Annals of Operations Research (2025) 351:809–847
https://doi.org/10.1007/s10479-024-06136-9

ORIGINAL RESEARCH

Hybrid metaheuristic for the dial-a-ride problem with private fleet and common carrier integrated with public transportation

Cleder M. Schenekemberg[1,2] · Antonio A. Chaves[1] · Thiago A. Guimarães[3] · Leandro C. Coelho[4]

Received: 13 June 2023 / Accepted: 20 June 2024 / Published online: 3 July 2024

Abstract
Dial-a-ride operations consist of door-to-door transportation systems designed for users with specific needs. Governments and companies offer such services, and due to the flexibility and service level required by the users, it is considerably more costly than public transportation, besides emitting higher levels of CO_2. Hence, it is crucial to analyze alternatives to improve operational costs and efficiency without compromising the quality of the service. This paper introduces a variant for the dial-a-ride problem with private fleets and common carriers (DARP-PFCC) integrated with public transportation. Requests can be served by the private fleet, the common carrier, or by integrating them into the public transportation system. In this case, users are collected at the pickup locations and taken to bus stops. After the bus trip, other vehicles serve them from the bus stops to their final destination. Bus schedules must be considered when deciding on the best integration trip. As a methodology, we solve this extension of the DARP-PFCC with a metaheuristic and machine learning hybrid method by combining a biased random key genetic algorithm with the Q-Learning and local search heuristics (BRKGA-QL). This paper also introduces some improvements to this method, particularly with respect to population quality and diversity, thanks to a new mutation method in the classical crossover operator and deterministic rules for the learning process. Computational experiments on a new benchmark data set with realistic data from Québec City show that our BRKGA-QL outperforms its previous version. In addition, we provide a qualitative analysis for the DARP-PFCC, showing that the middle mile integration with public trans-

✉ Cleder M. Schenekemberg
cledercms@hotmail.com

Antonio A. Chaves
antonio.chaves@unifesp.br

Thiago A. Guimarães
thiago.guimaraes@ifpr.edu.br

Leandro C. Coelho
leandro.coelho@fsa.ulaval.ca

1 Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil

2 Aeronautics Institute of Technology (ITA), São José dos Campos, Brazil

3 Federal Institute of Science and Technology of Paraná (IFPR), Curitiba, Brazil

4 Canada Research Chair in Integrated Logistics, CIRRELT, Université Laval, Québec, Canada

portation can save up to 20% in operating costs, besides reducing the traveled distances of private vehicles and common carriers.

Keywords Dial-a-ride · Inter-modal mobility · Common carrier · Bus · Genetic algorithm

1 Introduction

Dial-a-ride problems (DARP) aim to offer responsive door-to-door transportation services for users with specific needs. From an operational side, a transportation provider serves the requests using its private fleet, subject to constraints such as vehicle capacity, pairing, and precedence. Regarding the service level, constraints like time window, duration, and ride time are also considered (Cordeau & Laporte, 2003a). Historically, the DARP has attracted great attention in the literature, and several algorithms have been proposed for its standard version and variants (Braekers et al., 2014; Brevet et al., 2019; Cordeau, 2006; Cordeau & Laporte, 2003b; Diana & Dessouky, 2004; Masmoudi et al., 2016, 2018; Qu & Bard, 2015; Ropke et al., 2007; Ropke & Cordeau, 2009). The survey of Ho et al. (2018) provides a review and recent developments.

Due to the flexibility required, a dial-a-ride operation is high-costly and the vehicles usually travel with low occupancy. Besides the low operational efficiency, the DARP is also environmentally inefficient as a means of public transport, given the associated high levels of direct and indirect CO_2 emissions (Shiraki et al., 2020), since it also impacts traffic congestion (Beojone & Geroliminis, 2021). On the other hand, dial-a-ride services may help connect zones that do not have flow demand and are disconnected from the public transportation system. Therefore, an alternative to improve operational and environmental efficiency is the integration of private fleets with public transportation, where some legs of the journey are performed by fixed routes of trains and buses (Posada et al., 2017). However, an integrated, coordinated, multi-modal public transportation system substantially increases operational complexity. It must offer wide coverage, real-time response to demand, quality service, and energy-efficient operations (Kuo et al., 2023).

The integration of DARP and public transportation, despite promoting better operational and environmental efficiency, could cause a reduction in the quality of service, as the user has to carry out multiple hop-on hop-offs, in addition to being subject to possible delays in buses and trains. Therefore, another type of integration is proposed by Schenekemberg et al. (2022), where the demand can be served either by the private fleet, mobile-on-demand transportation, or their combination. The authors introduce and model the DARP with private fleet and common carrier (DARP-PFCC) and implement a branch-and-cut (B&C) algorithm to solve it. A hybrid biased random key genetic algorithm (BRKGA) is also designed, where a reinforcement learning method optimizes the algorithm parameters, called BRKGA-QL. This method successfully solved both the DARP classical instances and DARP-PFCC instances.

To incorporate better operational and environmental performance from public transportation and promote a better quality of service, this paper extends the work of Schenekemberg et al. (2022) and integrates common carriers and public transportation into the DARP-PFCC. Requests can be served by the private fleet, the common carrier, or by integrating them into the public transportation system. In this last case, the middle mile is performed by a bus while the first and last legs of the trip are performed by a mobile-on-demand vehicle, such as a taxi, Uber, or equivalent. Bus schedules must be taken into account when deciding on the best integration trip. To solve this novel problem variant, we introduce BRKGA-QL v2.0, with

improvements in population management and learning process. Computational experiments performed on a new benchmark data set with realistic characteristics of Québec City show that BRKGA-QL v2.0 outperforms its previous version. The contributions of this paper are to:

- Combine a private fleet, common carriers, and public transportation in a DARP-PFCC variant. Such an approach has several benefits. According to their profile, users can be served more quickly by common carriers, while providers can reduce their costs by transferring part of the demand to the buses. The quality must also be considered by factors such as walking distance, unserved demand, and potential new demand. Finally, global benefits are generated for society since fewer vehicles travel long distances with low occupancy, reducing urban traffic and enabling lower CO_2 emissions;
- Present a data pre-processing technique to define the best bus line, timetable, and stops for integrating each request. Since this task is extremely time-consuming, this procedure enables evaluating a subset of promising bus stops close to the request nodes (pickup or delivery), avoiding computing all possibilities from the public transportation database;
- Design a BRKGA-QL v2.0 algorithm to solve the DARP-PFCC variant, which is more efficient than its previous version, benchmarking it on the instances without public transportation. We make the source code (problem-independent) of our algorithm available to the research community;
- Generate a new instance set for the DARP-PFCC variant, with all instances and results available online;
- Apply our BRKGA-QL v2.0 to solve real-case scenarios of the DARP-PFCC variant, showing that the middle mile integration with public transportation improves operational performance by reducing the traveled distances of private vehicles and common carriers.

The remainder of the paper is organized as follows. Section 2 provides a literature revision, whereas Sect. 3 formally describes the DARP-PFCC variant and details the synchronization procedure with public transportation. The BRKGA-QL v2.0 is presented in Sect. 4 while encoding, decoding, and local search strategies are described in Sect. 5. In Sect. 6, we present the instances generator and the results of extensive computational experiments. Our conclusions and directions for future works follow in Sect. 7.

2 Literature review

Due to its complexity, many metaheuristics designed for the classical DARP are found in the literature. Regarding state-of-the-art algorithms, we refer to deterministic annealing algorithm (DA) (Braekers et al., 2014), evolutionary local search (ELS) (Chassaing et al., 2016), genetic algorithm with local search (GA+LS) (Masmoudi et al., 2017), large neighborhood search (LNS) (Molenbruch et al., 2017), hybrid iterated local search algorithm with set partitioning (ILS-SP) (Malheiros et al., 2021), and the adaptive large neighborhood search algorithm (ALNS) (Gschwind & Drexl, 2019). These methods are not statistically different from BRKGA-QL in the DARP context (Schenekemberg et al., 2022).

Regarding integrating dial-a-ride and public transportation, we highlight next the relevant literature. These works explore strategies to deal with unstable demand in remote areas, the first mile/last mile problem, and optimizing transportation networks in coastal, rural, and suburban regions.

Chu et al. (2022) point out that designing efficient fixed bus routes in remote areas is difficult because demand is unstable and scattered. The authors propose integrated planning

of a fixed bus route with dial-a-ride transportation to improve the service level. A two-stage stochastic programming is introduced, where a set partitioning optimizes the route and timetable of a bus line, while a DARP is formulated in the second stage. A hybrid algorithm combining depth-search and heuristic procedures is developed to solve the problem. Results based on a case study show that integrating bus and dial-a-ride operations considerably reduces unserved demands and improves the service level.

Addressing the first mile/last mile problem, Kumar and Khani (2021) integrate peer-to-peer ridesharing with a schedule-based transit system. The first mile consists of a ride to a transit station, where the rider can travel by public transportation to their final destination in the last mile. A matching algorithm is designed to find an optimal match between riders and drivers and synchronize the vehicle trip with the rest of the itinerary. To reduce the network search, the study identifies the space-time area a traveler can reach. A rolling horizon-based algorithm is also presented, and the computational experiments involving real data show a significant reduction of vehicle-hours in the traffic system.

Aslaksen et al. (2021) investigate the integration of dial-a-ride and ferry transportation systems in coastal cities. The study considers on-demand services with autonomous small ferry units, and the authors propose a request assignment ferry routing procedure in a simulation framework. Computational experiments were performed for a case study in Kiel, Germany to assess the relationship between fleet sizing and demand satisfaction under different demand scenarios. In Posada and Häll (2020), the integration between a special transportation service with a fixed route public transportation for rural areas is considered. An adaptive large neighborhood search (ALNS) is proposed to minimize the distance traveled by the demand-responsive vehicles, reaching around 16% of distance reduction.

Azadeh et al. (2022) focus on the integration of demand-responsive transportation and fixed line and scheduled transportation networks in suburban areas, aiming to enhance accessibility to public transport services from stop-to-stop rather than door-to-door. The study investigates the temporal and spatial feasibility of this integration, considering passengers' behavior. The proposed mathematical model determines an optimal allocation of stations for each network, while customer satisfaction is evaluated based on the number of served requests. The model only solves small-size instances. Then, an ALNS algorithm is proposed to address larger instances efficiently. The results demonstrate that the integrated mobility network significantly improves service levels, primarily during evening hours, weekends, and public holidays.

As pointed out, the relevant literature has focused on a two-level integration, specifically combining private fleet operations with public transportation in the context of the DARP. Our contribution expands it by introducing a novel three-level integration approach incorporating private fleet, common carrier, and public transportation into the same framework. This arrangement can explore a wider paradigm, taking advantage of all modals in the DARP.

3 Problem definition

The DARP-PFCC integrated with public transportation is defined as follows. Let n be the number of requests from a set of pickup nodes $\mathcal{P} = \{1, \ldots, n\}$ to a set of delivery nodes $\mathcal{D} = \{n+1, \ldots, 2n\}$. Each request i represents a pair of nodes $(i, n+i)$, with $i \in \mathcal{P}$ and $n+i \in \mathcal{D}$, and its demand q_i can be satisfied by either a private vehicle, a common carrier vehicle, or by integrating common carriers and buses. A limited fleet of m private vehicles of capacity C is available. All private vehicles depart from the depot 0, visit a set of nodes

(pickup or delivery), and return to the depot $2n + 1$, without violating the maximum route duration T. In this transportation option, users share the same vehicle, which travels among nodes according to a scheduling plan that aims to efficiently assign vehicles and determine their routes to minimize operational costs. Regarding outsourced vehicles, we assume an unlimited fleet from a common carrier, where each vehicle has sufficient capacity to satisfy any request. Finally, some users can be served by integrating common carriers with public transportation. In the first mile, users from node i are picked up by a common carrier vehicle and transported to a departure bus stop k. In the middle mile, these users travel from k to an arrival bus stop k'. Finally, the last mile from k' to the delivery node $n+i$ is made by another outsourced vehicle. Both bus stops k and k' must be visited by the same bus line l.

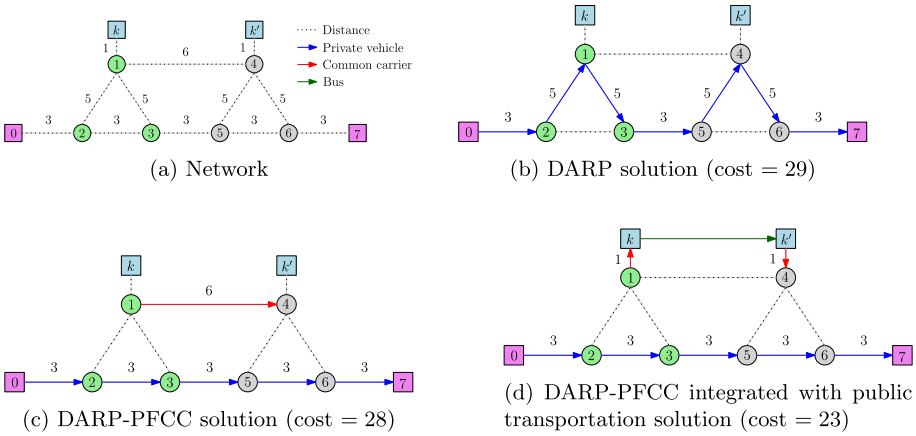
This proposed problem originates from a real-world application observed within a Québec-based dial-a-ride service provider. As reported by the company, some users exhibit low levels of restrictions, enabling them to use public transportation. Consequently, we introduced a compatibility matrix between users and public transportation, examining various integration levels (ranging from 0% to 100%).

We consider the following transportation cost scheme. For private vehicles, a routing cost $c_{i,j}$ is multiplied by the distance $d_{i,j}$ traveled along arc (i, j) connecting nodes i and j, where $i, j \in \mathcal{P} \cup \mathcal{D} \cup \{0, 2n + 1\}$. For the common carrier vehicles, serving a request i incurs a fixed cost f and a variable cost v, where the total cost is $f + v \cdot d_{i,n+i}$. Lastly, synchronizing a request between common carriers and public transportation splits the costs into three parts. For the first mile, we have an outsourced vehicle traveling from i to bus stop k with cost $f + v \cdot d_{i,k}$; for the middle mile, a ticket cost p_l for bus line l is charged; and for the last mile, another extra vehicle is used from bus stop k' to the delivery node $n + i$, incurring a cost $f + v \cdot d_{k',n+i}$. These costs are calculated based on the case of Québec City. In this case, all requests are for single users. The costs are, therefore, independent of the demand.

To define a feasible solution for the DARP-PFCC integrated with public transportation, we also assume all operational constraints of the DARP, such as pairing, precedence, capacity, time windows, route connectivity, and ride time. A solution to this DARP-PFCC variant minimizes the total costs while determining whether users are served by the private fleet, common carrier, or common carrier and public transportation. Schenekemberg et al. (2022) proposed a mathematical model for the DARP-PFCC that can be adapted to this variant using our algorithm for the integration between requests and bus lines presented below.

Figure 1 presents an example of the DARP variations. We consider three requests (with pickup nodes $\{1, 2, 3\}$ and their respective delivery nodes $\{4, 5, 6\}$), two depots (0 and 7), and two bus stops (k and k'). We also consider the bus cost as $p_l = 2$, and a fixed and a variable cost for the common carrier given by $f = 1$ and $v = 2$, respectively. Figure 1a shows the network and the distances $d_{i,j}$ of each arc (i, j). In Fig. 1b we have a solution for the classical DARP with a cost equal to 29. Figure 1c presents a solution to the DARP-PFCC with a cost equal to 28 $(15 + [1 + 2 \times 6])$, in which a common carrier satisfies request 1, while Fig. 1d presents a solution for the DARP-PFCC integrated with public transportation considering that the user of request 1 is picked up in node 1 and taken to bus stop k; at bus stop k' another vehicle picks the user up and travels to delivery node 4. The cost of this solution is equal to 23 $(15 + [1 + 2 \times 1] + 2 + [1 + 2 \times 1])$.

We consider all available bus lines, timetables, and bus stops in our DARP-PFCC. However, finding the best integration at runtime is a task that increases the computational time of the algorithm. Therefore, we propose pre-processing the data to define the best line/timetable/bus stops for each request, considering only bus stops within 20 min of the pickup/delivery node. This is done to balance the common carrier costs and bus ticket costs.



(a) Network (b) DARP solution (cost = 29)

(c) DARP-PFCC solution (cost = 28) (d) DARP-PFCC integrated with public transportation solution (cost = 23)

Fig. 1 Examples of DARP variants. The depots (0 and 7) are depicted by pink nodes. The pickup nodes of the requests are green, while grey nodes indicate their delivery nodes. The blue nodes represent bus stops. The colored arcs indicate the path of each vehicle type (the private vehicle is blue, the common carrier is red, and the bus is green)

This value needs to be calibrated with the data of the problem at hand. The cost of the integration options is calculated by equation (1), where we add the cost of common carriers in the first and last mile and the cost of the bus in the middle mile.

$$cost_{(i,n+i)} = [f + v \cdot d_{i,k}] + p_l + [f + v \cdot d_{k',n+i}]. \tag{1}$$

In the example of Fig. 2, we consider two bus lines (A and B) with two available timetables. For line A, we have a bus stop close to the pickup node i (point k') and a bus stop close to the delivery node $j = n + i$ (point k''). Line B has a bus stop k next to the pickup node and a bus stop k'' next to the delivery node. With the distances and travel times (including service time) presented in the figure (in parentheses) and considering $f = 3$, $v = 2$, and $p_l = 3$ for both lines, we have the following options for integration:

- Route $[i, k', k'', j]$, using line A from bus stop k' at timetable 615 and with a total travel time of 40 min. Upon arrival at the delivery node, the user has a waiting time before the time window opens. The cost of this integration is 31;
- Route $[i, k', k'', j]$, using line A from bus stop k' at timetable 625 and with a total travel time of 40 min but no waiting time at the delivery node. The cost of this integration is also 31;
- Route $[i, k, k'', j]$, using line B from bus stop k at timetable 610 and a total travel time of 45 min without waiting at the delivery node. The cost of this integration is 27;
- Route $[i, k, k'', j]$, using line B from bus stop k at timetable 635. However, this option is infeasible due to the arrival at node j after the end of the time window.

After enumerating the options for each request, we select the one with the lowest cost. The third option has the lowest cost in the example of Fig. 2. Therefore, we use line B, bus stops k and k'', and timetable 610 if it is economically advantageous to integrate this request with public transportation.

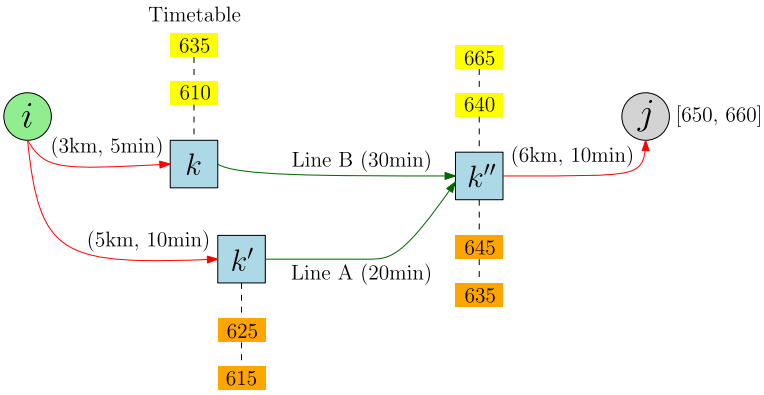


Fig. 2 Example of the integration between request nodes and bus lines. The pickup node i is represented in green, the delivery node $j = n + i$ in grey, and bus stops k, k', and k'' in blue. The colored arcs indicate the path of each vehicle type (common carrier in red, bus in green), and timetables are depicted in orange and yellow for lines A and B, respectively

4 Biased random key genetic algorithm with *Q*-Learning

Genetic Algorithms (GAs) (Holland, 1975) can efficiently solve combinatorial optimization problems. BRKGA (Gonçalves & Resende, 2011) is a recent GA variation that has successfully solved several routing problems (Schenekemberg et al., 2022; Silva et al., 2019; Andrade et al., 2019). BRKGA uses double elitism in the Parameterized Uniform Crossover (PUX) (Spears & Jong, 1991), selecting a parent from an elite set, and the offspring inherits genes with a higher probability from this parent.

BRKGA represents a solution by a random key vector with dimension n ($v \in [0, 1)^n$), and the decoder function maps these vectors to the solution space of the problem at hand. Thus, the evolutionary process is independent of the problem, and one can reuse the code to accelerate the development of new applications. However, a disadvantage of the BRKGA is the large number of parameters to be configured. It is a difficult task that requires tuning tools (e.g., iRace or ParamILS) to obtain good results (Andrade et al., 2021). Chaves et al. (2017) proposed an adaptive BRKGA with deterministic and self-adaptive rules to control the parameters during the evolutionary process, and Chaves & Lorena (2021) developed a BRKGA with the Q-Learning algorithm to learn the most appropriate parameter configuration during the evolutionary process. Q-Learning (Watkins, 1989) is a model-free reinforcement learning algorithm that updates the value function based on rewards. The agent uses some parameters to explore the environment and update the Q-Table.

In this paper, we propose a new version of the BRKGA with Q-Learning, called BRKGA-QL v2.0, with improvements to perform efficient parameter control and balance intensification and diversification during the learning and search process. BRKGA-QL has a set of states S (parameters) and a set of actions A that represents a parameter value. There is only one state for each BRKGA parameter, and the value $Q(s_i, a_j)$ represents the quality of using the value a_i in the parameter s_i. At each generation, we choose each parameter value from Q-Table using the ϵ-greedy policy. The agent aims to maximize its total reward, with the ϵ-greedy policy selecting the action with the highest Q in state s_i with a probability of $1 - \epsilon$, or any action otherwise.

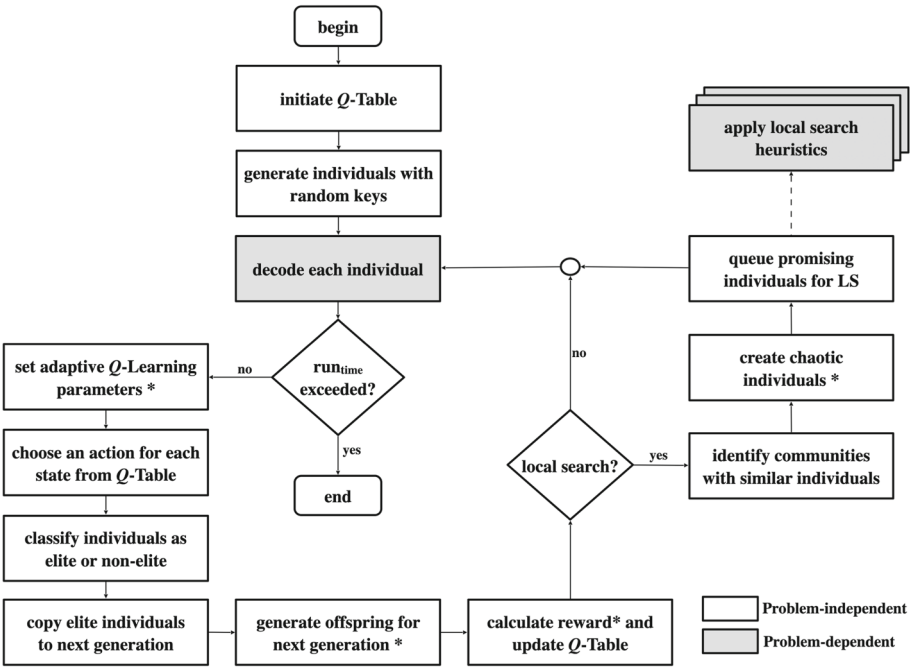


Fig. 3 View of the BRKGA-QL algorithm version 2.0. Components marked with an $*$ are new in this version. The mutant component was dropped and a mutation probability was added into the crossover component

Figure 3 presents the flowchart of BRKGA-QL v2.0. The decoder and the local search are the only problem-dependent components. All other components are generic and modular; thus, we can encapsulate them in a framework. The new elements of version 2.0 are represented by i) the adaptive parameter setting of the Q-Learning during the evolutionary process aiming to balance exploration and exploitation in the agent, ii) the proportional pruning in the elite set, iii) the mutation operator in the evolutionary process instead of mutants, iv) the reward function, and v) the chaotic individuals generated from the elite set. This set of new components allows for more diversification in the search process, which makes the method more efficient for escaping local optima. Also, with the new learning mechanism, we allow the agent to perform more experiments with the BRKGA parameters. This feature has been shown to be better than using static parameter values (Karafotias et al., 2014; Hutter et al., 2015). The BRKGA-QL v2.0 is compared against the previous version (Schenekemberg et al., 2022) in Sect. 6.

The first steps of the BRKGA-QL v2.0 are the same as the previous version: initiate Q-Table, generate individuals of the initial population, and decode each individual according to the last gene. However, the Q-Table has only four parameters (population size, p; elite set size, p_e; probability that an offspring inherits the gene from its elite parent, ρ_e; and mutation probability, p_m). The last parameter was introduced in this version to incorporate a probability of generating a random value for each gene in the crossover operator. Then, the possible actions for this parameter are $p_m = \{0.0, 0.01, 0.02, 0.03, 0.04, 0.05\}$.

Another difference between the versions is that the population can be reduced or expanded proportionally in the new version. The first case performs a proportional pruning in the elite

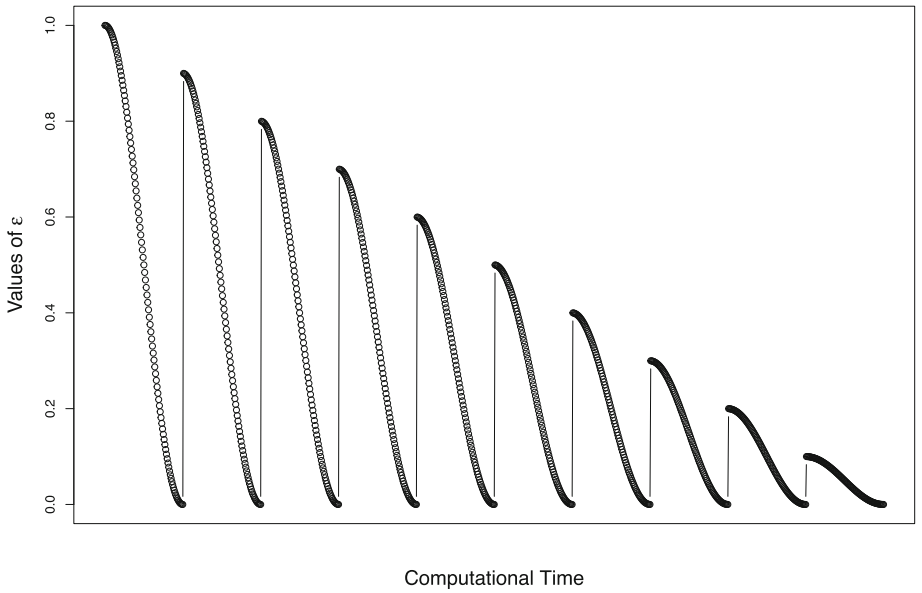


Fig. 4 Varying values of ϵ

and non-elite individuals. The second case generates new individuals with the crossover operator.

Unlike the first version of the BRKGA-QL, in version 2.0, the Q-Learning parameters are set during the evolutionary process by deterministic rules. We adopt a cosine annealing decay for the value of the ϵ-greedy policy (Loshchilov & Hutter, 2017). Figure 4 shows the schemes of ϵ values over the computational time with decaying and restarts according to equation (2):

$$\epsilon = \epsilon_{min} + \frac{1}{2} \cdot \left(\epsilon^i_{max} - \epsilon_{min}\right) \cdot \left(1 + cos\left(\frac{T^i_{cur}}{I}\pi\right)\right), \tag{2}$$

where $\epsilon_{min} = 0$ and $\epsilon^i_{max} \in \{1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$ are ranges for the values of ϵ, T^i_{cur} accounts for the run time since the last restart, I is equal to 10% of the maximum computational time defined by the user (run_{time}), and i is the index of the restart.

We promote exploration at the beginning of the search process when the agent needs more information about the environment. Over the generations, the agent gains knowledge about the most appropriate parameter settings and intensifies the search using this knowledge.

The learning factor (l_f) and the discount factor (d_f) are parameters of the Q-Learning used to update the Q-Table (equation (6)). The value of l_f is updated following equation (3). Thus, initially, we have a higher priority for newly gained information. The value of d_f is set to 0.8, looking for a higher, long-term reward (Samma et al., 2020).

$$l_f = 1 - \left(0.9 \cdot \frac{\sum_i T^i_{cur}}{run_{time}}\right). \tag{3}$$

In the first version, the Q-Learning parameters were also in the Q-Table and the agent had to learn to calibrate the BRKGA parameters and its own. Now, we establish a compromise between frequently applying good actions (exploitation), and eventually exploring actions

Table 1 Example of the Parameterized Uniform Crossover (PUX) with mutation

x	0.45	0.87	0.01	0.23	0.49
y	0.61	0.79		0.44	0.13
$a =$	**(0.25**	0.19	0.67	**0.05**	**0.89)**
$b =$	(0.63	**0.90**	0.76	0.93	0.08)
$c =$	(0.25	0.90	0.38	0.05	0.89)

that can lead to the discovery of new candidates for optimizing the expected total return (exploration).

The operators of the Q-Learning are the classic ones (Watkins, 1989): choose an action for each state using the ϵ-greedy policy, calculate the reward, and update the Q-Table. The ϵ values are initially high; the agent will explore the environment and randomly choose actions. This occurs logically since the agent does not know much about the environment. As the agent explores the environment, the ϵ rate decreases, and the agent exploits the environment better.

The evolutionary process of BRKGA-QL v2.0 differs from those of BRKGA and BRKGA-QL. The population is partitioned into two sets (elite and non-elite) based on the parameter p_e, and the elite partition is copied to the next population. The offspring is created with the Parameterized Uniform Crossover (PUX) operator using p_m and ρ_e. In previous versions, mutants (random solutions) are generated, but these are unlikely to have good fitness. Here, the PUX was modified to incorporate mutation probability. For each gene, first, it verifies the possibility of mutation (p_m). We generate a random value $x \in [0, 1)$, whether $x \leq p_m$ changes the gene to a random value. Otherwise, flip a biased coin $y \in [0, 1)$. If $y \leq \rho_e$, the elite parent passes the gene to the offspring; otherwise, it is taken from the non-elite parent. Table 1 shows an example of crossover/mutation with two parents (elite parent a and non-elite parent b), $p_m = 0.03$, and $\rho_e = 0.65$. This feature allows the method to have diversity and generate good individuals. The new individuals are evaluated by the decoder specified in random key $n + 1$, and the new population is sorted by fitness.

A reward R is calculated after each generation. We propose a new reward function:

$$R = \begin{cases} 1 + \Delta_f, & \text{if } f_b^t < f_b^{t-1} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where f_b^t is the best fitness in the population at generation t and Δ_f is defined by:

$$\Delta_f = \frac{1}{p}\left(\frac{f_b^{t-1}}{f_b^t} - 1\right) \times M. \tag{5}$$

The reward function combines the improvement of the current best fitness and binary rewards proposed by Hutter et al. (2015). This function is less scale insensitive than the binary reward, including the magnitude of the improvement and the population size. M is a big number; we set $M = 10^5$.

The function $Q(s_i, a_j)$ defines the value associated with the state-action pair (s_i, a_j) and represents how good the choice of this parameter value is when optimizing the expected total return. The expression for updating the value of Q is based on equation (6):

$$Q(s_i, a_j) = Q(s_i, a_j) + l_f \left[R + d_f \times Q_{max} - Q(s_i, a_j)\right], \tag{6}$$

where s_i is the current state, a_j is the action performed in state s_i, and Q_{max} is the target Q-value (the highest value in the Q-Table).

The BRKGA-QL v2.0 integrates the same local search module as Schenekemberg et al. (2022), enhancing its efficiency. When a new best solution is found, or no improvement occurs for a given number of generations, the Label Propagation (LP) method (Raghavan et al., 2007) groups elite solutions based on Pearson correlation. Simultaneously, the Random Variable Neighborhood Search (RVND) approach (Penna et al., 2013) is employed in parallel using the best solution of each group as an initial solution.

We also increase the diversity of the elite set by generating chaotic individuals (Li & Pei, 2019) from similar individuals. The chaotic individuals are generated by the PUX operator using mutants (random solutions) and individuals in the elite set that have not passed through the local search module. The chaotic individuals replace these similar individuals in the population.

The BRKGA-QL v2.0 also uses the run time (run_{time}) as a stopping criterion. However, run_{time} is considered a parameter and needs to be tuned based on the available computational resources and the complexity of the decoders and local search heuristics.

5 BRKGA-QL for the DARP-PFCC integrated with public transportation

This section shows how solutions are encoded to a vector of random keys and decoded from a vector of random keys into a DARP-PFCC solution integrated with public transportation. We also present some local search procedures to polish promising solutions.

5.1 Encoding a solution to a vector of random keys

A DARP-PFCC solution comprises feasible scheduling plans for a set of n requests with a pickup and delivery pair $(i, n+i)$. Thus, we adopt a chromosome array of random keys with $n+1$ genes with values in the interval $[0, 1)$ to represent a solution. Each gene $i \in \{1, \ldots, n\}$ is associated with a request in the format $(i, n+i)$, while the gene $i = n+1$ represents different decoder functions used to map the chromosome to the DARP-PFCC solution space. Next, we present all the decoders implemented in our paper.

5.2 Decoding a solution from a vector of random keys

Schenekemberg et al. (2022) presented four decoder functions to successfully solve the DARP-PFCC that can be adapted to handle instances with public transportation data. A decoder function is chosen according to the value of gene $n+1$ to map a random key vector to the DARP-PFCC solution space. For values in the range $[0.0,\ 0.25)$, the first decoder is selected; values in the interval $[0.25,\ 0.50)$ choose the second decoder; values in $[0.50,\ 0.75)$ select the third decoder, while values within $[0.75,\ 1.00)$ select the fourth decoder. The decoders used to solve DARP-PFCC integrated with public transportation are:

1. *Decoder 1* The first decoder is applied in two stages. Each request i is initially assigned to a vehicle $v \in \{1, \ldots, m\}$ of the private fleet. The vehicle v is determined by $v = \lfloor (r_i) \cdot m \rfloor + 1$, where r_i is the value of the random key associated with the request i. Then, routes are created for each vehicle v following a list Δ'_v of requests i sorted by non-descending order of r_i. The users of the first request $i \in \Delta'_v$ are added to the route between depots 0 and $2n+1$, obtaining the path $\mathcal{R}_v = (0, i, n+i, 2n+1)$. The remaining elements in Δ'_v are scheduled one by one in $\mathcal{R}_v$, in a common carrier or using integration with public

transportation. For that, we evaluate the cost of serving i and $n+i$ in the best positions in $\mathcal{R}_v$, considering only the partial path formed from the previously inserted pickup i to the depot $2n+1$. The cheapest cost for vehicle v is compared to the costs of scheduling the request in a common carrier or integrating it with a bus. The most economical option is chosen to serve each request. We also evaluate the route's pairing, time window, capacity, and ride time feasibility. Infeasible solutions are disregarded. Adding a request only after the pickup previously inserted in v avoids planning a complete schedule for the entire route. It is enough to evaluate the vicinity of the insertion and all nodes on the partial path leading to the depot $2n+1$. Appendix A shows an example of a solution and decoder considering the network and distances proposed in Schenekemberg et al. (2022).

2. *Decoder 2* This decoder is very similar to Decoder 1. The difference is that the cost of inserting i and $n+i$ is evaluated over the entire path of the vehicle v. We emphasize that the computational effort of this decoder is greater than the previous one, as it is necessary to plan a schedule for each node along the entire path.
3. *Decoder 3* This decoder does not apply the first stage as the previous ones to assign requests for a specific vehicle. It initially sorts the random keys by value and tries to schedule each request considering all private vehicles simultaneously. It also evaluates the cost of serving i and $n+i$ by common carriers and integrating with public transportation, always opting for the lowest cost option. Thus, this decoder is even more general than the first two decoders.
4. *Decoder 4* This decoder aims to minimize the usage of common carrier vehicles. It works as Decoder 3, except that an outsourced vehicle only serves a request if a feasible solution cannot be found using a private vehicle or an integration of a common carrier with a bus. If there is a feasible solution, the cheapest cost is used to select the transport option.

The four decoders described are specific constructive heuristics and help increase the solutions' diversity. During the evolutionary process of the BRKGA-QL, decoders that find better solutions are applied more frequently. However, all decoders can be invoked throughout the optimization process, generating different solutions.

5.3 Local search heuristics

The decision to use the common fleet or public transportation is binary. In contrast, the decision to serve by private fleet involves deciding which vehicle to use and which position on the route, respecting all precedence and time window constraints. For this reason, in this paper, most operators are designed for private fleets.

We implemented eight local search heuristics for the DARP-PFCC integrated with public transportation. The initial five heuristics mirror those outlined in Schenekemberg et al. (2022), while two have been modified to incorporate integration with public transportation services. The final heuristic, introduced in this research, aims to reduce the volume of requests managed through public transportation integration. All procedures are embedded into the RVND framework described in Sect. 4 and applied with parallel processing techniques, improving promising solutions without compromising the evolutionary process of the BRKGA-QL v2.0. At each iteration, a neighborhood is randomly chosen and a local optimal solution is found. If this solution is better than the current one, a new neighborhood is randomly chosen. Otherwise, the selected neighborhood is removed from the list of candidate neighborhoods. Figure 6 shows examples of these neighborhoods in a feasible solution represented by Fig. 5. The moves of the eight local search heuristics implemented in this paper are:

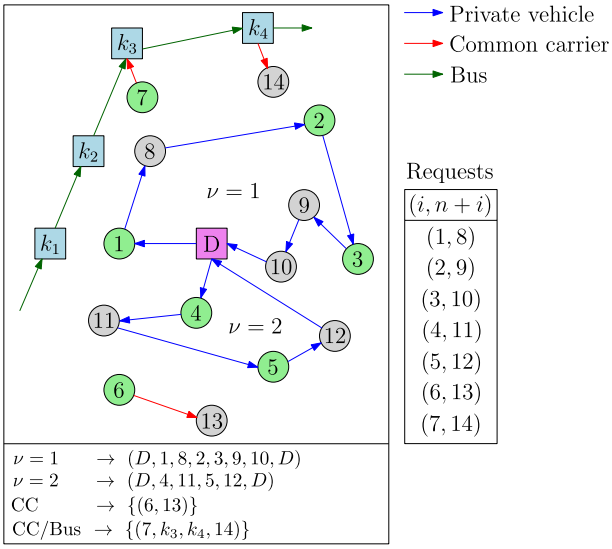
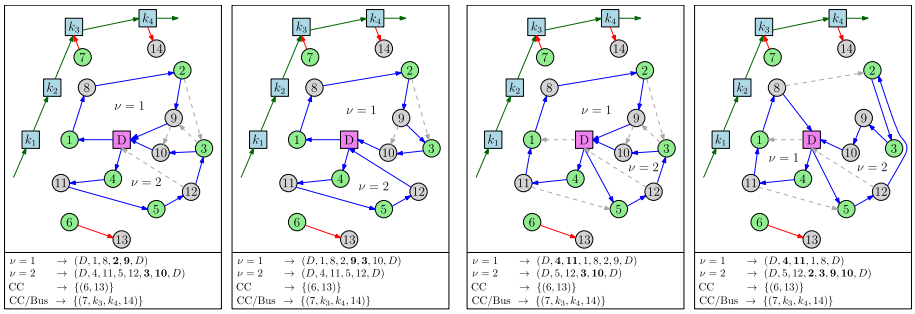
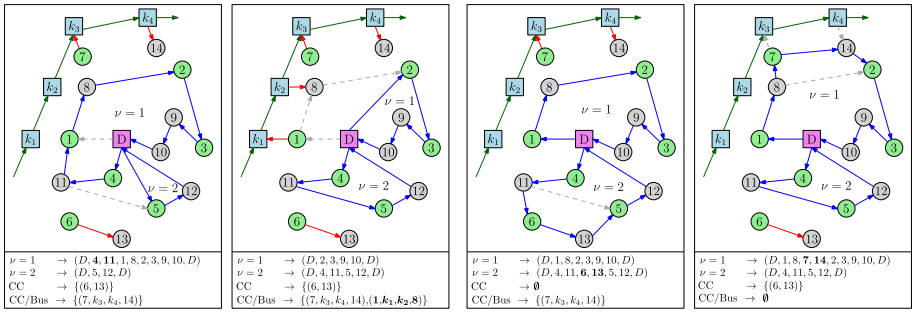


Fig. 5 A feasible solution for an instance with 7 requests

1. *Shift inter-route* This move removes a request i of a vehicle ν of the private fleet. Then, the pickup node i and delivery node $n+i$ are added to the route of another vehicle $\nu' \neq \nu$ according to the cheapest insertion rule;
2. *Swap up to four intra-route nodes* This move chooses a vehicle ν of the private fleet and a vertex u visited by it such that the partial path $[u, v, w, z]$ of four successive nodes belongs to the route $v_R = [0, \ldots, u, v, w, z, \ldots, 2n+1]$. Then, all 24 possible permutations in the partial path are computed, and the most economical among the feasible ones is selected.
3. *Swap two requests inter-route* This move selects two vehicles of the private fleet and exchanges two requests, i and i', between them. Requests i and i' are selected according to the greatest removal savings in their routes and are inserted in the best positions in the new routes by observing the cheapest insertion rule;
4. *Swap partial path inter-route* This move swaps partial paths between two selected vehicles from the private fleet. In this case, partial paths are natural sequences involving arcs where the load of the vehicle is null (Parragh et al., 2010);
5. *Remove* **1/3** This move randomly selects up to 1/3 of the requests served by vehicles of the private fleet and moves them to other vehicles or in different positions into the same vehicle. All insertions observe the feasibility of each route and the cheapest insertion rule;
6. *Remove a request from a private vehicle* This move selects a request i served by a vehicle of the private fleet and schedules it to a common carrier vehicle or the integration with public transportation. The cheapest cost is used to select the option to serve the request;
7. *Remove a request from a common carrier* This move removes a request i served by an outsourced vehicle and transfers its service to a vehicle of the private fleet or the integration with public transportation. Here, the cheapest insertion cost among all private vehicles is compared to the cost of the integration using common carriers and public transportation;
8. *Remove a request from the public transportation* This move schedules a request i served by the integration with a bus to a vehicle of the private fleet or a common carrier vehicle, following the cheapest cost rule.



(a) Shift inter-route. (b) Swap intra-route. (c) Swap inter-route. (d) Swap partial paths



(e) Remove 1/3. (f) Remove from PF. (g) Remove from CC. (h) Remove from PT.

Fig. 6 Example of neighborhoods

6 Computational experiments

In this section, we present the results of the extensive computational experiments we performed to evaluate the performance of the new version of the BRKGA-QL algorithm to solve the DARP integrated with common carriers and public transportation.

We compare the BRKGA-QL v2.0 against the BRKGA-QL algorithm developed by Schenekemberg et al. (2022). The authors solved the DARP-PFCC without integration with public transportation. We adapted their code to allow for comparisons between the two algorithms and used all decoder functions and local search procedures presented in Sect. 5 for both algorithms. The algorithms were implemented in C++ and ran on computers with Intel(R) Xeon(R) E5-2683 v4 @ 2.10GHz CPU.

All data sets and computational results are available at https://www.leandro-coelho.com/darp/, and the source code (problem-independent) of BRKGA-QL v2.0 is available at https://github.com/antoniochaves19/BRKGA-QL_v2.0.

6.1 Instances generator

To evaluate our algorithm, we use a real data set from a transportation provider that operates a dial-a-ride service in Québec, Canada. For confidentiality reasons, we slightly modified the user information so that it is not possible to determine the original pickup and delivery addresses. Figure 7 shows the pickup and delivery nodes of the requests along with the

Fig. 7 Map of Québec City with depot and pickup/delivery nodes

location of the depot. It can be observed that the requests are evenly distributed across the urban area with few isolated nodes.

Regarding the integration, we generated different scenarios varying the percentage of requests that can be integrated. In particular, we adopted the values $\zeta \in \{0\%, 25\%, 50\%, 75\%, 100\%\}$, where $\zeta = 0\%$ indicates a strict scenario in which no request should be served by bus. In contrast, $\zeta = 100\%$ represents a fully flexible scenario, allowing all requests to be integrated. We emphasize that when $\zeta = 0\%$, a DARP-PFCC instance Schenekemberg et al. (2022) is obtained. Finally, values 25%, 50%, and 75% represent intermediate scenarios. These scenarios are justified by noting that public transportation may be prohibitive for some users in real cases.

The DARP parameters were generated as follows. For each instance, Table 2 presents the number of requests (n), the number of vehicles (m) with identical capacity ($C = 5$), the maximum route duration (T), the maximum ride time duration (L_i), the demand (q_i), and the service duration (d_i) of each request i. The remaining columns in Table 2 show how we generated time windows ($[e_i, l_i]$) of outbound requests (from home to a destination) $i = 1, \ldots, n/2$ and inbound requests (return trip to home) $i = n/2+1, \ldots, n$. The parameters T, L_i, d_i, e_i, and l_i are represented in seconds, where $T = 28800$ denotes an eight-hour working day, in which each route must start only after 8 a.m (28,800 s) and finish before 4 p.m (57,600 s).

Given $n \in \{30, 60, 90, 120, 155\}$, we created one instance for each $\zeta \in \{0\%, 100\%\}$ and five instances for $\zeta \in \{25\%, 50\%, 75\%\}$, totaling 85 cases. We emphasize that in intermediate scenarios, all data are the same for each instance containing the same number of requests, only varying which requests can be integrated. In addition, travel times and distances were obtained from the road network and the public transportation database. As for cost values, the distances were multiplied by 0.4 for the private fleet, while the bus cost was assumed to be 3 for each trip. For the common carrier, we defined a fixed cost $f = 3$ and a variable cost

$v = 2$. These costs have been defined to represent the case of Québec City properly and need to be adjusted for other case studies.

Regarding public transportation, we considered three bus lines, where each line is described by buses that depart from the starting point at a particular time and travel through all stops at predetermined times until reaching the final stop. In our experiments, line 1 contains 50 departures with 61 stops each, while lines 2 and 3 have 3 and 47 departures containing 42 and 65 stops, respectively.

6.2 Computational results of BRKGA-QL and BRKGA-QL v2.0

Our experiments adopted $run_{time} = 2 \cdot n$ seconds as the stopping criterion for the two algorithms. The run_{time} parameter denotes the time in seconds for each run. Note that this is the only parameter to be set.

First, we compare the results between the two versions of BRKGA-QL. Table 3 presents the computational results for the 85 instances with different levels of integration. Each instance was run 10 times, and columns "Best Z" and "Avg. Z" show the values of the objective function of the best and average solutions, respectively. Both versions use the maximum running time (run_{time}) as a stopping criterion. Thus, the column "Best T(s)" shows the average computational time for the algorithm to find the best solution in each run and the column "T(s)" provides the total execution time in seconds. Additionally, the columns "Δ (%) Best Z" and "Δ (%) Avg. Z" present the relative percentage deviations concerning the best and the average solution obtained by both versions, respectively. The deviation value "Δ (%) Best Z" is calculated as $100 \cdot (Z_{v2.0} - Z)/Z$, where Z and $Z_{v2.0}$ denote the "Best Z" for BRKGA-QL and BRKGA-QL v2.0, respectively. Negative values indicate that version 2.0 found better solutions. The same idea is valid for "Δ (%) Avg. Z".

When no requests can be integrated with buses ($\zeta = 0\%$), we have the same case presented in Schenekemberg et al. (2022). Comparing the BRKGA-QL versions only on these instances, it can be seen that there are significant improvements in version 2.0 on instances with more than 30 requests. We can also observe that BRKGA-QL v2.0 improved the results in all analyzed scenarios except for the instances with the lowest number of requests (30). BRKGA-QL v2.0 obtains objective function values on average 5.75% better than the original version. The computational time to achieve the best solution is similar in both versions. These times increase according to the number of requests but are stable for different integration levels. The data pre-processing presented in Sect. 3 was fundamental in supporting this computational time stability.

Figure 8 shows the box plot of the two versions considering the gap between the best and average solutions to the best-known ones. We can see that BRKGA-QL v2.0 finds the best solution in almost all instances, while BRKGA-QL has a median of 6.8%. In terms of the average solutions, the third quartile of BRKGA-QL v2.0 was similar to the first quartile of BRKGA-QL (7.04% and 6.53%), highlighting the excellent performance of the new components in version 2.0 of the algorithm. Figure 9 presents the performance profile (Dolan & More, 2002) of the BRKGA-QL versions considering tolerance of quality result as a gap of less than 1% to the best-known solution. We can observe that BRKGA-QL v2.0 dominates the previous version and obtains the best solutions regarding computational time and accuracy. BRKGA-QL v2.0 was the fastest algorithm in 92% of the instances, and BRKGA-QL was the fastest algorithm in 8% of the instances. Version 2.0 found the target solution in 96% of the instances, while version 1.0 found the target only in 26%. We also performed statistical analysis with the Wilcoxon signed-rank test (Rey & Neuhäuser, 2011), proving a significant

Table 2 Dial-a-ride parameters

n	m	T	L_i	q_i	d_i	e_i	l_i	e_{n+i}	l_{n+i}
30	1	28,800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
60	2	28,800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
90	3	28,800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
120	3	28,800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
155	3	28,800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$

Table 3 Computational results of the BRKGA-QL and BRKGA-QL v2.0 applied to solve DARP-PFCC integrated with public transportation

n	Integration ζ (%)	BRKGA-QL				BRKGA-QL v2.0				Δ (%)	Δ (%)
		Best Z	Avg. Z	Best T(s)	T(s)	Best Z	Avg. Z	Best T(s)	T(s)	Best Z	Avg. Z
30	0	281.50	282.76	50.52	60.00	281.50	281.53	28.42	60.00	0.00	−0.46
	25	257.02	260.95	38.79	60.00	258.04	260.56	24.94	60.00	0.42	−0.11
	50	257.11	259.73	34.50	60.00	255.54	258.31	24.14	60.00	−0.59	−0.54
	75	239.76	241.65	35.10	60.00	239.84	240.91	24.20	60.00	0.04	−0.29
	100	224.22	224.81	33.55	60.00	224.22	224.40	23.96	60.00	0.00	−0.18
60	0	633.27	670.24	97.24	120.00	577.22	617.51	107.93	120.00	−8.85	−7.86
	25	593.03	623.64	99.26	120.00	536.39	566.89	111.28	120.00	−9.48	−9.09
	50	558.24	584.61	98.10	120.00	510.83	539.40	111.62	120.00	−8.51	−7.73
	75	535.91	563.05	94.15	120.00	493.21	521.16	107.61	120.00	−7.94	−7.44
	100	529.36	547.15	101.40	120.00	481.70	499.97	100.82	120.00	−9.00	−8.63
90	0	941.39	977.06	129.54	180.00	855.86	918.73	148.07	180.00	−9.09	−5.98
	25	881.72	923.15	134.13	180.00	778.76	851.75	159.50	180.00	−11.68	−7.74
	50	827.84	875.29	136.67	180.00	736.83	814.97	161.53	180.00	−10.98	−6.89
	75	770.39	807.59	144.74	180.00	690.67	753.49	157.54	180.00	−10.25	−6.70
	100	701.03	759.54	137.81	180.00	660.41	717.30	175.62	180.00	−5.79	−.5.56
120	0	1617.92	1675.18	186.79	240.00	1480.04	1576.25	221.61	40.00	−8.52	−5.91
	25	1471.02	1545.24	205.61	240.00	1436.10	1491.39	226.41	240.00	−2.30	−3.48
	50	1411.16	1464.07	197.07	240.00	1333.82	1402.93	228.71	240.00	−5.48	−4.18
	75	1294.00	1361.58	182.19	240.00	1266.30	1314.65	226.04	240.00	−2.08	−3.44
	100	1218.69	1279.57	211.08	240.00	1178.86	1238.29	216.35	240.00	−3.27	−3.23

Table 3 continued

n	Integration ζ (%)	BRKGA-QL				BRKGA-QL v2.0				Δ (%)	Δ (%)
		Best Z	Avg. Z	Best T(s)	T(s)	Best Z	Avg. Z	Best T(s)	T(s)	Best Z	Avg. Z
155	0	2600.56	2704.95	292.08	310.00	2407.82	2569.92	292.55	310.00	−7.41	−4.99
	25	2385.11	2468.15	273.74	310.00	2272.08	2385.28	297.59	310.00	−4.74	−3.36
	50	2201.34	2289.52	282.48	310.00	2054.71	2189.23	308.87	310.00	−6.65	−4.38
	75	2032.72	2099.35	253.62	310.00	1924.67	2017.74	294.83	310.00	−5.28	−3.89
	100	1871.35	1959.01	277.77	310.00	1751.11	1889.27	292.35	310.00	−6.43	−3.56
Avg.		1053.42	1097.92	149.11	182.00	987.46	1045.67	162.90	182.00	−5.77	−4.62

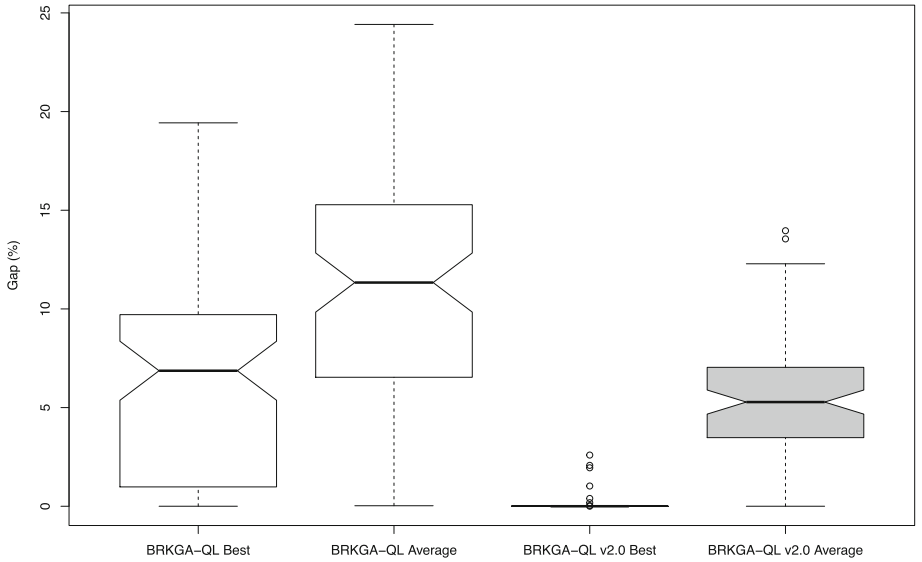


Fig. 8 Box plot of the gap between the best and average BRKGA-QL and BRKGA-QL v2.0 solutions to the best-known solutions of the DARP-PFCC integrated with public transportation

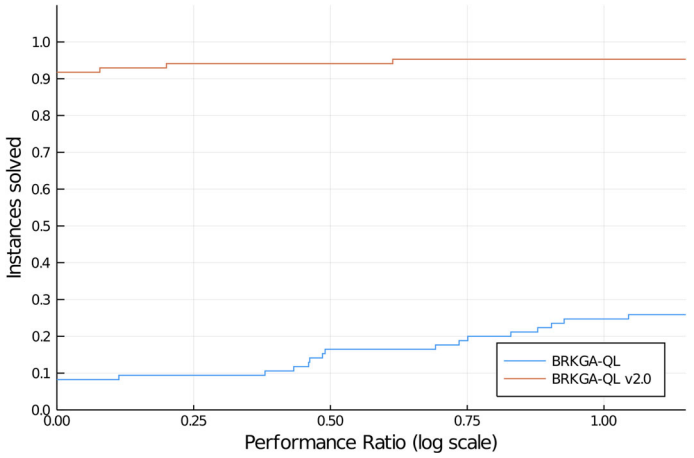


Fig. 9 Performance profile of runtime for BRKGA-QL and BRKGA-QL v2.0 to find the best solution of the DARP-PFCC integrated with public transportation

difference between the solutions of the two versions (p-value $= 3.455e-13$). These results show that the BRKGA-QL v2.0 is significantly more robust and efficient.

We observe that as the percentage of requests eligible for integration increases, there is a notable enhancement in the cost efficiency of the solution, averaging a 6% improvement across different integration levels. This improvement stems from the algorithm having a broader range of options to determine which requests should be served by a private fleet, common carrier, or integrated with a bus. Consequently, the most cost-effective requests are assigned to vehicle routes, while others are served by common carriers or integrated buses, always prioritizing the most advantageous option. Thus, the more requests can be integrated,

the more accurate the choices become. Compared to the DARP-PFCC version proposed by Schenekemberg et al. (2022) with 0% integration, we achieved an average cost reduction of 27% by incorporating bus integrations. We provide a deeper analysis of these effects in the next section.

We also compare the results of the BRKGA-QL v2.0 with the results of the branch-and-cut algorithm (B&C) proposed by Schenekemberg et al. (2022) to solve the DARP-PFCC. In Appendix B, we show how to adapt the B&C algorithm to handle public transportation data and provide a comparative analysis of both algorithms.

6.3 Qualitative analysis of DARP-PFCC integrated with public transportation

Table 4 presents the costs related to the routes considering five levels of integration. Costs are separated into private fleet costs, common carrier costs (fixed and variable), and public transportation (bus) costs. The column "Total" shows the total cost to fulfill all requests. The columns "Δ (%) Distance" and "Δ (%) $Z_{0\%}$" show the deviations in distance and costs concerning the solution without integration with public transportation ($\zeta = 0\%$). The deviation values are computed as in Sect. 6.2. When compared to the scenario with 0% integration, positive values for "Δ (%) Distance" indicate an increase in the distance traveled, whereas negative values for "Δ (%) $Z_{0\%}$" denote a decrease in the total cost.

We can observe that costs decrease as the integration with public transportation increases. However, we observe that the costs of the private fleet remain stable, which shows the efficient use of this fleet in all scenarios analyzed. Therefore, the cost reduction is related to decreased distances traveled by the common carrier. For instances with less integration (25%), there is an average savings of 6.6%, and for instances with 100% of integration, there is an average savings of 21.5%. There was a decrease in the distances traveled by users in two scenarios, and there was a maximum increase of 3.44% for instances with the highest number of requests.

Tables 5 and 6 show the time and distance components of integrated trips. All integration scenarios present similar percentages for the first, middle, and last miles. The middle mile accounted for 44% of ride times and 67% of distances due to longer bus routes. The "Ride Time" and "Distance Total" columns present the total times and distances of the three miles, and the "Direct trip" columns present hypothetical values if the trips were carried out point-to-point with a common carrier. Interestingly, ride times increase considerably, but the total distance is similar to the integrated bus option. Another interesting observation is that times and distances decrease as more users can be integrated, allowing the algorithm to select the most economical users to perform the integration.

Figure 10 shows the results for each integration percentage, representing the relative quality and cost to test various scenarios of the DARP-PFCC integrated with buses. The x-axis represents the value of ζ, while the y-axis represents the percentage of cost relative to the cost with 0% integration and the percentage of additional ride time for users compared to direct trips. Each point on the graph corresponds to a specific value of ζ and shows the corresponding percentages for cost and quality. We can observe that with 25% integration, costs decrease to 93%, while ride time increases by 64%. At 50% and 75% integration levels, costs decrease by 88% and 83%, respectively, with ride times remaining constant at 66% and 65%, respectively. With 100% integration, the cost represents 79% of the cost of the solution with 0% integration, while ride time has increased by 75%. The decision maker needs to observe the tradeoff between cost and quality of service.

Figure 11 shows an analysis of the time components of the private fleet considering each level of integration. We break down the operating time of all vehicles into (i) traveling

Table 4 Comparison of route cost distribution

n	Integration ζ (%)	Cost components				Total	Δ (%) Distance	Δ (%) $Z_{0\%}$
		Private	Fixed	Variable	Bus			
30	0	89.46	24.00	168.04	–	281.50	–	–
	25	91.83	19.80	121.08	25.33	258.04	2.95	−8.33
	50	91.91	18.60	100.09	44.94	255.54	5.14	−9.22
	75	92.21	15.60	67.75	64.28	239.84	4.94	−14.80
	100	94.04	12.00	22.62	95.56	224.22	7.99	−20.35
60	0	212.18	51.00	314.04	–	577.22	–	–
	25	213.10	36.60	199.64	87.05	536.39	3.03	−7.07
	50	214.22	27.60	141.59	127.42	510.83	4.26	−11.50
	75	210.80	21.00	85.18	176.23	493.21	4.44	−14.55
	100	216.14	18.00	65.94	181.62	481.70	5.89	−16.55
90	0	332.98	66.00	456.88	–	855.86	–	–
	25	326.34	50.40	284.13	117.89	778.76	−0.44	−9.01
	50	318.30	34.20	162.58	221.75	736.83	0.64	−13.91
	75	321.26	25.80	88.67	254.94	690.67	1.64	−19.30
	100	295.67	27.00	63.98	273.76	660.41	−4.98	−22.84
120	0	334.88	135.00	1010.16	–	1480.04	–	–
	25	328.20	108.60	799.16	200.14	1436.10	2.99	−2.97
	50	338.44	87.60	557.04	350.74	1333.82	5.73	−9.88
	75	314.83	57.60	337.29	556.58	1266.30	4.54	−14.44
	100	322.80	45.00	211.90	599.16	1178.86	2.13	−20.35

Table 4 continued

n	Integration ζ (%)	Cost components				Total	Δ (%) Distance	Δ (%) $Z_{0\%}$
		Private	Fixed	Variable	Bus			
155	0	360.02	249.00	1798.80	–	2407.82	–	–
	25	340.15	191.40	1434.87	305.66	2272.08	0.55	−5.64
	50	340.10	142.20	887.87	684.54	2054.71	1.17	−14.67
	75	338.55	114.00	597.88	874.24	1924.67	3.44	−20.07
	100	347.91	102.00	437.18	864.02	1751.11	0.88	−27.27
Avg.		259.45	67.20	416.57	305.29	987.46	2.95	−13.44

Table 5 Time components of integrated trips

Integration ζ (%)	Percentage of time					Ride time	Direct trip
	First mile	Middle mile	Last mile	Service	Waiting		
25	6.26	44.98	8.95	10.57	29.24	3404.31	1268.62
50	5.21	42.66	7.41	10.44	34.28	3447.27	1175.90
75	5.76	43.94	7.45	10.77	32.08	3344.02	1157.51
100	5.40	44.31	7.81	10.83	31.65	3323.93	842.82
Avg.	5.65	43.97	7.90	10.65	31.81	3379.88	1111.21

Table 6 Distance components of integrated trips

Integration ζ (%)	Percentage of distance			Distance total	Direct trip
	First mile	Middle mile	Last mile		
25	13.49	64.94	21.57	18.31	18.05
50	11.96	69.21	18.83	17.05	16.62
75	13.29	68.02	18.69	16.70	16.17
100	12.94	67.14	19.92	16.31	15.84
Avg.	12.92	67.32	19.75	17.09	16.67

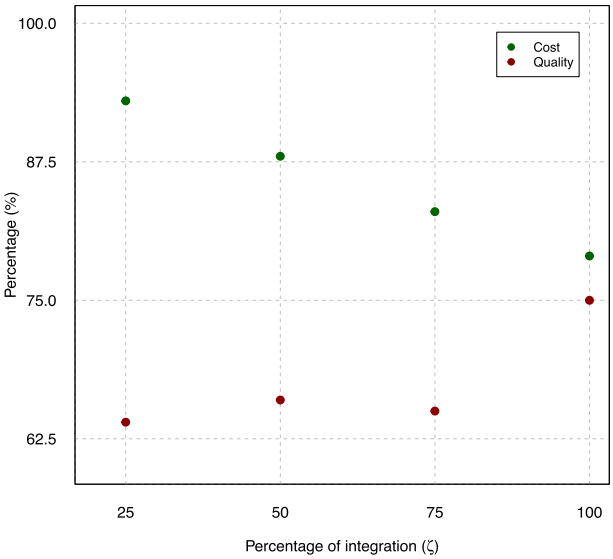


Fig. 10 Cost vs Quality for different percentages of integration

between two nodes (Traveling), (ii) waiting at a node to perform the service (Waiting), (iii) serving a node (Service), and (iv) housed at the depot (Parked). We observe that all scenarios had similar values (70% for traveling, 3% for waiting, 19% for service, and 8% for parked). On average, each vehicle serves users (traveling or service) around 89% of the time. These values indicate good operational performance for the private fleet vehicles.

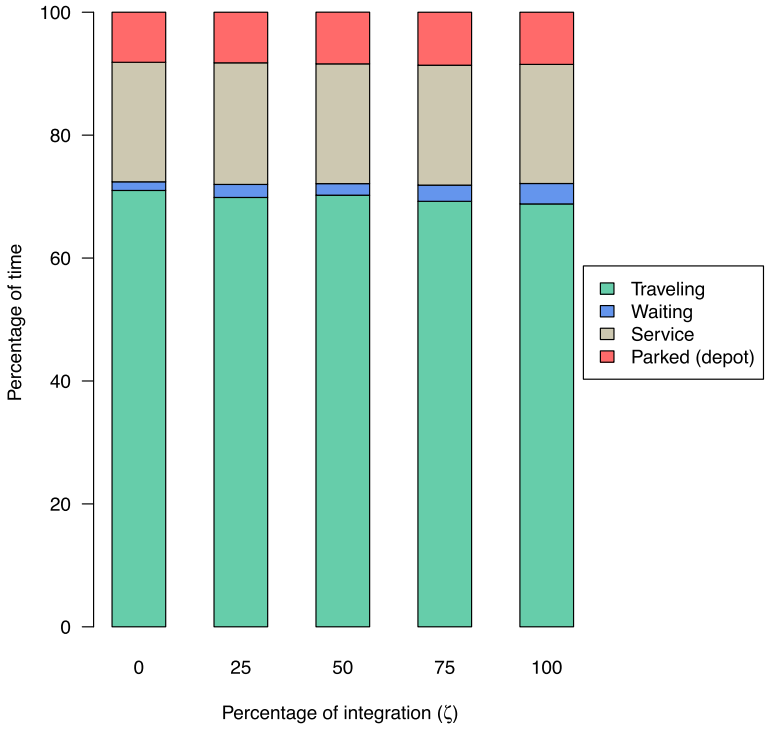


Fig. 11 Private fleet operating time breakdown

In Fig. 12, we analyze the mileage savings in each scenario. We consider the worst-case scenario to satisfy all requests with a point-to-point common carrier to measure these savings. We observed that the DARP-PFCC without bus integration ($\zeta = 0\%$ of integration) presents an average reduction of 14% of the distances traveled by vehicles. As more requests can use the integration, we observe larger savings. In the scenario with 100%, we have an average saving of 30% of the distances. These savings mean fewer vehicles in traffic and less CO_2 emissions.

Tables 7 and 8 display the average costs and distances per request for the three modes of transport: private fleet, common carrier, and common carrier integrated with buses. For the latter, we calculated the average per request if they were served door-to-door on direct trips. The last column shows the variation in cost or distance for this case. It can be observed that the average costs and distances for requests handled by the private fleet do not vary significantly. This data is supported by Fig. 11 and demonstrates that private vehicles have good operational performance across all scenarios. Conversely, the distances and costs of common carriers decrease as the number of requests that can be integrated increases. For the integrated requests, the average cost per request is similar to that of the common carrier. However, if these requests were handled by common carriers alone, the average cost would be 81% higher. In terms of distance, common carrier or integrated service is equivalent.

On the other hand, Table 9 presents the average ride times per request handled by the private fleet, integrated with buses and common carriers. We calculated the average ride time for the private fleet and integrated cases as if these requests were to make direct trips using the common carrier. It can be observed that the ride time with the common carrier decreases

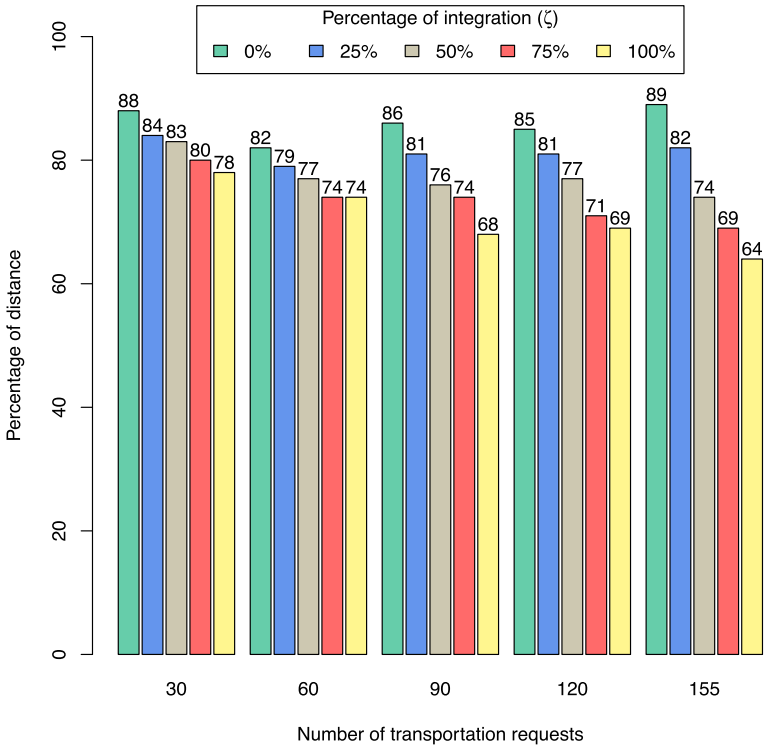


Fig. 12 Percentage of distance traveled relative to the worst-case scenario

as more requests can be integrated. This characteristic shows that the algorithm prioritizes assigning shorter trips to the common carrier, considering its high variable cost. The average ride time for requests integrated with buses exceeds that of requests handled by a private fleet. However, the increase is proportionate to the direct ride time using the common carrier in both cases. This suggests that trips served with integration are longer. Therefore, the level of service quality remains comparable for requests served by a private fleet and those integrated with buses.

7 Conclusion

This paper considered the integration of the dial-a-ride problem with common carriers and public transportation to reduce the operational costs of the pickup and delivery service. A new version of the BRKGA-QL metaheuristic was developed to obtain good solutions in a short computational time. We also developed a pre-processing method to enable efficient integration between the transportation modes, considering bus lines, schedules, and bus stop locations. Two BRKGA-QL versions solved 85 instances with different integration scenarios, where only some requests can use public transportation integration. The results show that BRKGA-QL v2.0 had a better performance than the previous version, mainly due to a better balance of diversification and intensification of the Q-Learning method and the use of mutation in the evolutionary process of BRKGA to allow diversity and good quality fitness

Table 7 Average cost per requisition with different levels of integration

n	Integration ζ (%)	Average cost			Direct trip	Δ (%)
		Private fleet	Common carrier	Integrated		
30	0	4.1	24.0	–	–	–
	25	4.2	21.2	16.4	35.4	115.9
	50	4.3	18.8	18.5	40.3	117.8
	75	4.3	15.3	18.9	35.7	88.9
	100	4.5	8.7	19.1	36.5	91.1
60	0	4.9	21.5	–	–	–
	25	4.8	19.2	27.3	47.9	75.5
	50	4.8	18.4	19.2	36.0	87.5
	75	4.8	15.1	19.6	34.4	75.5
	100	4.9	14.0	18.2	32.7	79.7
90	0	4.9	23.8	–	–	–
	25	4.8	20.2	21.5	37.3	73.5
	50	4.7	16.9	19.5	36.2	85.6
	75	4.8	13.4	18.6	35.4	90.3
	100	4.6	10.1	16.1	30.2	87.6
120	0	4.5	25.4	–	–	–
	25	4.4	25.1	21.5	37.3	73.5
	50	4.7	22.1	18.9	33.8	78.8
	75	4.3	20.6	20.8	35.5	70.7
	100	4.2	17.1	21.4	34.7	62.1
155	0	5.0	24.7	–	–	–
	25	4.4	25.5	22.5	37.6	67.1
	50	4.5	21.8	21.4	34.9	63.1
	75	4.6	18.8	20.4	35.6	74.5
	100	4.8	15.9	18.0	32.5	80.6
	Avg	4.59	19.10	19.89	36.00	81.96

solutions. In terms of integration, we observed a decrease in trip costs as the number of requests that can use public transportation increased. In the best case, there were 27% cost savings without a significant increase in distance. However, the integration led to longer trip times due to the use of buses.

Computational experiments have shown the advantages of this integration in terms of cost savings while maintaining ride times equivalent to those of the private fleet. Therefore, given the limitations on fulfilling all requests with a private fleet, the integration of common carriers and buses can reduce costs and traveled distances by common carriers. It is an important issue that saves CO_2 emissions, considering that vehicles travel shorter distances in the first and last miles, while public transportation is used in the middle mile.

New studies to measure CO_2 emissions considering other characteristics may indicate alternative public policies for the integration of different transport modes. For future research, we recommend generating new instances with different characteristics and studying the use of the private fleet and car-sharing in the first and last miles, as well as other types of transport such as electric mini cars, scooters, and bicycles.

Table 8 Average distance per requisition with different levels of integration

n	Integration ζ (%)	Average distance			Direct trip	Δ (%)
		Private fleet	Common carrier	Integrated		
30	0	10.2	10.5	–	–	–
	25	10.5	9.1	16.2	16.2	0.0
	50	10.7	7.9	19.3	18.7	−3.1
	75	10.8	6.2	16.9	16.4	−3.0
	100	11.2	2.8	17.2	16.8	−2.3
60	0	12.3	9.2	–	–	–
	25	12.1	8.1	22.5	22.5	0.0
	50	12.1	7.7	16.8	16.5	−1.8
	75	12.1	6.1	16.4	15.7	−4.3
	100	12.3	5.5	15.5	14.9	−3.9
90	0	12.2	10.4	–	–	–
	25	12.1	8.6	17.6	17.2	−2.3
	50	11.8	6.9	17.0	16.6	−2.4
	75	11.9	5.2	16.8	16.2	−3.6
	100	11.5	3.6	13.9	13.6	−2.2
120	0	11.2	11.2	–	–	–
	25	11	11	17.5	17.2	−1.7
	50	11.7	9.5	15.8	15.4	−2.5
	75	10.6	8.8	16.7	16.3	−2.4
	100	10.5	7.1	16.4	15.9	−3.0
155	0	12.5	10.8	–	–	–
	25	11	11.2	17.7	17.3	−2.3
	50	11.4	9.4	16.4	16.0	−2.4
	75	11.4	7.9	16.7	16.3	−2.4
	100	11.9	6.4	15.1	14.7	−2.6
	Avg	11.48	8.04	16.92	16.52	−2.41

Appendix A: The encoding and decoding processes

We now describe the instance set proposed by Schenekemberg et al. (2022) for the DARP-PFCC without integration with public transportation. In the original instance, the task involves transporting $n = 8$ requests from a set of pickup nodes, $\mathcal{P} = \{1, \ldots, 8\}$, to their corresponding delivery nodes, $\mathcal{D} = \{9, \ldots, 16\}$. Each node $i \in \mathcal{P}$ has a maximum ride-time $L_i = 40$, a service duration $d_i = 3$, and a demand $q_i = 1$ ($q_i = -1$ for $i \in \mathcal{D}$). The private fleet consists of $m = 2$ vehicles with capacity $Q = 3$. Both vehicles must start their routes from the origin depot (node 0) and conclude their journeys at the destination depot (node 17), ensuring that the total travel time does not exceed the maximum limit of $T = 120$. For the common carrier vehicles, a fixed cost of $f = 3$ and a variable cost of $v = 2$ are considered. To create a DARP-PFCC integrated with public transportation instance, we consider a bus line l with bus stops with a distance of 3 units from any node $i \in \mathcal{P} \cup \mathcal{D}$. Finally, we also consider a ticket cost of $p_l = 3$ for each bus trip.

Table 9 Average ride time per request with different levels of integration

n	Integration ζ (%)	Average ride time						
		Private fleet	Direct trip	Δ (%)	Integrated	Direct trip	Δ (%)	Common carrier
30	0	2729.4	755.0	261.51	–	–	–	729.3
	25	2824.8	751.9	275.69	2967.0	1026.5	189.04	660.8
	50	2937.6	751.8	290.74	4157.3	1130.0	267.90	602.1
	75	3098.9	758.6	308.50	3263.1	1012.2	222.38	501.5
	100	3351.3	760.9	340.44	3441.3	1038.3	231.44	318.6
60	0	2457.0	1000.4	145.60	–	–	–	621.2
	25	2798.7	950.0	194.60	3967.1	1410.8	181.20	571.1
	50	2691.6	943.4	185.31	3378.5	1060.8	218.49	540.6
	75	2900.1	936.0	209.84	3524.0	1034.6	240.61	455.1
	100	3077.9	937.7	228.24	3575.6	987.3	262.16	407.6
90	0	2604.6	933.4	179.04	–	–	–	748.3
	25	2835.6	938.1	202.27	3327.7	1125.1	195.77	629.9
	50	2827.7	916.9	208.40	3246.0	1063.3	205.28	529.7
	75	2833.1	913.6	210.10	3325.3	1056.5	214.75	427.1
	100	2882.8	957.9	200.95	3491.6	922.7	278.41	327.4
120	0	2844.3	905.3	214.18	–	–	–	752.0
	25	2783.1	865.5	221.56	3322.7	1092.8	204.05	751.9
	50	2762.8	884.6	212.32	3143.0	994.4	216.07	665.4
	75	2815.2	837.5	236.14	3323.7	1043.4	218.55	619.1
	100	2752.9	843.6	226.33	3245.6	1045.9	210.32	499.9

Table 9 continued

n	Integration ζ (%)	Average ride time						
		Private fleet	Direct trip	Δ (%)	Integrated	Direct trip	Δ (%)	Common carrier
155	0	3039.2	985.5	208.39	–	–	–	726.4
	25	2830.8	872.1	224.60	3436.9	1087.9	215.92	767.3
	50	2815.5	896.6	214.02	3311.6	1030.9	221.23	650.9
	75	2840.3	878.6	223.28	3284.0	1040.9	215.50	570.7
	100	2746.1	944.8	190.65	3263.2	956.4	241.20	481.4
	Avg	2843.3	884.8	224.51	3399.8	1058.0	222.51	582.2

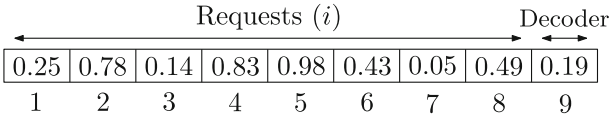


Fig. 13 Encoding the instance in a nine-element chromosome

A DARP-PFCC variant solution is represented by a nine-element chromosome, where the transport requests in the form $(i, n+i)$ are placed from 1 to 8 positions, and the index 9 defines the decoder. Figure 13 details random key values for such chromosome, following the same values presented in Schenekemberg et al. (2022). In this case, the value 0.19 indicates that **Decoder 1** must be applied to transform the remaining random key values into a solution for the problem.

The **Decoder 1** is applied in two stages. First, requests from 1 to 8 are assigned to a private vehicle $\nu \in \{1, 2\}$ according to $\nu = \lfloor (r_i) \cdot 2 \rfloor + 1$. Then, each vehicle route ν is designed to observe all the requests sorted by the random key values (list Δ'_ν). In Fig. 14, we show how the list of requests $\Delta'_1 = \{7, 3, 1, 6, 8\}$ and $\Delta'_2 = \{2, 4, 5\}$ are generated from the array of random key values. In what follows, we design only the route for vehicle $\nu = 1$. For that, time windows as in Table 10 and distance and travel time values as in Table 11 are considered.

Table 12 details each iteration of the decoding process using **Decoder 1**. The first column presents the current iteration, while the column $(i, n+i)$ denotes the request $i \in \{7, 3, 1, 6, 8\}$ being served during that iteration in particular. The third and fourth columns show all possible routes for the private vehicle $\nu = 1$. For each route, the current pickup (delivery) node appears in red (blue). The next three columns calculate the marginal cost of serving i by the private vehicle, by a common carrier vehicle, or by integrating a common carrier vehicle with a bus. Smaller values mean less impact on the objective function. In addition, costs marked with "*" denote an infeasible route for the private vehicle. Finally, "Best Option" presents the chosen option to serve i, while the "$Z(s)$" is the objective function value. All iterations needed to define a solution for the list Δ'_1 are described in sequence.

Request 7 is scheduled between nodes 0 and 17 in the route of $\nu = 1$ in the first iteration, such that $0-7-15-17$ and $Z(s) = 20$. In the second iteration, the request 3 can be inserted only after the last inserted pickup node in the route of $\nu = 1$. Thus, we examine all partial paths between 7 and 17. Of the three positions available, the cheapest cost is $\# = 1$. This option is even more advantageous than serving the request by a common carrier vehicle or its integration with the bus. The current route is now $0-7-3-11-15-17$ with $Z(s) = 35$. The third iteration selects a common carrier vehicle to serve the request 1. Thus, the route of $\nu = 1$ is the same as the previous iteration and $Z(s) = 48$. Request 6 can be inserted in the route of $\nu = 1$ in six ways in the fourth iteration, where $\# = 5$ is the cheapest option. The chosen route is $0-7-3-11-6-15-14-17$ with $Z(s) = 67$. Finally, the request 8 is served by integrating common carrier vehicles and buses. In this case, all options using $\nu = 1$ are infeasible. Therefore, request 1 is served by a common carrier vehicle, request 8 by the integration with bus, and the remaining requests are scheduled in the route of $\nu = 1$ $(0-7-3-11-6-15-14-17)$ with $Z(s) = 88$.

Appendix B: Branch-and-cut algorithm results

Schenekemberg et al. (2022) developed a branch-and-cut (B&C) algorithm to exactly solve the DARP-PFCC, which can be adapted to handle instances with public transportation data.

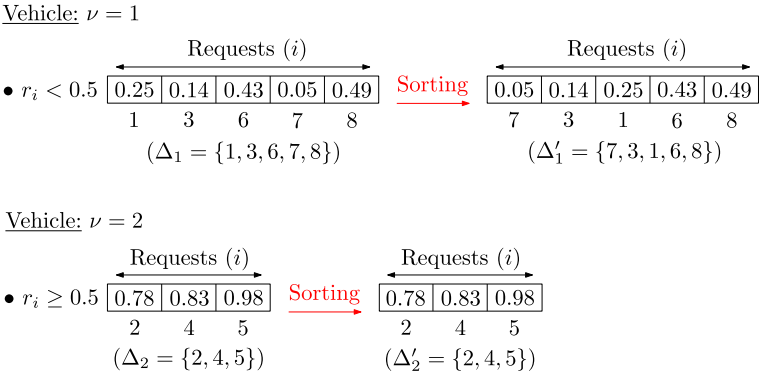


Fig. 14 Decoding the instance using Decoder 1

Table 10 Time windows

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

The algorithm focuses on optimizing both the routing cost of the private fleet, calculated as $c_{private} = c_{i,j} \cdot d_{i,j}$ for each edge (i, j), and the fixed and variable costs associated with common carriers, denoted as $c_{extra}(i) = f + v \cdot d_{i,n+i}$ for each request i.

To solve instances of the DARP-PFCC integrated with public transportation, one can define $c_{extra}(i) = min\{f + v \cdot d_{i,n+i};\ f + v \cdot d_{i,k} + p_l + f + v \cdot d_{k',n+i}\}$. Here, $c_{extra}(i)$ represents the minimum cost for serving request i either via a common carrier or by integrating common carriers and buses at stops k and k'. In this context, it is crucial to compute the cost $c_{extra}(i)$ of request i before the optimization process.

Table 13 provides a comparison between the results achieved by running the B&C algorithm for 7,200 s and the results obtained from the BRKGA-QL v2.0, as discussed in Sect. 6.2. The first six columns follow the same definitions as described in Sect. 6.2. On the other hand, columns "UB", "LB", "$GAP(\%)$", and "$T(s)$" correspond to the upper and lower bounds, optimality gap, and execution time in seconds for the B&C algorithm, respectively. Lastly, the column "$\Delta\%\ UB$" defines the relative percentage deviation of the best solution obtained by both algorithms. Negative values indicate that BRKGA-QL 2.0 found better solutions. The results show that the B&C algorithm achieved an average gap greater than 87%, with an upper bound UB (solution cost) notably worse than that of the solutions obtained by BRKGA-QL. Furthermore, no optimal solution was proven during the B&C runs.

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 Canadian Natural Sciences and Engineering Research Council (NSERC) 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

Table 11 Distance and travel time values used in the decoding process

i	j	$c_{i,j} = t_{i,j}$	i	j	$c_{i,j} = t_{i,j}$	i	j	$c_{i,j} = t_{i,j}$	i	j	$c_{i,j} = t_{i,j}$	i	j	$c_{i,j} = t_{i,j}$
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			

Table 12 Iterations of the Decoder 1

Iter.	$(i, n+i)$	Private Fleet		Marginal Cost			Best	$Z(s)$
		Route	#	PF	CC	CC/Bus	Option	
1	(7, 15)	0–7–15–17	1	$c_{0,7} + c_{7,15} + c_{15,17} - c_{0,17} = 20$	25	21	Route #1	20
2	(3, 11)	0–7–3–11–15–17	1	$c_{7,3} + c_{3,11} + c_{11,15} - c_{7,15} = 15$	23	21	Route #1	35
		0–7–3–15–11–17	2	$c_{7,3} + c_{3,15} + c_{15,11} + c_{11,17} - c_{7,15} - c_{15,17} = 34$				
		0–7–15–3–11–17	3	$c_{15,3} + c_{3,11} + c_{11,17} - c_{15,17} = 37^*$				
3	(1, 9)	0–7–3–1–9–11–15–17	1	$c_{3,1} + c_{1,9} + c_{9,11} - c_{3,11} = 30^*$	13	21	C. Carrier	48
		0–7–3–1–11–9–15–17	2	$c_{3,1} + c_{1,11} + c_{11,9} + c_{9,15} - c_{3,11} - c_{11,15} = 47^*$				
		0–7–3–1–11–15–9–17	3	$c_{3,1} + c_{1,11} + c_{15,9} + c_{9,17} - c_{3,11} - c_{15,17} = 56^*$				
		0–7–3–11–1–9–15–17	4	$c_{11,1} + c_{1,9} + c_{9,15} - c_{11,15} = 26^*$				
		0–7–3–11–1–15–9–17	5	$c_{11,1} + c_{1,15} + c_{15,9} + c_{9,17} - c_{11,15} - c_{15,17} = 50^*$				
		0–7–3–11–15–1–9–17	6	$c_{15,1} + c_{1,9} + c_{9,17} - c_{15,17} = 30^*$				
4	(6, 14)	0–7–3–6–14–11–15–17	1	$c_{3,6} + c_{6,14} + c_{14,11} - c_{3,11} = 34^*$	37	21	Route #5	67
		0–7–3–6–11–14–15–17	2	$c_{3,6} + c_{6,11} + c_{11,14} + c_{14,15} - c_{3,11} - c_{11,15} = 47^*$				
		0–7–3–6–11–15–14–17	3	$c_{3,6} + c_{6,11} + c_{15,14} + c_{14,17} - c_{3,11} - c_{15,17} = 31^*$				
		0–7–3–11–6–14–15–17	4	$c_{11,6} + c_{6,14} + c_{14,15} - c_{11,15} = 34^*$				
		0–7–3–11–6–15–14–17	5	$c_{11,6} + c_{6,15} + c_{15,14} + c_{14,17} - c_{11,15} - c_{15,17} = 19$				
		0–7–3–11–15–6–14–17	6	$c_{15,6} + c_{6,14} + c_{14,17} - c_{15,17} = 37^*$				
5	(8, 16)	0–7–3–11–6–8–16–15–14–17	1	$c_{6,8} + c_{8,16} + c_{16,15} - c_{6,15} = 37^*$	29	21	Integration	88
		0–7–3–11–6–8–15–16–14–17	2	$c_{6,8} + c_{8,15} + c_{15,16} + c_{16,14} - c_{6,15} - c_{15,14} = 42^*$				
		0–7–3–11–6–8–15–14–16–17	3	$c_{6,8} + c_{8,15} + c_{14,16} + c_{16,17} - c_{6,15} - c_{14,17} = 45^*$				
		0–7–3–11–6–15–8–16–14–17	4	$c_{15,8} + c_{8,16} + c_{16,14} - c_{15,14} = 38^*$				
		0–7–3–11–6–15–8–14–16–17	5	$c_{15,8} + c_{8,14} + c_{14,16} + c_{16,17} - c_{15,14} - c_{14,17} = 54^*$				
		0–7–3–11–6–15–14–8–16–17	6	$c_{14,8} + c_{8,16} + c_{16,17} - c_{14,17} = 39^*$				

Table 13 Comparison between the B&C and BRKGA-QL v2.0 algorithms

n	ζ (%)	BRKGA-QL v2.0				B&C				$\Delta\%\ UB$
		Best Z	Avg. Z	Best $T(s)$	$T(s)$	UB	LB	$GAP(\%)$	$T(s)$	
30	0	281.5	281.5	28.4	60.0	477.6	105.5	77.9	7202.1	−41.1
	25	258.0	260.6	24.9	60.0	443.2	105.5	76.1	7202.5	−41.8
	50	255.5	258.3	24.1	60.0	425.1	105.4	75.2	7201.9	−39.9
	75	239.8	240.9	24.2	60.0	382.9	106.0	72.3	7201.4	−37.4
	100	224.2	224.4	24.0	60.0	331.4	106.0	68.0	7204.9	−32.3
60	0	577.2	617.5	107.9	120.0	1801.0	179.2	90.1	7203.4	−68.0
	25	536.4	566.9	111.3	120.0	1638.5	178.5	89.1	7201.1	−67.3
	50	510.8	539.4	111.6	120.0	1483.8	178.4	88.0	7201.8	−65.6
	75	493.2	521.2	107.6	120.0	1349.7	177.3	86.8	7203.0	−63.5
	100	481.7	500.0	100.8	120.0	1219.6	179.7	85.3	7206.1	−60.5
90	0	855.9	918.7	148.1	180.0	2666.2	202.9	92.4	7200.2	−67.9
	25	778.8	851.7	159.5	180.0	2442.8	203.1	91.7	7200.1	−68.1
	50	736.8	815.0	161.5	180.0	2226.4	202.5	90.9	7200.2	−66.9
	75	690.7	753.5	157.5	180.0	1984.3	203.2	89.8	7200.3	−65.2
	100	660.4	717.3	175.6	180.0	1756.6	203.4	88.4	7200.2	−62.4
120	0	1480.0	1576.2	221.6	240.0	3454.0	225.6	93.5	7200.3	−57.2
	25	1436.1	1491.4	226.4	240.0	3199.7	226.1	92.9	7246.7	−55.1
	50	1333.8	1402.9	228.7	240.0	2934.3	226.0	92.3	7204.9	−54.5
	75	1266.3	1314.7	226.0	240.0	2651.7	226.0	91.5	7222.3	−52.2
	100	1178.9	1238.3	216.4	240.0	2375.7	227.6	90.4	7210.6	−50.4

Table 13 continued

n	ζ (%)	BRKGA-QL v2.0				B&C				$\Delta\%\ UB$
		Best Z	Avg. Z	Best $T(s)$	$T(s)$	UB	LB	$GAP(\%)$	$T(s)$	
155	0	2407.8	2569.9	292.6	310.0	4457.1	248.5	94.4	7201.6	−46.0
	25	2272.1	2385.3	297.6	310.0	4106.9	246.6	94.0	7201.6	−44.7
	50	2054.7	2189.2	308.9	310.0	3755.9	246.9	93.4	7262.2	−45.3
	75	1924.7	2017.7	294.8	310.0	3438.8	247.1	92.8	7299.6	−44.0
	100	1751.1	1889.3	292.4	310.0	3051.5	247.7	91.9	7200.3	−42.6
Avg.		987.5	1045.7	162.9	182.0	2162.2	192.2	87.6	7211.2	−53.6

valuable insights and real data. We thank the editor and the two reviewers for their constructive remarks and suggestions in an early version of the paper.

References

Andrade, C. E., Silva, T., & Pessoa, L. S. (2019). Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. *Expert Systems with Applications, 128*, 67–80.

Andrade, C. E., Toso, R. F., Gonçalves, J. F., & Resende, M. G. (2021). The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research, 289*(1), 17–30.

Aslaksen, I. E., Svanberg, E., Fagerholt, K., Johnsen, L. C., & Meisel, F. (2021). A combined dial-a-ride and fixed schedule ferry service for coastal cities. *Transportation Research Part A: Policy and Practice, 153*, 306–325.

Azadeh, S. S., Zee, J., & Wagenvoort, M. (2022). Choice-driven service network design for an integrated fixed line and demand responsive mobility system. *Transportation Research Part A: Policy and Practice, 166*, 557–574.

Beojone, C. V., & Geroliminis, N. (2021). On the inefficiency of ride-sourcing services towards urban congestion. *Transportation Research Part C: Emerging Technologies, 124*, 102890.

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

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

Chassaing, M., Duhamel, C., & Lacomme, P. (2016). An els-based approach with dynamic probabilities management in local search for the dial-a-ride problem. *Engineering Applications of Artificial Intelligence, 48*, 119–133.

Chaves, A. A., & Lorena, L. H. N. (2021). An adaptive and near parameter-free BRKGA using Q-learning method. In *2021 IEEE congress on evolutionary computation (CEC)* (pp. 2331–2338).

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

Chu, J.C.-Y.C., Chen, A. Y., & Shih, H.-H. (2022). Stochastic programming model for integrating bus network design and dial-a-ride scheduling. *Transportation Letters, 14*(3), 245–257.

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

Cordeau, J.-F., & Laporte, G. (2003). The dial-a-ride problem (DARP). *Variants, Modeling Issues and Algorithms, 1*(2), 89–101. 4OR.

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

Diana, M., & Dessouky, M. M. (2004). 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.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*(2), 201–213.

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

Gschwind, T., & Drexl, M. (2019). Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science, 53*(2), 480–491.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Technical report, Michigan: University of Michigan Press.

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

Karafotias, G., Eiben, A. E., & Hoogendoorn, M. (2014). Generic parameter control with reinforcement learning. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, (pp. 1319–1326).

Karafotias, G., Hoogendoorn, M., & Eiben, A. (2015). Evaluating reward definitions for parameter control. In *European conference on the applications of evolutionary computation* (pp. 667–680). Springer.

Kumar, P., & Khani, A. (2021). An algorithm for integrating peer-to-peer ridesharing and schedule-based transit system for first mile/last mile access. *Transportation Research Part C: Emerging Technologies, 122*, 102891.

Kuo, Y.-H., Leung, J. M. Y., & Yan, Y. (2023). Public transport for smart cities: Recent innovations and future challenges. *European Journal of Operational Research, 306*, 1001–1026.

Li, T., & Pei, Y. (2019). Chaotic evolution algorithms using opposition-based learning. In *2019 IEEE congress on evolutionary computation (CEC)* (pp. 3292–3299). IEEE.

Loshchilov, I., & Hutter, F.(2017). SGDR: Stochastic gradient descent with warm restarts. In *International conference on learning representations*.

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

Masmoudi, M. A., Braekers, K., Masmoudi, M., & Dammak, A. (2017). A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & Operations Research, 81*, 1–13.

Masmoudi, M. A., Hosny, M., Braekers, K., & Dammak, A. (2016). 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.

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

Molenbruch, Y., Braekers, K., & Caris, A. (2017). Benefits of horizontal cooperation in dial-a-ride services. *Transportation Research Part E: Logistics and Transportation Review, 107*, 97–119.

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

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

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

Posada, M., & Häll, C. H. (2020). A metaheuristic for evaluation of an integrated special transport service. *International Journal of Urban Sciences, 24*(3), 316–338.

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

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

Rey, D., & Neuhäuser, M. (2011). In: M. Lovric (ed.) *Wilcoxon-Signed-Rank Test* (pp. 1658–1659). Springer.

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

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

Samma, H., Mohamad-Saleh, J., Suandi, S. A., & Lahasan, B. (2020). Q-learning-based simulated annealing algorithm for constrained engineering design problems. *Neural Computing and Applications, 32*(9), 5147–5161.

Schenekemberg, C. M., Chaves, A. A., Coelho, L. C., Guimarães, T. A., & Avelino, G. G. (2022). The dial-a-ride problem with private fleet and common carrier. *Computers & Operations Research, 147*, 1–14.

Shiraki, H., Matsumoto, K., Shigetomi, Y., Ehara, T., Ochi, Y., & Ogawa, Y. (2020). Factors affecting CO_2 emissions from private automobiles in Japan: The impact of vehicle occupancy. *Applied Energy, 259*, 114196.

Silva, T. T., Chaves, A. A., Yanasse, H. H., & Luna, H. P. L. (2019). The multicommodity traveling salesman problem with priority prizes: A mathematical model and metaheuristics. *Computational and Applied Mathematics, 38*(4), 1–25.

Spears, W. M., & Jong, K. A. D. (1991). On the virtues of parametrized uniform crossover. In *Proc. of the fourth international conference on genetic algorithms* (pp. 230–236).

Watkins, C. J. C. H. (1989). Learning from delayed rewards. PhD thesis, King's College, University of Cambridge.