. This behavior is associated with the demands of requests that vary from 1 to Q. Thus, the reduction in vehicle capacity has a considerable impact on the operating costs due to the greater use of the outsourced fleet.

Table 6: B&C and BRKGA-QL results for the DARP-PFCC on instances of set A

Type	Costs		B&C					BRKGA-QL			$\Delta\%\overline{UB}$	$\Delta\%\overline{UB}$
	Fixed	Variable	#Opt.	$\overline{UB}$	$\overline{LB}$	$\overline{Gap\ (\%)}$	$\overline{T\ (s)}$	$\overline{UB}$Best	$\overline{UB}$Avg.	$\overline{T(s)}$ Best	Best	Avg.
Original	low	low	21	664.28	664.28	0.00	133.75	664.48	665.36	29.23	0.03	0.16
		high	21	669.69	669.69	0.00	138.71	670.07	670.63	26.05	0.06	0.14
	high	low	21	667.51	667.51	0.00	179.13	667.82	668.40	27.02	0.05	0.13
		high	21	670.21	670.21	0.00	160.08	670.49	671.05	23.65	0.04	0.13
$2/3 \times Q$	low	low	21	671.69	671.69	0.00	41.36	672.04	672.76	32.82	0.05	0.16
		high	21	677.66	677.66	0.00	42.70	677.92	678.53	28.78	0.04	0.13
	high	low	21	675.09	675.09	0.00	46.18	675.35	675.99	30.46	0.04	0.13
		high	21	678.21	678.21	0.00	44.66	678.33	678.93	25.47	0.02	0.11
$m-1$	low	low	21	676.43	676.43	0.00	164.02	676.56	677.27	30.42	0.02	0.12
		high	21	697.31	697.31	0.00	252.51	697.36	697.94	23.99	0.01	0.09
	high	low	21	681.42	681.42	0.00	181.58	681.47	682.29	27.74	0.01	0.13
		high	21	699.41	699.41	0.00	173.53	699.54	700.10	23.23	0.02	0.10
Avg.			**21.00**	**677.41**	**677.41**	**0.00**	**129.85**	**677.62**	**678.27**	**27.41**	**0.03**	**0.13**

Figure 3 presents a boxplot with the objective function values found by B&C and BRKGA-QL methods. We consider the best and average values found by the metaheuristic. We can observe that the boxplots are very similar. The median values (second quartile) are 719.9 for the B&C and 720.1 for the BRKGA-QL (best and average upper bounds). The first and third quartile are also equal in the boxplots of the B&C and BRKGA-QL (best), respectively, 506.7 and 919.8. Finally, we performed a statistical analysis to compare B&C and BRKGA-QL considering all instances of this test. We apply the Wilcoxon signed-rank test [33]

Table 7: B&C and BRKGA-QL results for the DARP-PFCC on instances of set B

Type	Costs		B&C					BRKGA-QL			$\Delta\%\overline{UB}$	$\Delta\%\overline{UB}$
	Fixed	Variable	#Opt.	$\overline{UB}$	$\overline{LB}$	$\overline{Gap\ (\%)}$	$\overline{T\ (s)}$	$\overline{UB}$Best	$\overline{UB}$Avg.	$\overline{T(s)}$ Best	Best	Avg.
Original	low	low	20	714.12	713.21	0.08	725.05	714.33	714.73	28.95	0.03	0.21
		high	20	724.50	723.37	0.10	832.97	724.55	724.86	23.05	0.01	0.21
	high	low	20	718.32	717.16	0.10	872.13	718.37	718.68	27.30	0.01	0.21
		high	20	725.57	724.06	0.13	709.00	725.64	725.91	24.27	0.01	0.26
$2/3 \times Q$	low	low	21	896.18	896.18	0.00	38.97	896.18	896.23	14.82	0.00	0.01
		high	21	1181.05	1181.05	0.00	37.80	1181.05	1181.05	7.14	0.00	0.00
	high	low	21	925.48	925.48	0.00	38.16	925.48	925.53	12.37	0.00	0.01
		high	21	1208.35	1208.35	0.00	38.66	1208.35	1208.35	4.90	0.00	0.00
$m-1$	low	low	20	728.97	727.56	0.12	669.00	729.00	729.51	34.14	0.00	0.27
		high	20	760.13	758.89	0.10	665.24	760.17	760.35	23.51	0.00	0.19
	high	low	20	736.21	734.73	0.12	720.67	736.13	736.64	29.97	-0.01	0.26
		high	20	763.44	762.06	0.12	607.62	763.46	763.65	21.64	0.00	0.21
Avg.			**20.33**	**840.19**	**839.34**	**0.07**	**496.27**	**840.23**	**840.46**	**21.01**	**0.00**	**0.15**

to compare B&C with BRKGA-QL (best and average upper bounds). This test shows that the B&C ranks, were not statistically significantly different than the BRKGA-QL ranks, (p-value = 0.9371) when compared with best upper bounds, nor when compared with average upper bounds (p-value = 0.8406).

6.3. Case study with the DARP-PFCC

We conducted a case study based on a data set collected from a company that operates a dial-a-ride service in Québec, Canada. We present instances with up to 713 requests for the DARP-PFCC. For confidentiality reasons, we do not disclose the real data, but for each database request, we identify the postal code of the pickup and delivery nodes. We then generate an instance by selecting node locations within the same postal codes. For each of the 10 generated instances, Table 8 lists the instance name, the number of requests (n), number of homogeneous vehicles ($m = 3$ and $Q = 5$), the demand ($q_i = 1$), the service time ($d_i = 2$), the maximum ride-time duration ($L = 90$), and the maximum route duration ($T = \{480, 600, 720\}$). Time windows were generated according to the definition of Section 6.1. We use different time windows according to the number of users. Users $1, ..., n/2$ are outbound requests and $n/2 + 1, ..., n$ are inbound requests. Distances and travel times are obtained using the road network. The routing costs of the private fleet are generated by multiplying the distances by 0.4. Regarding the common carrier, we consider two different fixed costs $f_i \in \{5, 10\}$, and the variable cost is calculated by multiplying the distances by 2.

The results for the case study are presented in Tables 9 and 10. The B&C results for these instances are substantially worse than the results obtained by BRKGA-QL since B&C could not prove optimality for any instance tested, and the average gap was 96%. The large number of requests and the less tight time

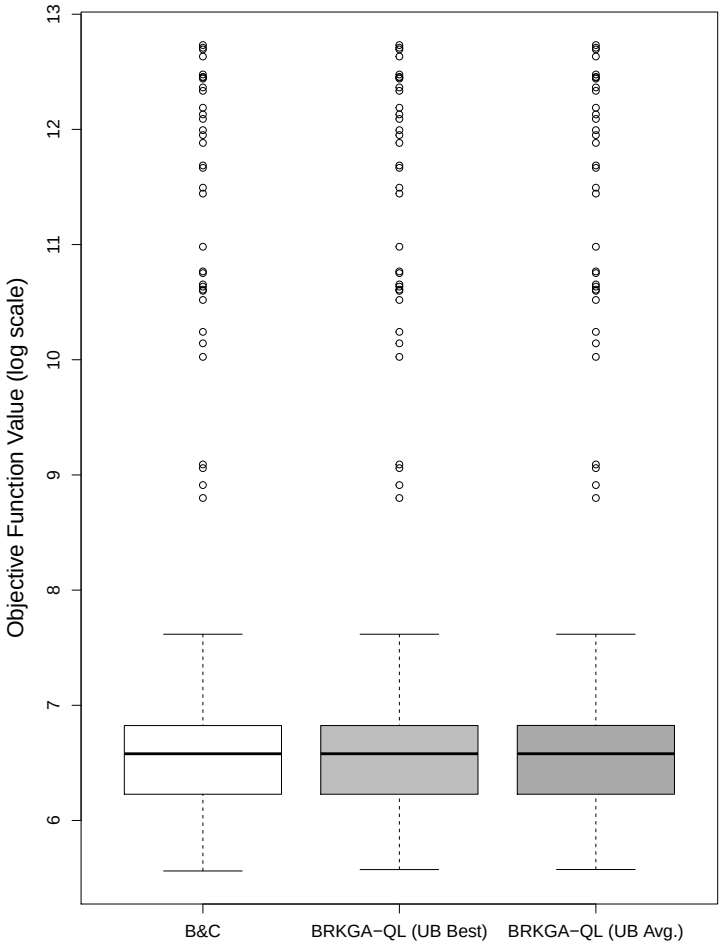

Figure 3: Boxplot with solution costs found by B&C and BRKGA-QL (best and average upper bounds).

Table 8: Real data parameters

Instance	n	m	q_i	d_i	L	T	e_{n+i}	l_{n+i}	e_i	l_i
a100	100	3	1	2	90	480	$l_{n+i} - 30$	$[60, T]$	$[0, T-60]$	$e_i + 30$
a150	150	3	1	2	90	480	$l_{n+i} - 30$	$[60, T]$	$[0, T-60]$	$e_i + 30$
a200	200	3	1	2	90	480	$l_{n+i} - 30$	$[60, T]$	$[0, T-60]$	$e_i + 30$
a250	250	3	1	2	90	480	$l_{n+i} - 30$	$[60, T]$	$[0, T-60]$	$e_i + 30$
a300	300	3	1	2	90	600	$l_{n+i} - 45$	$[60, T]$	$[0, T-60]$	$e_i + 45$
a350	350	3	1	2	90	600	$l_{n+i} - 45$	$[60, T]$	$[0, T-60]$	$e_i + 45$
a400	400	3	1	2	90	600	$l_{n+i} - 45$	$[60, T]$	$[0, T-60]$	$e_i + 45$
a500	500	3	1	2	90	720	$l_{n+i} - 60$	$[60, T]$	$[0, T-60]$	$e_i + 60$
a600	600	3	1	2	90	720	$l_{n+i} - 60$	$[60, T]$	$[0, T-60]$	$e_i + 60$
a713	713	3	1	2	90	720	$l_{n+i} - 60$	$[60, T]$	$[0, T-60]$	$e_i + 60$

windows affect B&C's performance. Therefore, in this section, we only present the results obtained by the BRKGA-QL.

Table 9 presents the operational costs for the private fleet (column PF cost) and common carrier (column CC cost), and the number of users (column #Req) served by each one, separated by fixed costs f_i. We emphasize that the BRKGA-QL stop criteria was set to $10 \times n$ seconds. It is possible to observe that increasing the number of requests significantly increases the common carrier operational cost; that is, the private fleet cannot satisfy all requests, and we have many more users being served by the outsourced vehicles. Despite this, we noticed that when more requests need to be served, the number of users served by the private fleet is larger. The instance with 713 requests has twice as many points served by the private fleet as 100 requests (84 and 182). We highlight that the private fleet size and their capacity are the same for all instances, and this behavior is related to the greater number of options that can be selected to be served by the private fleet.

Table 10 presents an operational analysis of the routing service and also contributes to this conclusion. We break down the operating time into *i*) traveling with passengers (TP), *ii*) traveling empty (TE), *iii*) service duration (SD), and *iv*) waiting (W). We can observe that vehicles travel most of the time transporting passengers (on average 62% of the time), and the travel time without passengers decreases as there are more requests to be met. With 100 users, we have vehicles on average 5.49% ($f_i = 5$) and 6.15% ($f_i = 10$) of the time without passengers, while with 713 users, we have an average of 2.79% ($f_i = 5$) and 1.08% ($f_i = 10$) of the time with empty displacements. The same behavior can be observed in the vehicle waiting times. Thus, these results show that the vehicle fleet is not sufficient to satisfy all users but that a larger number of requests allows for more efficient routing of the private fleet.

Table 9: Private fleet (PF) and common carrier (CC) operation costs

Instance	$f_i = 5$				$f_i = 10$			
	PF Cost	**#Req**	**CC Cost**	**#Req**	**PF Cost**	**#Req**	**CC Cost**	**#Req**
a100	273.16	84	416.34	16	270.45	83	508.66	17
a150	279.73	94	1259.56	56	290.24	94	1530.74	56
a200	274.35	103	2378.66	97	289.49	99	2866.06	101
a250	305.10	102	3661.73	148	315.47	101	4413.40	149
a300	326.92	151	4011.31	149	336.47	141	4870.80	159
a350	336.35	147	5583.73	203	326.30	149	6746.70	201
a400	341.37	148	6683.57	252	350.56	147	8021.70	253
a500	428.69	170	8746.82	330	407.15	172	10384.00	328
a600	385.94	189	10898.51	411	421.13	191	13030.78	409
a713	411.00	182	13830.87	531	373.44	195	16478.83	518
Avg.	**336.26**	**137.00**	**5747.11**	**219.30**	**338.07**	**137.20**	**6885.17**	**219.10**

Finally, in Figure 4 we compare the number of requests served by the private fleet concerning its idleness.

Table 10: Private fleet operating time break down

Instance	$f_i = 5$				$f_i = 10$			
	$\Delta\%TP$	$\Delta\%TE$	$\Delta\%SD$	$\Delta\%W$	$\Delta\%TP$	$\Delta\%TE$	$\Delta\%SD$	$\Delta\%W$
a100	62.13	5.49	27.05	5.33	61.20	6.15	26.83	5.82
a150	62.95	2.98	28.62	5.45	64.51	4.29	28.86	2.34
a200	62.62	3.27	31.17	2.93	63.77	2.55	29.07	4.60
a250	64.55	3.26	29.43	2.77	66.03	2.82	29.33	1.82
a300	62.09	2.13	35.05	0.73	63.23	1.68	33.50	1.60
a350	59.73	4.32	34.73	1.22	59.22	4.05	35.21	1.52
a400	62.93	2.02	34.16	0.89	62.90	2.51	33.68	0.91
a500	64.37	1.31	32.26	2.07	62.27	2.69	33.70	1.35
a600	59.87	1.93	36.22	1.99	61.33	2.22	35.76	0.69
a713	61.68	2.79	34.68	0.85	59.93	1.08	37.68	1.31
Avg.	**62.29**	**2.95**	**32.34**	**2.42**	**62.44**	**3.00**	**32.36**	**2.20**

TP: traveling with passengers; TE: traveling empty; SD: service duration; W: waiting.

We consider as idle time the total of the traveling empty (TE) and waiting (W) times. This idle time does not generate revenue for the transportation provider and can be reduced through optimization. The figure shows that as the number of requests to be served increases, the combination between private fleet and common carrier allows the transportation provider to select more profitable users, increasing the number of requests served by the private fleet, and the efficient use of these assets by reducing the total idle time from 11.97% to 2.39%.

7. Conclusions and future works

In this paper, we have introduced, modeled, and solved the dial-a-ride problem with private fleet and common carrier (DARP-PFCC) that makes it possible to outsource part of the demand unmet by the transportation provider to mobility-on-demand services and taxis. We have designed a state-of-the-art branch-and-cut algorithm to solve the DARP-PFCC exactly, and we have also developed a near parameter-free parallel BRKGA-QL to handle large instances. Both algorithms are flexible enough to solve the classical DARP instances, obtaining excellent results. We have adapted these instances and generated a dataset of 504 instances for the DARP-PFCC. Our results show that the cost increases 1.15% on average when we reduce 1/3 of vehicle capacity, while reducing one vehicle from the private fleet the impact is about 3.06% on average. For a case study with up to 713 requests, the results point to the benefits of combining the private fleet and common carriers in dial-a-ride problems. Remarkably when increasing the total number of user requests, the idleness of the private fleet substantially decreases by reducing the empty traveling from 5.8% when we have 100 requests to 1.9% when the number of requests is 713. Similarly, the waiting time decreases from 5.58% to 1.08% regarding the same range of requests. Our instances and source codes (problem-independent

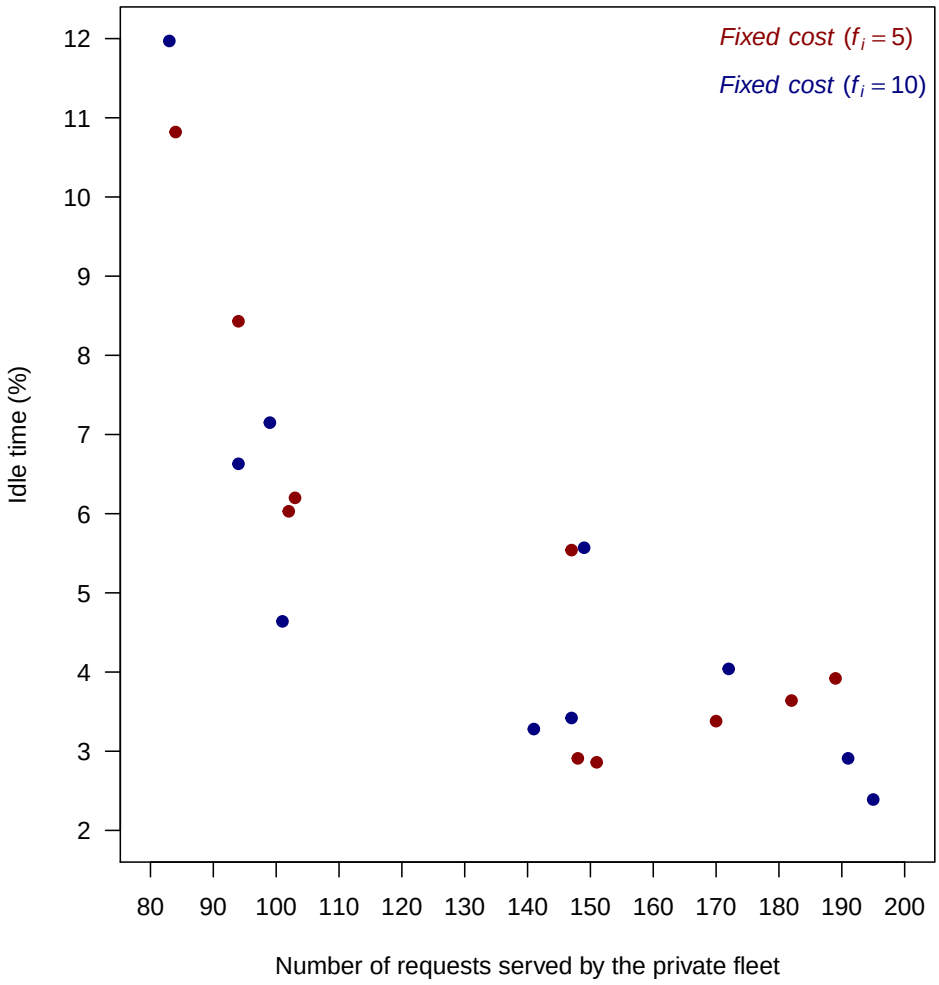

Figure 4: Idle time of the private fleet

BRKGA-QL) are made publicly available for the research community.

From the methodological perspective, we suggest that future studies hybridize B&C with BRKGA-QL in a shared parallelized framework. Thus, the efficiency of BRKGA-QL in obtaining good solutions in a shorter computational time may be advantageous to B&C, which could increase the number of optimal solutions proved. Regarding the scope of the problem, a promising direction could involve the integration of the private fleet, common carrier, and public transportation. This new context would reach a broader profile of users, including those who can use buses and trains. Synchronization constraints need to be incorporated as public transport operates within a fixed timetable. On the other hand, more options for serving users would reduce operating costs for transportation providers.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36(2):69–79, 2000.

[2] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, 106(2–3):546–557, 1998.

[3] K. Braekers, A. Caris, and G. K. Janssens. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, 67: 166–186, 2014.

[4] D. Brevet, C. Duhamel, M. Iori, and P. Lacomme. A dial-a-ride problem using private vehicles and alternative nodes. *Journal on Vehicle Routing Algorithms*, 2(1-4):89–107, 2019.

[5] A. A. Chaves and L. H. N. Lorena. An adaptive and near parameter-free BRKGA using Q-learning method. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 2331–2338, 2021.

[6] A. A. Chaves, L. A. N. Lorena, E. L. F. Senne, and M. G. C. Resende. Hybrid method with CS and BRKGA applied to the minimization of tool switches problem. *Computers & Operations Research*, 67: 174–183, 2016.

[7] A. A. Chaves, J. F. Gonçalves, and L. H. N. Lorena. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Computers & Industrial Engineering*, 124:331–346, 2017.

[8] J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3): 573–586, 2006.

[9] J.-F. Cordeau and G. Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *4OR*, 1(2):89–101, 2003.

[10] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

[11] S. Dabia, D. Lai, and D. Vigo. An exact algorithm for a rich vehicle routing problem with private fleet and common carrier. *Transportation Science*, 53(4):986–1000, 2019.

[12] M. Diana and M. M. Dessouky. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, 38(6):539–557, 2004.

[13] Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22, 2020.

[14] J. Euchi. The vehicle routing problem with private fleet and multiple common carriers: Solution with hybrid metaheuristic algorithm. *Vehicular Communications*, 9:97–108, 2017.

[15] J. F. Gonçalves and M. G. C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[16] J. F. Gonçalves and M. G. C. Resende. A parallel multi−population biased random-key genetic algorithm for a container loading problem. *Computers & Operations Research*, 39(2):179–190, 2012.

[17] J. F. Gonçalves and M. G. C. Resende. A biased random-key genetic algorithm for the unequal area facility layout problem. *European Journal of Operational Research*, 246(1):86–107, 2015.

[18] J. F. Gonçalves, M. G. C. Resende, and J. J. M. Mendes. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics*, 17(5):467–486, 2011.

[19] S. C. Ho, W. Y. Szeto, Y.-H. Kuo, J. M. Y. Leung, M. Petering, and T. W. H. Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421, 2018.

[20] J.-J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257, 1986.

[21] A. F. N. Kummer, L. S. Buriol, and O. C. B. Araújo. A biased random key genetic algorithm applied to the VRPTW with skill requirements and synchronization constraints. *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, (5):717–724, 2020.

[22] I. Malheiros, R. Ramalho, B. Passeti, T. Bulhões, and A. Subramanian. A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem. *Computers & Operations Research*, 129:105196, 2021.

[23] M. A. Masmoudi, M. Hosny, K. Braekers, and A. Dammak. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, 96:60–80, 2016.

[24] M. A. Masmoudi, M. Hosny, E. Demir, K. N. Genikomsakis, and N. Cheikhrouhou. The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation Research Part E: Logistics and Transportation Review*, 118:392–420, 2018.

[25] Y. Molenbruch, K. Braekers, P. Hirsch, and M. Oberscheider. Analyzing the benefits of an integrated mobility system using a matheuristic routing algorithm. *European Journal of Operational Research*, 290 (1):81–98, 2021.

[26] J. Narayan, O. Cats, N. van Oort, and S. Hoogendoorn. Integrated route choice and assignment model for fixed and flexible public transport systems. *Transportation Research Part C: Emerging Technologies*, 115:102631, 2020.

[27] J. Paquette, J.-F. Cordeau, and G. Laporte. Quality of service in dial-a-ride operations. *Computers & Industrial Engineering*, 56(4):1721–1734, 2009.

[28] S. N. Parragh, K. F. Doerner, and R. F. Hartl. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 37(6):1129–1138, 2010.

[29] P.H.V. Penna, A. Subramanian, and L.S. Ochi. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19:201–232, 2013.

[30] M. Posada, H. Andersson, and C. H. Häll. The integrated dial-a-ride problem with timetabled fixed route service. *Public Transport*, 9(1-2):217–241, 2017.

[31] Y. Qu and J. F. Bard. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, 49(2):254–270, 2015.

[32] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[33] D. Rey and M. Neuhauser. Wilcoxon-signed-rank test. In M. Lovric, editor, *International Encyclopedia of Statistical Science*, pages 1658–1659. Springer Berlin Heidelberg, 2014.

[34] S. Ropke and J.-F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(4):267–286, 2009.

[35] S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272, 2007.

[36] K. S. Ruland and E. Y. Rodin. The pickup and delivery problem: faces and branch-and-cut algorithm. *Computers & Mathematics with Applications*, 33(12):1–13, 1997.

[37] M. W. P. Savelsbergh. The vehicle routing problem with time windows: minimizing route duration. *INFORMS Journal on Computing*, 4(2):146–154, 1992.

[38] C. M. Schenekemberg, C. T. Scarpin, J. E. Pécora Jr., T. A. Guimarães, and L. C. Coelho. The two-echelon inventory-routing problem with fleet management. *Computers & Operations Research*, 121: 104944, 2020.

[39] W. M. Spears and K. A. De Jong. On the virtues of parameterized uniform crossover. *Proc. of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

[40] K. Steiner and S. Irnich. Strategic planning for integrated mobility-on-demand and urban public bus networks. *Transportation Science*, 54(6):1616–1639, 2020.

[41] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, 1989.