CIRRELT

CIRRELT-2022-32

Hybrid Metaheuristic for an Integrated Dial-a-Ride Problem with Common Carrier and Public Transportation

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

November 2022

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2022-009.

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

Hybrid Metaheuristic for an Integrated Dial-a-Ride Problem with Common Carrier and Public Transportation

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

1. Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil, Aeronautics Institute of Technology (ITA), São José dos Campos, Brazil
2. Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil
3. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Canada Research Chair in Integrated Logistics
4. Research Group of Technology Applied to Optimization (GTAO), Federal University of Paraná (UFPR), Curitiba, Brazil

Abstract. Dial-a-ride operations consist of door-to-door transportation systems, generally 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 CO2. 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 by considering the integration with common carriers and public transportation, which we call DARP-CCPT. Requests can be served either 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. This entire operation needs to be synchronized with the bus schedules. As a methodology, we solve the DARP-CCPT with a hybrid of metaheuristics and machine learning by combining a biased random-key genetic algorithm with the Q-Learning method (BRKGA-QL) and local search heuristics. This paper introduces a new version of this method, called BRKGA-QL v2.0, particularly improving the population quality and diversity thanks to a new mutation method in the classical crossover operator. Computational experiments on a new benchmark data set with realistic data from Québec City show that BRKGA-QL v2.0 outperforms its previous version. In addition, we provide a qualitative analysis for the DARP-CCPT, showing that the middle mile integration with public transportation can save up to 20% in operating costs, besides reducing the traveled distances of private vehicles and common carriers.

Keywords: dial-a-ride problem, integration, public transportation, common carrier, metaheuristic.

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 312747/2021-7 and 405702/2021-3. Leandro C. Coelho was supported by the Natural Sciences and Engineering Council of Canada (NSERC) under grant 2019-00094. We also thank Compute Canada for providing parallel computing facilities.

---

* Corresponding author: cledercms@hotmail.com

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2022

1. Introduction

Dial-a-ride (DAR) problems (DARP) aim to offer responsive door-to-door transportation services, generally for users with some mobility impairment. 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 and Laporte, 2003b). 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; Cordeau, 2006; Brevet et al., 2019; Cordeau and Laporte, 2003a; Diana and Dessouky, 2004; Masmoudi et al., 2016, 2018; Qu and Bard, 2015; Ropke and Cordeau, 2009; Ropke et al., 2007). The survey of Ho et al. (2018) provides a review and recent developments.

Due to the flexibility required, a DAR operation is high-costly and the vehicles usually travel with low occupancy. Besides the low operational efficiency, the DAR is also environmentally inefficient, given the associated high levels of CO_2 emissions (Shiraki et al., 2020). 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). On the other hand, an integrated coordinated multi-modal public transportation system leads to a substantial increase in operational complexity since they must offer wide coverage, real-time response to demand, quality service, and energy-efficient operations (Kuo et al., 2022).

From the relevant literature, Chu et al. (2022) point out that designing efficient fixed bus routes in remote areas is difficult because demand is unstable and scattered. To improve the service level, the authors propose integrated planning of a fixed bus route with the DAR transportation. 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 the integration of bus and DAR operations considerably reduces unserved demands and improve the service level.

Addressing the first mile/last mile (FMLM) problem, Kumar and Khani (2021) integrate peer-to-peer ridesharing with a schedule-based transit system. 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.

Aslaksen et al. (2021) investigate the integration of DAR 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 performed for a case study of Kiel, Germany, 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.

Despite promoting better operational and environmental efficiency, the integration of DAR and public transportation causes a reduction in the quality of service, as the user has to carry out multiple hop-on hop-off, in addition to being subject to possible delays in buses and trains. 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 by 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 BRKGA-QL is also designed, where the algorithm parameters are optimized by the reinforcement learning method. BRKGA-QL successfully solved both the DARP classical instances and DARP-PFCC instances. The study also provided a real case study showing the benefits for both the provider and the users.

In order to incorporate better operational and environmental performance from public transportation and promote a better quality of service, this paper integrates common carriers and public transportation into the DARP framework, which we call DARP-CCPT. Requests can be served either 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. This entire operation needs to be synchronized with the bus schedules. To solve this novel problem variant, we introduce BRKGA-QL v2.0, with significant improvements on the population management. 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 new DARP 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. 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;
- design a BRKGA-QL v2.0 algorithm to solve the DARP-CCPT, 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;
- 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;
- generate a new instance set for the new problem, with all the instances and results available online;
- apply our BRKGA-QL v2.0 to solve real-case scenarios of the DARP-CCPT, 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 formally describes the DARP-CCPT and details the synchronization procedure with public transportation. The BRKGA-QL v2.0 is presented in Section 3, while its encoding, decoding, and local search strategies are described in Section 4. In Section 5, we present the instances generator and the results of extensive computational experiments performed. Our conclusions and directions for future works follow in Section 6.

2. Problem definition

The DARP-CCPT is defined as follows. Let n be the number of requests from a set of pickup nodes $\mathcal{P} = \{1, ..., n\}$ to a set of delivery nodes $\mathcal{D} = \{n+1, ..., 2n\}$. 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

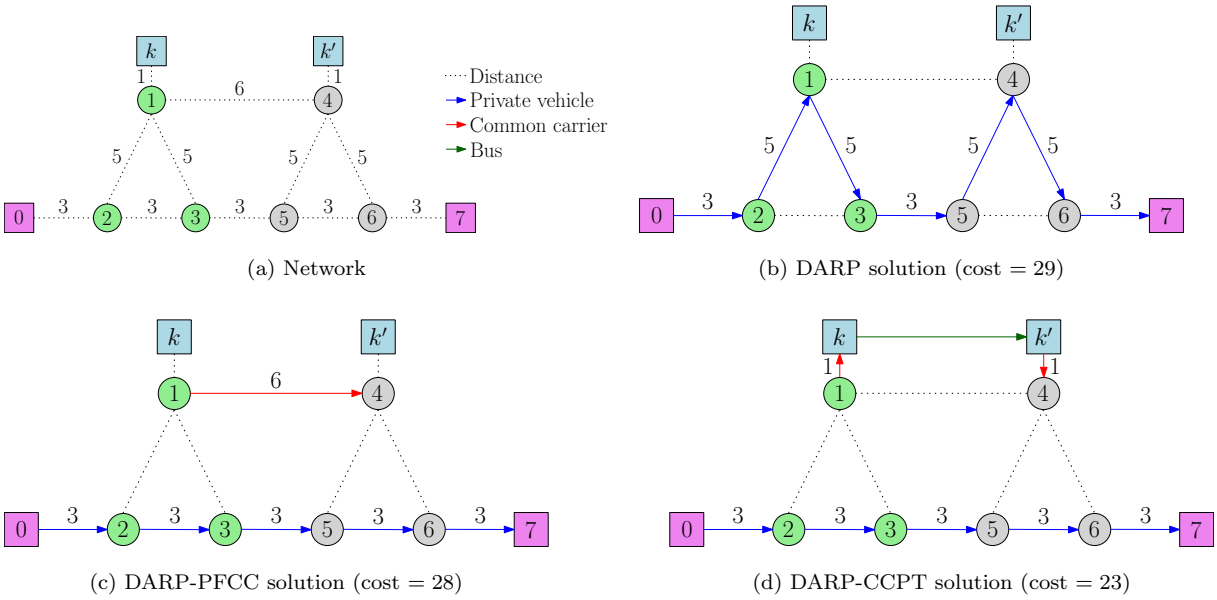
either a private vehicle, a common carrier vehicle, or by integrating common carriers and buses. A limited fleet of m private vehicles of capacity C is available. All private vehicles depart from the depot 0, visit a set of pickup and delivery nodes, and return to the depot $2n + 1$, without violating the maximum route duration T. In this transportation option, users share the same vehicle, which travels among pickup and delivery nodes according to a scheduling plan. 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. Here, 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.

We consider the following transportation cost scheme. For private vehicles, a routing cost c_{ij} is multiplied by the distance d_{ij} 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 $f + v \cdot d_{in+i}$. 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 $f + v \cdot d_{ik}$; 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 $f + v \cdot d_{k'n+i}$.

As in the DARP, each node $i \in \mathcal{P} \cup \mathcal{D}$ is associated with a demand q_i, a service duration d_i, a maximum ride time L_i, and a time window $[e_i, l_i]$, with e_i and l_i being the earliest and latest times to serve i, respectively. For nodes 0 and $2n+1$, the time windows $[e_0, l_0]$ and $[e_{2n+1}, l_{2n+1}]$ define the working horizon for vehicles of the private fleet. Finally, t_{ij} represents the travel time from i to j, and we also consider $d_0 = d_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$, and $q_i > 0, q_i = -q_{n+i}, i \in \mathcal{P}$, and $n + i \in \mathcal{D}$. In order to define a feasible solution for the DARP-CCPT, we also assume all operational constraints of the DARP, such as pairing, precedence, capacity, time windows, route connectivity, and ride time. A solution to the DARP-CCPT minimizes the total costs while determining whether users are served by the private fleet, common carrier, or public transportation.

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 of each arc. In Figure 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, in which a common carrier satisfies request 1, while Figure 1d presents a solution for the DARP-CCPT 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.



(a) Network

(b) DARP solution (cost = 29)

(c) DARP-PFCC solution (cost = 28)

(d) DARP-CCPT solution (cost = 23)

Figure 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).

We consider all available bus lines, timetables, and bus stops in our DARP-CCPT. 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 minutes of the pickup/delivery node. This is done to balance the common carrier costs and bus ticket costs. 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.

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

In the example of Figure 2, we consider two bus lines (A and B) with two available timetables ($t = 1, 2$). 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''). For line B we have 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 minutes. The user has a waiting time upon arrival at the delivery node 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 minutes 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 minutes 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 all options for each request, we select the one with the lowest cost. In the example of Figure 2, the third option has the lowest cost. 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.

3. Biased random-key genetic algorithm with *Q*-Learning

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

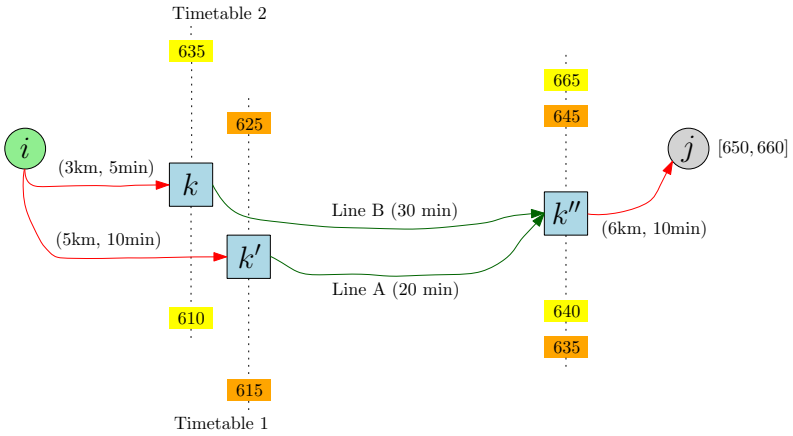


Figure 2: Example of the integration between request nodes and bus lines. The pickup node i is represented in green, the delivery node $j = n + i$ in grey, and bus stops k, k', and k'' in blue. The colored arcs indicate the path of each vehicle type (common carrier in red, bus in green), and timetables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

BRKGA represents a solution by a random-key vector with dimension n ($x \in [0, 1]^n$), 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 and 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 the Bellman equation. The agent uses the parameters ϵ, learning factor (l_f), and discount factor (d_f) 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 version 2.0, with improvements to perform efficient parameter control and balance intensification and diversification during the search. BRKGA-QL v2.0 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 $[s_i, a_j]$. 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.

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 the adaptive parameter setting of the Q-Learning during the evolutionary process aiming to balance exploration and exploitation in the agent, the chaotic individuals generated from the communities, and mainly the mutation operator in the PUX instead of mutants. This new version of the algorithm is compared against the previous one in Section 5.

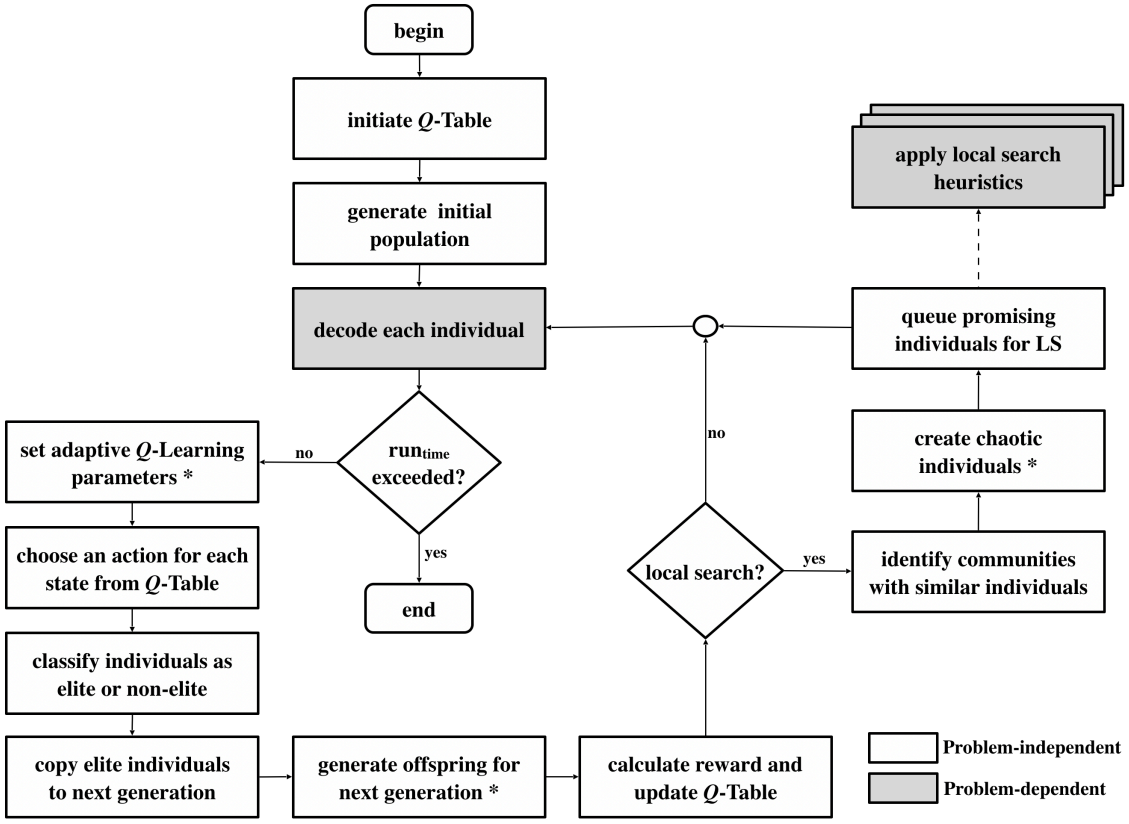


Figure 3: View of the BRKGA-QL algorithm version 2.0. Components marked with an $*$ are new in this version.

BRKGA-QL v2.0 starts by initializing the Q-Table values at 0. Table 1 shows the action list of each parameter. There are four parameters (states): population size (p), elite set size (p_e), mutation probability (p_m), and probability that an offspring inherits the gene from its elite parent (ρ_e). The parameter values are the ones proposed by Chaves and Lorena (2021), except for the mutation probability, which is a new feature of this version.

The second step consists of generating the BRKGA-QL population (P) with p random-key vectors. Each vector position is generated randomly in the real interval $[0, 1]$. The value of p

Table 1: Action list of the BRKGA parameters based on Chaves and Lorena (2021)

Parameter	Action List
p	{233, 377, 610, 987, 1597, 2584}
p_e	{0.10, 0.15, 0.20, 0.25, 0.30}
p_m	{0.0, 0.01, 0.02, 0.03, 0.04, 0.05}
ρ_e	{0.55, 0.60, 0.65, 0.70, 0.75, 0.80}

is set as the largest population size defined in the Q-Table. In each generation, the population can be reduced or expanded. The first case performs a proportional pruning in the elite and non-elite solutions. The second case generates new solutions with the PUX.

Different decoders map the solution space in different ways. In the BRKGA-QL it is possible to implement k decoders. A random-key in position $n+1$ defines the decoder of each solution. We apply the chosen decoder and calculate the fitness of the solution.

Unlike the first version of the BRKGA-QL, in this version, the Q-Learning parameters are set during the evolutionary process by deterministic rules. We adopt a cosine annealing decay (equation (2)) for the value of ϵ (Loshchilov and Hutter, 2017). The value of l_f is updated following equation (3), and d_f is set to 0.8 looking for a higher, long-term reward (Samma et al., 2020).

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

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

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 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 PUX operator using ρ_e. However, a mutation can occur during the selection of genes with a probability p_m. This characteristic allows the method to have diversity and generate good solutions. In previous versions, mutants (random solutions) are generated, but these are unlikely to have good fitness. The new solutions 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 the following 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 equation (5):

$$\Delta_f = \frac{1}{p}\left(\frac{f_b^{t-1}}{f_b^t} - 1\right) \times 10^5. \tag{5}$$

The reward function combines the improvement of the current best fitness and binary rewards proposed by Karafotias et al. (2015). This function is less scale insensitive than the binary reward, including the magnitude of the improvement and the population size.

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).

A local search module (Schenekemberg et al., 2022) is also used in the BRKGA-QL v2.0. We apply the Label Propagation method (Raghavan et al., 2007) on the elite set whenever BRKGA-QL finds a new best solution ($R > 1$) or when the number of generations without improvement is greater than n. This method groups the elite solutions using a weighted graph through the Pearson correlation measure. The best chromosome from each cluster is sent to the local search. We do not add the local optima in the population. The Random Variable Neighborhood Search (RVND) (Penna et al., 2013) is used to select the neighborhood heuristic order to be applied in each iteration. We also increase the diversity of the elite set by generating chaotic individuals

(Li and Pei, 2019) using PUX with mutants (random solutions) and similar chromosomes in the elite set. The chaotic individuals replace these similar individuals in the population.

The BRKGA-QL v2.0 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.

4. BRKGA-QL for the DARP-CCPT

In this section, we show how solutions are encoded to a vector of random keys and how they are decoded from a vector of random keys into a DARP-CCPT solution. We also present some local search procedures to polish promising solutions.

4.1. Encoding a solution to a vector of random keys

A DARP-CCPT solution is composed of 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 having values in the range $[0, 1]$ to represent a solution. Each gene $i \in \{1, ..., n\}$ 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-CCPT solution space. Next, we present all decoders implemented in our paper.

4.2. 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 DARP-CCPT instances. A decoder function is chosen according to the value of gene $n+1$ to map a random-key vector to the DARP-CCPT 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 $[75,\ 100)$ select the fourth decoder. The decoders used to solve DARP-CCPT are:

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

Δ'_ν are scheduled one by one in $\mathcal{R}_\nu$, 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 $\mathcal{R}_\nu$, considering only the partial path formed from the previously inserted pickup i to the depot $2n+1$. The cheapest cost for vehicle ν 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 ν avoids planning a complete schedule for the entire route. That is, it is enough to evaluate the vicinity of the insertion and all nodes on the partial path leading to the depot $2n+1$.

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 ν. 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 maximize the usage of buses and vehicles of the private fleet. It works as **Decoder 3**, except that a request is served by an outsourced vehicle only if a feasible solution cannot be found using a private vehicle or an integration of a common carrier with a bus. On the contrary, 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, although all decoders are invoked periodically throughout the optimization process, generating different sets of solutions.

4.3. Local search procedures

We implemented eight local search procedures for the DARP-CCPT based on the heuristics developed for the DARP-PFCC by Schenekemberg et al. (2022). All procedures are embedded into the framework described in Section 3 and applied with parallel processing techniques, improving promising solutions without compromising the evolutionary process of the BRKGA-QL v2.0. These heuristics are:

1. **Exchange a request to another private vehicle:** This heuristic removes at random 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 heuristic randomly 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 $\nu_R = [0, ..., u, v, w, z, ..., 2n+1]$. Then, all 24 possible permutations in the partial path are computed, and the most economical among the feasible ones is selected. Performing swaps between nodes next to the route endpoints may not be interesting, as precedence, ride-time, and time windows constraints are frequently infeasible. This observation justifies using partial paths with four consecutive nodes;
3. **Swap two requests inter-route:** This heuristic 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 heuristic swaps partial paths between two randomly selected vehicles from the private fleet. In this case, partial paths are natural sequences (see Parragh et al. (2010)) involving arcs where the load of the vehicle is null;
5. **Remove 1/3**: This heuristic 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 heuristic 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 heuristic 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 heuristic 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.

5. 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 new version of 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. To allow for comparisons between the two algorithms, we adapted their code and used all decoder functions and local search procedures presented in Section 4. The algorithms were implemented in C++ and run on computers with Intel(R) Xeon(R) E5-2683 v4 @ 2.10GHz CPU.

All data sets and computational results are available at `https://tobeinsertedafterreview` and the source code (problem-independent) of BRKGA-QL v2.0 is available at `https://tobe.insertedafterreview`.

5.1. Instances generator

To evaluate our algorithm, we use a real data set from a transportation provider that operates a DAR service in Québec, Canada. For confidentiality reasons, we slightly modify the user information, so that it is not possible to determine the original pickup and delivery addresses.

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, while $\zeta = 100\%$ represents a fully flexible scenario, allowing all requests to be integrated. We emphasize that when $\zeta = 0\%$ a DARP-PFCC instance is obtained. Finally, values $25\%, 50\%$, and 75%

represent moderate scenarios. These scenarios are justified by noting that in real cases the use of public transportation may be prohibitive for some users.

The DAR 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) $i = 1, ..., n/2$ and inbound requests (return trip to home) $i = n/2 + 1, ..., n$. 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 (28800 seconds) and finish before 4 p.m (57600 seconds).

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 moderate 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 $v = 2$.

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	28800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
60	2	28800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
90	3	28800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
120	3	28800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$
155	3	28800	5400	1	120	$[T, 2 \cdot T - 2700]$	$e_i + 2700$	$l_{n+i} - 2700$	$[T + 2700, 2 \cdot T - 900]$

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.

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

Our experiments were performed adopting $run_{time} = 2 \cdot n$ as the stopping criterion for the two algorithms. The run_{time} parameter denotes the time in seconds for each run and is the only

parameter that needs 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 "Average Z" show the values of the best and average solutions, respectively. Both versions use the maximum running time (run_{time}) as a stopping criterion, thus, the "Best Time" column shows the average computational time for the algorithm to find the best solution in each run. The "Δ" column presents the relative percentage deviation of the best solution obtained by both versions. Negative values indicate that version 2.0 found better solutions. We can 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 Section 2 was fundamental in supporting this computational time stability.

Figure 4 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 5 presents the performance profile (Dolan and Moré, 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 in terms of computational time and level of 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 BRKGA-QL found the target only in 26%. We also performed statistical analysis with the Wilcoxon signed-rank test (Rey and Neuhauser, 2014), proving a significant 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.

Table 3: Computational results of the BRKGA-QL and BRKGA-QL v2.0 applied to solve DARP-CCPT.

n	Integration ζ (%)	BRKGA-QL			BRKGA-QL v2.0			Δ (%)
		Best Z	Average Z	Best Time	Best Z	Average Z	Best Time	
30	0	281.50	282.76	50.52	281.50	281.53	28.42	0.00
	25	257.02	260.95	38.79	258.04	260.56	24.94	0.42
	50	257.11	259.73	34.50	255.54	258.31	24.14	-0.59
	75	239.76	241.65	35.10	239.84	240.91	24.20	0.04
	100	224.22	224.81	33.55	224.22	224.40	23.96	0.00
60	0	633.27	670.24	97.24	577.22	617.51	107.93	-8.85
	25	593.03	623.64	99.26	536.39	566.89	111.28	-9.48
	50	558.24	584.61	98.10	510.83	539.40	111.62	-8.51
	75	535.91	563.05	94.15	493.21	521.16	107.61	-7.94
	100	529.36	547.15	101.40	481.70	499.97	100.82	-9.00
90	0	941.39	977.06	129.54	855.86	918.73	148.07	-9.09
	25	881.72	923.15	134.13	778.76	851.75	159.50	-11.68
	50	827.84	875.29	136.67	736.83	814.97	161.53	-10.98
	75	770.39	807.59	144.74	690.67	753.49	157.54	-10.25
	100	701.03	759.54	137.81	660.41	717.30	175.62	-5.79
120	0	1617.92	1675.18	186.79	1480.04	1576.25	221.61	-8.52
	25	1471.02	1545.24	205.61	1436.10	1491.39	226.41	-2.30
	50	1411.16	1464.07	197.07	1333.82	1402.93	228.71	-5.48
	75	1294.00	1361.58	182.19	1266.30	1314.65	226.04	-2.08
	100	1218.69	1279.57	211.08	1178.86	1238.29	216.35	-3.27
155	0	2600.56	2704.95	292.08	2407.82	2569.92	292.55	-7.41
	25	2385.11	2468.15	273.74	2272.08	2385.28	297.59	-4.74
	50	2201.34	2289.52	282.48	2054.71	2189.23	308.87	-6.65
	75	2032.72	2099.35	253.62	1924.67	2017.74	294.83	-5.28
	100	1871.35	1959.01	277.77	1751.11	1889.27	292.35	-6.43

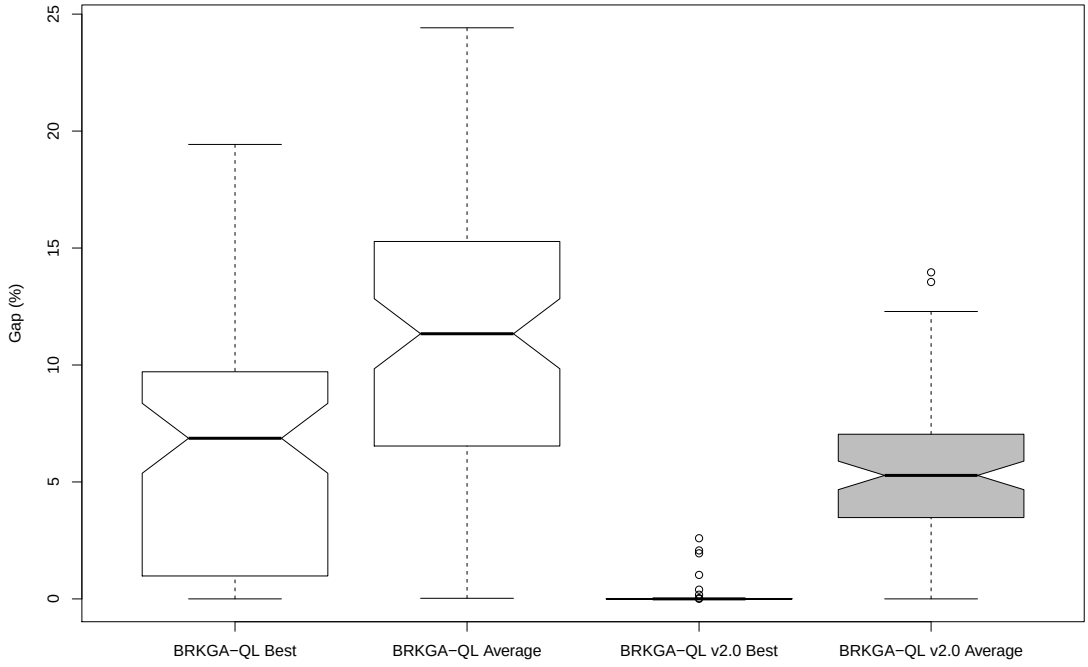


Figure 4: Box plot of the gap between the best and average solutions of BRKGA-QL and BRKGA-QL v2.0 to the best-known solutions.

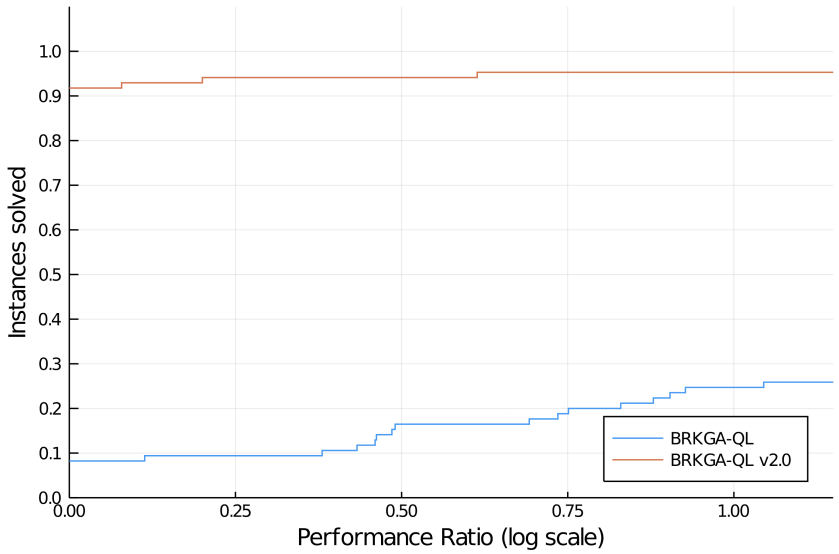


Figure 5: Performance profile of runtime for BRKGA-QL and BRKGA-QL v2.0.

5.3. Qualitative analysis of the BRKGA-QL v2.0 results

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 Total column shows the total cost to fulfill all requests. The columns $\Delta\%$ Distance and $\Delta\%$ $UB_{0\%}$ show the variations in distance and costs concerning the solution without integration with public transportation ($\zeta = 0\%$). 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 the decrease in the 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 6 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 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 very 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 vehicles of the private fleet.

In Figure 7, 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

Table 4: Comparison of route cost distribution.

n	Integration ζ (%)	Cost components				Total	Δ% Distance	Δ% $UB_{0\%}$
		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
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

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.

6. 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 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

Table 5: Time components of integrated trips.

Integration	**Percentage of time**					**Ride**	**Direct**
ζ (%)	**First mile**	**Middle mile**	**Last mile**	**Service**	**Waiting**	**time**	**trip**
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

Table 6: Distance components of integrated trips.

Integration	**Percentage of distance**			**Distance**	**Direct**
ζ (%)	**First mile**	**Middle mile**	**Last mile**	**total**	**trip**
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

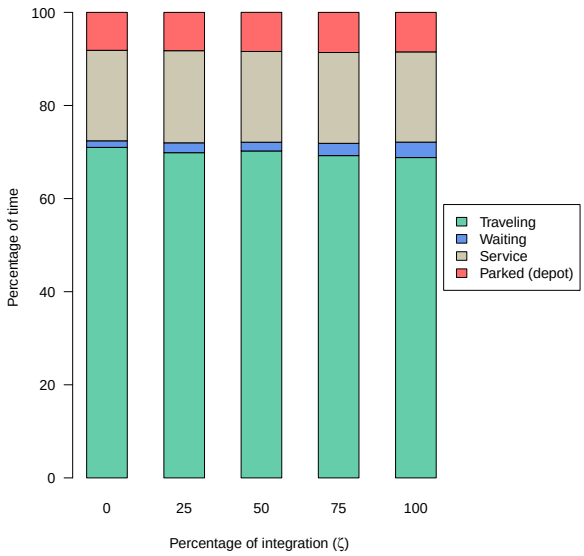


Figure 6: Private fleet operating time breakdown.

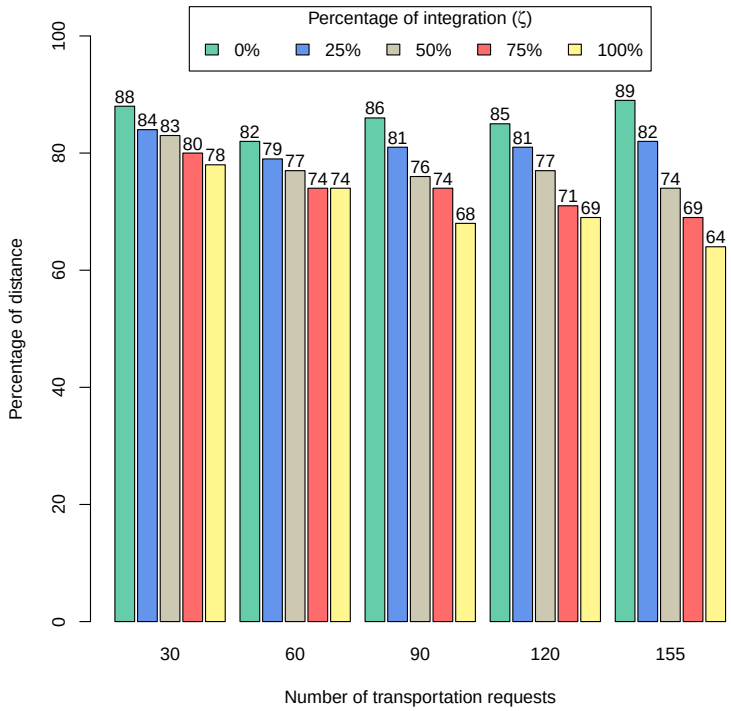


Figure 7: Percentage of distance traveled relative to the worst-case scenario.

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 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.

To the best of our knowledge, this is the first paper that considers the optimized use of private fleets integrated with common carriers and public transportation. Computational experiments have shown the advantages of this integration in terms of cost savings while maintaining the service quality for users. Another important issue is the possible savings in CO_2 emissions, considering that the first and last miles vehicles travel shorter distances, 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 studying the use of the private fleet and car-sharing in the first and last miles, as well as the use of other types of transport such as electric mini cars, scooters, and bicycles.

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, 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.

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, 89–107.

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.

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.

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, 245–257.

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

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

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

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, 539–557.

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

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

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.

Holland, J.H., 1975. Adaptation in natural and artificial systems. Technical Report. Michigan: University of Michigan Press.

Karafotias, G., Hoogendoorn, M., Eiben, A., 2015. Evaluating reward definitions for parameter control, in: European Conference on the Applications of Evolutionary Computation, Springer. pp. 667–680.

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., 2022. Public transport for smart cities: Recent innovations and future challenges. European Journal of Operational Research forthcoming.

Li, T., Pei, Y., 2019. Chaotic evolution algorithms using opposition-based learning, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 3292–3299.

Loshchilov, I., Hutter, F., 2017. SGDR: Stochastic gradient descent with warm restarts, in: Proceedings of the International Conference on Learning Representations (ICLR), preprint arXiv:1608.03983.

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.

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

Penna, P., Subramanian, A., Ochi, L., 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, 217–241.

Posada, M., Häll, C.H., 2020. A metaheuristic for evaluation of an integrated special transport service. International Journal of Urban Sciences 24, 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, 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, 036106.

Rey, D., Neuhauser, M., 2014. Wilcoxon-signed-rank test, in: Lovric, M. (Ed.), International Encyclopedia of Statistical Science. Springer Berlin Heidelberg, pp. 1658–1659.

Ropke, S., Cordeau, J.F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43, 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, 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, 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 co2 emissions from private automobiles in japan: The impact of vehicle occupancy. Applied Energy 259, 114196.

da 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, 1–25.

Spears, W.M., Jong, K.A.D., 1991. On the virtues of parameterized uniform crossover. Proc. of the Fourth International Conference on Genetic Algorithms , 230–236.

Watkins, C.J.C.H., 1989. Learning from delayed rewards. Ph.D. thesis. King's College, University of Cambridge.