Transportation Research Part A 178 (2023) 103862

Contents lists available at ScienceDirect

Transportation Research Part A

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

# A data-driven game theoretic multi-objective hybrid algorithm for the Dial-A-Ride Problem with multiple time windows

Slim Belhaiza [a,*], Rym M'Hallah [b], Munirah Al-Qarni [c]

[a] Department of Mathematics and IRC for Smart Logistics and Mobility, College of Computing & Mathematics, King Fahd University of Petroleum & Minerals, Dhahran, Kingdom of Saudi Arabia

[b] Engineering Department, Faculty of Natural, Mathematical and Engineering Sciences, Kings College, London, UK

[c] Department of Mathematics, KFUPM, Dhahran, Kingdom of Saudi Arabia

ARTICLE INFO

*Keywords:*
Dial-a-ride
Neural network
Game theory
Multiple-objective
Adaptive large neighborhood search
Time windows

ABSTRACT

The Dial-A-Ride Problem (DARP) designs pick-up and delivery routes for a set of customers. It arises in door-to-door transport services tailored to elderly and impaired people. It minimizes operational costs while accommodating as many drivers' and customers' constraints as possible; e.g., constraints on transit time and time windows on pick up and drop off.

This paper develops a two-layer and a three-layer artificial neural network (ANN) to predict DARP parameters. The ANN is trained with real data provided by a transit agency from the Canadian city/region of Vancouver. Experimental results show that a three-layer ANN with rectified linear activation functions in conjunction with a stochastic gradient descent optimizer provides the most accurate output forecasts.

Utilizing the predictions as parameter estimates for a deterministic mean DARP with multiple time windows (DARPMTW), this paper models the problem as a Company-Drivers-Customers' theoretic game that minimizes the total route duration and maximizes the minimal satisfaction of the drivers and the customers. It shows that non-dominated Pareto feasible solutions satisfy the Nash equilibrium conditions of the game-theoretic model. It then creates a multi-objective hybrid adaptive large neighborhood search (DD-HALNS) to estimate the Pareto front. The multi-objective DD-HALNS algorithm confines its search to feasible solutions that satisfy the Nash equilibrium conditions and employs local search operators based on their success rates. The operators are controlled by a learning mechanism that utilizes previous successful moves, cost savings, and utility maximization. Four hybridization operators are applied, namely Simulated Annealing, Tabu Lists, Genetic Crossover, and Restarts. The results of experiments conducted on real-world DARPMTW instances demonstrate the ability of the multi-objective DD-HALNS to enhance the best-known routing solutions, as well as its implementability.

## 1. Introduction

The subclass of Vehicle Routing Problems (VRP) known as the dial-a-ride problem (DARP) pertains to door-to-door transport for seniors and individuals with special needs. The aim is to find a minimal-cost routing plan that meets customers' requirements and satisfies ideal constraints such as maximum transit time, vehicle capacity, and route duration. DARPs can be either static or dynamic, with the former referring to situations where all requests are known at the time of scheduling, and the latter referring to scenarios where requests are received during the execution of a basic routing plan. Fluctuating traffic and road conditions introduce

* Corresponding author.
*E-mail addresses:* slimb@kfupm.edu.sa (S. Belhaiza), rym.mhallah@kcl.ac.uk (R. M'Hallah), g201901470@kfupm.edu.sa (M. Al-Qarni).

https://doi.org/10.1016/j.tra.2023.103862

Available online 24 October 2023

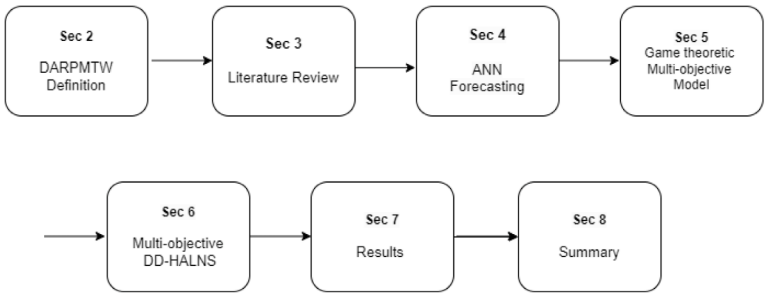


**Fig. 1.** Paper workflow diagram.

uncertainty to static DARP parameters, which further complicates the creation of efficient routing plans. To address this uncertainty, the current paper employs two- and three-layer artificial neural networks (ANN) to forecast travel times and service times.

Numerous competitive heuristics address DARP. However, their real-life implementation is limited because they neither address customers' constraints nor the companies' contractual agreements with their drivers. To implement solutions of existing heuristics, a company usually (i) makes a customer's preset time window (TW) a soft constraint, i.e. it offers its customers online and mobile tracking of their serving vehicle in lieu of abiding by the TWs, and (ii) compensates its drivers when their daily workloads are unbalanced or unfairly distributed. While imposing that a customer's pick-up and drop operations occur on the same route and during one of a customer's TWs enhances the implementability of solutions, it makes the resulting problem harder to solve. To overcome the limitations of existing heuristics, this paper proposes a game theoretic approach that minimizes a company's operating costs while maximizing the minimum levels of satisfaction of drivers and customers. That is, it models DARP with multiple TWs (MTW) as a three-objective game theoretic problem, denoted DARPMTW. The first objective minimizes the company's total route duration, expressed in time units. The second maximizes the minimum drivers' utility, which translates his/her satisfaction with the company's compliance to a contractual agreement on maximum loading capacity, scheduling sequence, and route duration. The third maximizes the minimum customer satisfaction with service quality.

The three-objective game theoretic model can be solved exactly for instances with a few customers and wide TWs. For large instances with narrow TWs, this paper proposes a multi-objective data-driven hybrid adaptive large neighborhood search (DD-HALNS). Multi-objective DD-HALNS selects its operators according to their updated success rates. It applies them according to a learning mechanism that examines previous successful moves, cost savings, and utility maximization parameters to intelligently guide the search. It applies four hybridization techniques: Simulated Annealing (SA), Tabu Lists, Genetic Crossover, and Restarting.

This paper brings four innovative contributions.

- It improves the implementability of DARP solutions by requiring that each customer be picked up and dropped during one of *several customer-defined TWs*. To our best knowledge, the MTW constraints were not considered in DARP. Experimental results suggest that accounting for MTWs improves routing plans both quantitatively and qualitatively.
- It offers a *new three-agent game model* that simultaneously minimizes company costs and maximizes driver and customer satisfaction. It shows that any Pareto non-dominated Nash equilibrium of the proposed game model is a DARPMTW feasible solution and vice versa.
- It uses predicted travel times and service times; all obtained using two- and three-layer ANNs, trained with real data provided by a transit agency from the Canadian city/region of Vancouver. It provides computational evidence that a three-layer ANN with rectified linear activation functions (relu) in conjunction with a stochastic gradient descent optimizer (SGD) provides the most accurate point estimates.
- It approximately solves the multi-objective mean value DARPMTW using DD-HALNS, which guides the search toward feasible regions of the search space, and directs heuristic selection using accumulated information on their success rate in identifying improving solutions.

The structure of the paper is illustrated by the workflow diagram in Fig. 1. Section 2 defines the problem. Section 3 provides some background on DARP, enumerates applications of game theoretic and multiple-objective DARP models, reviews the MTW literature, and enumerates ALNS-based heuristics used for DARP. Section 4 details the proposed ANN architecture for travel times and service times forecasting. Section 5 presents the game theoretic multiple-objective model, and lays the foundation of its optimization algorithm. Section 6 introduces the multi-objective DD-HALNS, and describes how it accounts for the three objectives. Section 7 analyzes multi-objective DD-HALNS' results on two sets of real-life instances and new benchmark instances. Finally, Section 8 provides a summary of the paper.

## 2. DARPMTW

DARPMTW is a pick-up and delivery routing problem with MTWs and maximum load, duration, and transit-time constraints. It involves a set $\mathcal{K}$ of $m$ drivers and a set $\mathcal{I}$ of $n$ customers. Driver $k \in \mathcal{K}$ starts and ends a route at a single common depot $D$. Because

**Table 1**
DARPMTW notation.

| General | Type | Description |
|---|---|---|
| $\mathcal{I}$ | Set | Set of all customers; |
| $\mathcal{N}$ | Set | Set of all pick-up and drop nodes; |
| $\mathcal{K}$ | Set | Set of all vehicles; |
| $\{D\}$ | Set | Set of depots; |
| $\mathcal{V}$ | Set | Vertex set $= N \cup \{D\}$; |
| $\mathcal{A}$ | Set | Arc set of links between pairs of nodes in $\mathcal{V}$; |
| $\mathcal{G}$ | Graph | DARPMTW graph; |
| **Customers** | **Type** | **Description** |
| $n$ | Integer | Number of customers; |
| $i$, $j$ | Index | Customer $i$ or $j$; |
| $i^+$, $i^-$ | Node | Pick-up and drop of customer $i$; |
| $i_\cdot$, $j_\cdot$ | Node | Pick-up or drop of customer $i$ or $j$, or depot $D$; |
| $s_{i_\cdot}$ | Real | Service time at $i_\cdot$; |
| $P_{i_\cdot}$ | Set | Set of time windows at $i_\cdot$; |
| $w_{i_\cdot}$ | Integer | Number of time windows at $i_\cdot$; |
| $l_{i_\cdot}^p$ | Real | Lower bound of time window $p$ at $i_\cdot$; |
| $u_{i_\cdot}^p$ | Real | Upper bound of time window $p$ at $i_\cdot$; |
| $d_{i_\cdot}$ | Integer | Number of customers to serve at $i_\cdot$; |
| $\tau_i$ | Real | Maximum transit time of customer $i$; |
| **Drivers** | **Type** | **Description** |
| $m$ | Integer | Number of drivers/vehicles/routes; |
| $k$ | Index | Driver or vehicle $k$; |
| $D$ | Node | Depot; |
| $l_k$, $u_k$ | Real | Lower and upper bound of the length of driver $k$'s work shift; |
| $L_k$ | Real | Loading capacity of vehicle $k$; |
| $T_k$ | Real | Maximum route duration of vehicle $k$; |

each customer is both dropped and picked, the set of $n$ customers defines the set $\mathcal{N}$ of $2n$ nodes. The vertex set $\mathcal{V} = \{D\} \cup \mathcal{N}$ is linked through the arc set $\mathcal{A}$. Thus, $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is the graph defining DARPMTW.

A customer $i \in \mathcal{I}$ must be picked up ($i_\cdot = i^+$) and dropped ($i_\cdot = i^-$) by the same vehicle, and must have a maximum transit time $\tau_i$; i.e., transfers from one vehicle to another are not allowed. Request $i_\cdot \in \mathcal{N}$ of customer $i$ requires a service time $s_{i_\cdot} > 0$, and a demand $d_{i_\cdot}$, where $d_{i^+} > 0$ and $d_{i^-} < 0$; usually $d_{i^+} = 1$ and $d_{i^-} = -1$. Service for $i_\cdot$ should start during a TW $p \in P_{i_\cdot} = \left\{[l_{i_\cdot}^p, u_{i_\cdot}^p],\ p = 1, \ldots, w_{i_\cdot}\right\}$, where $w_{i_\cdot}$ is the number of MTWs available at $i_\cdot$.

In general, drivers could be assigned several non-conflicting consecutive routes during their shifts. However, for simplification purposes, our model avoids matching a given driver index to many vehicles or routes.

Hence, a driver $k \in \mathcal{K}$ is mapped to a vehicle $k$, which is assigned to a single route. Driver $k$ starts at the depot $D$ and returns to $D$. S/he works during TW $[l_k, u_k]$, and has maximum loading capacity $L_k$ and maximum route duration $T_k$. S/he waits when arriving at a stop before the lower bound of the customer's confirmed pick-up TW. This is the modus operandi of most transport companies that offer DAR services. Thus, the travel duration of a route of driver $k$ is the sum of the total travel, waiting, and service times.

Table 1 summarizes the adopted DARPMTW notation.

## 3. Literature review

Section 3.1 presents the most relevant game theoretic and multi-objective approaches for DARP. Section 3.2 reviews VRPs with MTW (VRPMTW) literature. Section 3.3 surveys data-driven and machine learning methods employed for DARP. Finally, Section 3.4 lists some neighborhood search-based heuristics for DARP applications.

### *3.1. Game theoretic and multiple objective darp*

Game theory is a cross-discipline of economics, mathematics, and computer science. It is based on economic concepts and on mathematical tools that are particularly useful to engineering and management applications, among others. It addresses both cooperative and non-cooperative games. Non-cooperative game theory analyzes the strategic decisions of agents having partially or totally conflicting interests. It is well-suited to design routing strategies for DARPMTW, where the three agents have conflicting interests but must reach a fair compromise between the company's costs, drivers' utility, and customers' satisfaction.

Drakoulis et al. (2018) gamify the on-demand public transport of the medium-sized Greek city of Trikala. They motivate users to 'behave correctly' by optimizing the service from the perspectives of operators and customers. They model the problem as a DARPTW, and apply an insertion heuristic to handle dynamic demand.

Chevrier et al. (2012) improve quality of service and responsiveness to transport requests via a flexible three-objective evolutionary heuristic that hybridizes iterative local search (ILS) with evolutionary operators. They show the relevance of ILS in the mutation step and confirm that all hybridized algorithms outperform their respective counterparts. Molenbruch et al. (2017)

incorporate real-life characteristics of patient transport into a bi-objective DARP. They approximately solve their problem via a multi-directional local search that accounts for the trade-off between service quality and efficiency. They study the impact of restrictions on the set of drivers that can serve users with particular characteristics. For DARP, Zidi et al. (2012) present a multi-objective SA that minimizes route duration under quality of service constraints. Madsen et al. (2015) obtain shorter driving times, higher vehicle utilization, and lower costs via an insertion-based heuristic that allows a flexible multi-objective weighting of customers' preferences. The heuristic is the core of an online scheduling interface, where each service request is treated in less than one second. In the context of a public transport system, Guerriero et al. (2014) tackle a multi-objective variant of the DARP problem with the goal of minimizing both the total ride and waiting times, as well as the maximal duration. They propose a two-step approach to solve this problem. In the first step, they employ meta-heuristics to determine a set of feasible routes. Then, in the second step, they use a bi-objective set partitioning formulation and solve it using the epsilon-constraint method to identify efficient solutions. Hu and Zheng G.C (2017) consider multi-objective functions that include travel costs, service quality, and economic efficiency. They augment their model with speed level constraints that select a unique speed for each arc of the graph. Léhudé et al. (2014) propose a multi-criteria model from multi-attribute utility theory. They approximately solve their multi-criteria DARP via a large neighborhood search. They prove that their new scheduling procedure, which minimizes users' ride time, is fast and efficient.

### 3.2. MTWs

MTWs are increasingly encountered within the classic VRP. Favaretto et al. (2007) present an ant colony heuristic (ACH) for VRPMTW and test it on a set of instances. Belhaiza et al. (2014) design a hybrid variable neighborhood tabu search (HVNTS) that reduces the optimality gaps of ACH. Beheshti et al. (2015) study the multi-objective VRP with multiple non-overlapping TWs set by the company but prioritized by customer's preference. Belhaiza (2018) tests a hybrid VNS algorithm with a multiple-criterion approach on MTW real-life instances. Hoogeboom and Dullaert (2019) consider a variable neighborhood evaluation procedure, which turns out to be very competitive with HVNTS and HGVNS. M'Hallah (2020) designs mathematical and constraint programming-based hybrid heuristics for the same problem. Finally, Belhaiza et al. (2019) present three data-driven hybrid heuristics that are superior to state-of-the-art algorithms on real-life and benchmark instances.

### 3.3. Data-driven DARP approaches

For DARP, data-driven approaches are mainly used to predict demand or transport systems' capacities and strategies. Fu (2002) presents a simulation system for a variety of technology-based dial-a-ride para-transit systems. The simulation assesses the potential effects of automatic vehicle location and of digital communication on these systems and detects potential areas of improvement. Fu (2003) predicts fleet requirements, system capacity, and quality-of-service measures using an analytical model whose calibration is based on a large number of simulation scenarios. The model captures the relationship between fleet size, travel demand, quality-of-service measures, and operating conditions. Based on a computer simulation, Noda et al. (2003) infer that the usability of DAR fixed-route systems with a fixed number of buses drops sharply with increasing service requests and that the number of buses should be proportional to service requests. For a sufficiently large frequency of service requests, DAR systems offer a reasonable solution from both usability and profitability perspectives. Quadrifoglio et al. (2008) use a simulation model to study the impact of operation practices of demand-responsive DAR service providers of Los Angeles County. They investigate the effect of zoning strategies and of TWs on DAR system's performance, measured by total distance, deadhead miles, and fleet size. They infer that there exists a linear relationship between operational practices and performance measures. A one-minute increase of a TW's span saves up to 2 vehicles and 260 miles. Overriding the zoning strategy satisfies the same demand by using 60 vehicles less and driving 10,000 miles less. Marković et al. (2013) highlight the importance of considering entire day operations rather than focusing on peak periods. They identify key performance factors and develop comprehensive statistical and meta-modeling to determine a DAR's system capacity. Marković et al. (2016) propose a statistical machine learning approach that allows accurate predictions of the capacity and the operating costs of a DAR system under a variety of scenarios. Their approach uses a generalized linear model embedded in a support vector regression. It is part of an online system that helps transportation planners and regulators estimate system capacity and operating costs. The system helps planners, who set the target level of service, examine tradeoffs between cost and level of service, and compare the cost of DAR services with other transportation alternatives such as cabs and conventional transit. Neto and Cardoso (2017) present a decision-support system for the dynamic routing of wheelchair users' dynamic transport requests. They propose an algorithm that inserts new requests in existing routes in less than five seconds. Belhaiza introduces customized data-driven local search operators. He prioritizes their application within a neighborhood search heuristic according to four kinds of DARPTW infeasibility, three of which relate to the non-satisfaction of precedence constraints.

### 3.4. Neighborhood search algorithms for DARP

DARPMTW requires efficient optimization heuristics such as adaptive larger neighborhood search (ALNS) (Shaw, 1998; Ropke and Pisinger, 2006). In the context of DARP, the process of relocating requests within routes to more profitable positions is achieved through the repeated use of adaptive destroy–repair operators. While the large neighborhood search (LNS) approach applies a single destroy–repair operator, the adaptive large neighborhood search (ALNS) employs one or several of a variety of operators during each iteration. It selects an operator according to its success rate, which is continuously updated. Using a variety of operators (in a non-systematic manner), ALNS investigates a much larger portion of the search space than most LNS heuristics. In contrast to most

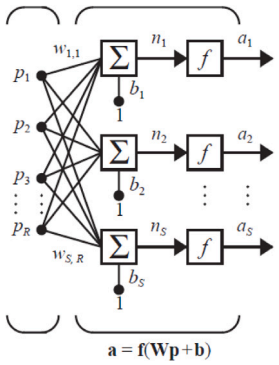


**Fig. 2.** First layer with $S$ neurons and $R$ inputs (Hagan et al., 2018).

LNS heuristics, which induce slight modifications to any solution, ALNS can induce major structural changes to a solution, with 30 to 40% alterations during a single iteration. Thus, ALNS can investigate a much larger neighborhood for any solution (Belhaiza, 2019) at a faster rate. This ALNS advantage is particularly noticeable when the constraints are tight. For example, ALNS is significantly efficient for the rich pickup and delivery problem with TWs (Ropke and Pisinger, 2006); a close problem to DARPMTW.

For a large-scale DARP, Muelas et al. (2015) design a distributed VNS that starts from a solution that minimizes constraints' violation. Their VNS couples several neighborhood classes with a greedy-based approach for solution perturbation (regardless of neighborhood size). Schilde et al. (2014) exploit available data on historical accidents to define a stochastic solution approach for the dynamic DARP. They couple dynamic deterministic and stochastic VNS. Their experiments show the usefulness of stochastic information on travel speeds. Parragh et al. (2010) propose a hybrid VNS whereas Parragh and Schmid (2013) hybridize column generation and LNS. Masson et al. (2013) address DARP with transfer points via an ALNS that generates around 8% of cost savings on real-life instances. For DARPTW, Belhaiza considers a hybridized VNS combining evolutionary crossovers and local search operators, while Belhaiza (2019) hybridizes ALNS with SA and genetic crossovers (GA) and shows its efficiency using benchmarks and real-world instances. Masmoudi et al. (2016) present three heuristics for the multi-depot and multi-trip heterogeneous DARP: improved ALNS, a bees algorithm (BA) hybridized with SA, and a BA with deterministic annealing.

## 4. ANN forecasts for DARPMTW

Parameters of DARP such as travel times and service times are unknown in advance. To estimate the unknown uncertain parameters $y^t$ of DARP, we use observed historical values $p_r^t,\ r = 1, \dots, R,\ t = 1, \dots, T$, of $R$ variables. These values were recorded at discrete time epochs $t$ of the historical horizon whose length is $T$. The estimate $\hat{y}^t$ of $y^t$ is then used as the mean value approximate for the parameters of DARP. A function $F(p^t)$ transforms the input $p^t = (p_1^t, \dots, p_r^t, \dots, p_R^t, 1)'$ into a predicted $\hat{y}^t$, where the prime symbol denotes the transpose.

In ANN terminology, parameter $r$ is an input variable, $R$ is the total number of input variables, $t$ is a time index, $T$ is the size of the training data set, and $y^t$ is the output of a neural network. An activation (transfer) function $f(.)$ is used in a neuron $s,\ s = 1, \dots, S$, of an $S-$neuron ANN, as depicted in Fig. 2. Function $f(.)$ approximates the output $y^t$ at time $t$ based on $p^t$, where $f(p^t) = \hat{y}^t$. The activation function generates a signal $a^t = w \cdot p^t = \sum_{r=1}^{R} w_r p_r^t + b$, where $w = (w_1, \dots, w_r, \dots, w_R, b)$ represents the input weights and $b$ is the bias. In a multi-layer ANN, the neurons of a layer $l,\ l = 1, \dots, L$ have their own activation function $f_l(.)$, and $F$ is a composite function of a complete forward pass through the ANN; thus, the estimate $\hat{y}^t$ of the output $y^t$ is

$$\hat{y}^t = F(p^t) = f_L(\dots f_l(\dots f_1(p^t))).$$

The accuracy of the estimation depends on a few factors such as the data set used to train and validate the ANN, the adopted mathematical model, and the ANN architecture. First, we explain how the historical data is pre-processed. Second, we present the ANN mathematical model, which uses gradient descent methods for weight optimization. Third, we describe the multi-layer ANN architecture and transfer functions. Finally, we determine the best ANN design that we use to predict the mean value estimates of the uncertain DARP parameters.

### 4.1. Training data set

The proposed ANN is trained on the Vancouver data set. A Canadian-based company, time-stamped the GPS locations of 40 DAR vehicles as they traveled through the city and suburbs of Vancouver for a one-year period. It further recorded vehicles' arrival at customers' pick-up and drop-off locations, waiting (if any), loading and unloading, and travel times. It mapped each GPS location to a node on the road map of Vancouver and its suburbs and reported the most frequently-used arcs of the road map.

There are over 1500 arcs in the network. For each arc of the network, there are at least 300 and up to 500 observations.


**Fig. 3.** Vancouver, Burrard st. to Highbury st.

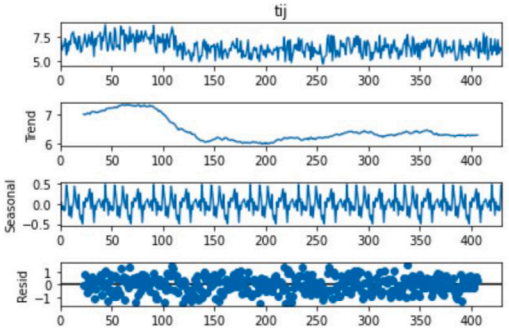


**Fig. 4.** Additive seasonality decomposition results.

For instance, the arc at 4th Avenue-West connecting node (0) Burrard St. to node (1) Highbury St. (cf. Fig. 3) was crossed 430 times. Although the distance in both directions is approximately 3.19 kilometers, the recorded travel times differ depending on the day, time period, road and traffic conditions, and other random events such as planned public activities at the neighboring Kitsilano Park and Beach. For instance, the average travel time from (0) to (1) is approximately 6.46 min, while the average travel time from (1) to (0) is 6.83 min. The fastest travel time from (0) to (1) is 4.86 min, and the slowest from (1) to (0) is 8.71 min.

To each arc $(i, j)$, connecting two DAR stops $i$ and $j$, is associated with a set of historical temporal data. This data consists of observations of independent (input) and dependent (output) variables. At the time $t$, input includes the longitude and latitude of $i$ and $j$, the day order, season, day of the week, and time of day. Output includes the arrival time, waiting time, loading/unloading time, and the observed travel time between $i$ and $j$ at $t$. The day order ranges from 1 to 365 and follows the sequence of days within a year. We consider three seasons, each lasting four months. The day of the week ranges from 1 on Monday to 7 on Sunday. A day is divided into four slots: 6 a.m. to 9 a.m., 9 a.m. to 12 p.m., 12 p.m. to 3 p.m., and 3 p.m. to 6 p.m.

To use the data for training the ANN, we first 'clean it,' by replacing 0-valued outputs with their corresponding average observed values over similar days, seasons, and periods. Second, we apply a principal component analysis (PCA) to reduce the input parameters. PCA reveals that geo-code coordinates do not have a direct impact on travel and service times at a given stop, i.e., the longitude and latitude have much smaller singular values than other input characteristics. On the other hand, day order has the largest singular value 12.94, followed by season and day of the week with 9.99 and 9.17 singular values, respectively. The period of the day has a singular value of 2.18. Third, we account for the impact of weather conditions on travel demand and times by conducting an additive and multiplicative seasonality decomposition using available statistics Python packages. For example, the travel times along arc $(0, 1)$ of Fig. 3 have a periodic trend and a seasonality component. This is reflected by Fig. 4, which shows the seasonality decomposition, and by the 0.1692 $p$-value of the Dickey–Fuller test.

### 4.2. ANN and mathematical formulation

The pre-processing of the data suggests that each arc $(i, j)$ should be characterized by four input and four output variables. The input variables are the day order $p_{ij}^{5t} \in \{1, 2, \ldots, 365\}$, the season $p_{ij}^{6t} \in \{1, 2, 3, 3\}$, the day of the week $p_{ij}^{7t} \in \{1, 2, 3, 6\}$, and the period of the day $p_{ij}^{8t} \in \{1, 2, 3, 4\}$. The output variables at time $t$ are: the arrival time $y_i^{1t}$ at node $i$, the waiting time $y_i^{2t}$, the (un)loading service times $y_i^{3t}$, and the travel time $y_{ij}^{4t}$ from $i$ to $j$.

Herein, we consider ANN designs that use different numbers of layers, activation functions, and weight optimization techniques. We use a two- and a three-layer ANN. We test four activation functions of the first (input) layer:

- the logistic-sigmoid (log-sigmoid) function $f_1(a) = \frac{1}{1+e^{-a}}$,
- the hyperbolic tangent function (tanh) $f_1(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$,
- the sigmoid linear unit function (selu) $f_1(a) = \frac{a}{1+e^{-a}}$, and
- the rectified linear unit function (relu) $f_1(a) = \max\{0, a\}$,

where $a_s = W_s \cdot (p_{ij}^5, \ldots, p_{ij}^8) + b_s$, $W_s$ is the weight vector at node $s$, $s = 1, \ldots, S$, and $b_s$ is the bias value. Finally, we use four gradient-based weight optimization techniques. In fact, after each forward pass $h$, the ANN updates its weights based on its total estimation error $E$ of the output. When it uses a training data set with $T$ observations,

$$E(T) = \sum_{t=1}^{T} (y^t - \hat{y}^t)^2 = \sum_{t=1}^{T} (y^t - F(p^t))^2.$$

When using a classic steepest descent, the ANN updates the weights, based on the average gradient of the entire data set:

$$w_r^{h+1} = w_r^h - \frac{\alpha}{\|\nabla E(T)\|} \frac{\partial E(T)}{\partial w_r},$$

$$b^{h+1} = b^h - \frac{\alpha}{\|\nabla E(T)\|} \frac{\partial E(T)}{\partial b},$$

where $r = 1, \ldots, R$, and $\alpha$ is a prefixed learning rate parameter. In vector form,

$$w^{h+1} = w^h - \frac{\alpha}{\|\nabla E(T)\|} \cdot \nabla E(T).$$

Herein, in lieu of using a classic steepest descent, we test four gradient optimizers: stochastic gradient descent (SGD), adaptive moment estimation (ADAM), adaptive gradient algorithm (AdaGrad), and an extended ADAM (AdaMax).

SGD is a gradient descent variant that is specifically designed to handle large data sets. It estimates the true gradient of the loss function using a randomly selected "mini-batch"of the data; i.e., a sample of size $n_s$. It then updates the model parameters in the direction of the negative gradient to minimize the loss function. Generally, SGD is computationally efficient, reaching a faster convergence and handling much larger data sets than traditional gradient descent methods. However, it may require careful tuning of its hyper-parameters and may fail to converge to the global minimum. Its pseudo-code is detailed in Algorithm 1, where $\nabla Q_j^h$ is an approximation of the gradient $\nabla E_h$ obtained using sample $j$ at iteration $h$, and $\epsilon$ is the precision parameter set herein at $10^{-8}$.

***Algorithm 1: SGD***

Start: Set $h \leftarrow 0$

**Do While** $\|w^{h+1} - w^h\| > \epsilon$:

- $h \leftarrow h + 1$
- Randomly shuffle samples in training set
- Select $j \in \{1, 2, \ldots, n_s\}$
- $w^{h+1} \leftarrow w^h - \alpha \nabla Q_j^h$

**End Do While**

End: return $w^h$

Adam improves the efficiency and convergence of SGD by using the first and second moments of the gradients of the loss function to update the model parameters while requiring relatively little memory. In addition, it can handle sparse gradients; thus, is suitable for deep learning models with a large number of parameters. However, it can be sensitive to the choice of its hyper-parameters and may sometimes converge to sub-optimal solutions if the learning rate is not set correctly. Its pseudo-code, given in Algorithm 2, uses a step size $\alpha > 0$, exponential decay rates $(\beta_1, \beta_2) \in [0, 1)^2$ of the gradient and its moment estimates, and iteration index $h$. Initially, it sets the weight vector $w_0$, the first moment vector $m_0$, and the initial second moment vector $v_0$ to zero vectors and $\alpha = 0.001$, $\beta_1 = 0.900$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

***Algorithm 2: Adam***

Start: Set $h \leftarrow 0$, $w_0 = m_0 = v_0 = \mathbf{0}$.

**Do While** $\|w^{h+1} - w^h\| > \epsilon$:

- $h \leftarrow h + 1$
- $\nabla E_h \leftarrow \nabla E(w_{h-1})$
- $m_h \leftarrow \beta_1 \cdot m_{h-1} + (1 - \beta_1) \cdot \nabla E_h$
- $v_h \leftarrow \beta_2 \cdot v_{h-1} + (1 - \beta_2) \cdot \nabla E_h{}^2$
- $\hat{m}_h \leftarrow \frac{m_h}{(1-\beta_1^h)}$
- $\hat{v}_h \leftarrow \frac{v_h}{(1-\beta_2^h)}$
- $w^h \leftarrow w^{h-1} - \alpha \cdot \hat{m}_h / (\sqrt{\hat{v}_h} + \epsilon)$

**End Do While**

End: return $w^h$

Adagrad adapts the learning rate for each parameter by dividing the learning rate by the sum of the squared gradients for that parameter. Thus, parameters whose gradients are large have small learning rates while those whose gradients are small have large learning rates. However, summing squared gradients over time may cause the learning rate to become too small, prohibiting the algorithm from escaping local minima of the loss function. Regardless, Adagrad handles sparse data very well and eliminates the need to manually set the learning rate, which is automatically adjusted for each parameter. Its basic version updates the weights as:

$$w^{h+1} = w^h - \alpha \cdot \frac{\nabla E_h}{\sqrt{\sum_{i=1}^{h} \nabla E_i^2}}.$$

Adamax generalizes Adam to the infinite norm (max). In addition, it uses the maximum value so far observed instead of computing the moving average of the squared gradients as Adam does. Thus, Adamax is more robust for large gradients, handles sparse data better, and converges faster than Adam. It is relatively easy to implement and requires little memory. However, it may be unsuitable for some problems and requires fine-tuning of its hyper-parameters. Algorithm 3 gives the pseudo-code of Adamax, where $(\frac{\alpha}{1-\beta_1^h})$ is the learning rate with a bias-correction term for the first moment. The first moment vector $m_0$ and the initial exponentially weighted infinity norm $u_0$ are initialized to zero, $\alpha = 0.002$, $\beta_1 = 0.900$ and $\beta_2 = 0.999$.

***Algorithm 3: Adamax***

Start: Set $h \leftarrow 0$

**Do While** $\|w^{h+1} - w^h\| > \epsilon$:

- $h \leftarrow h + 1$
- $\nabla E_h \leftarrow \nabla E_h(w_{h-1})$
- $m_h \leftarrow \beta_1 \cdot m_{h-1} + (1 - \beta_1).\nabla E_h$
- $u_h \leftarrow max(\beta_2 \cdot u_{h-1}, |\nabla E_h|)$
- $w^h \leftarrow w^{h-1} - (\frac{\alpha}{1-\beta_1^h}) \cdot \frac{m_h}{u_h}$

**End Do While**

End: return $w^h$

### 4.3. ANN forecasting results

Table 2 reports the residual mean squared errors (RMSE) for the travel time data set using a two-layered ANN with $S_1 = 5, 10, 25, 50$ neurons in the first layer and activation function(s) $f_1$ = **log-sigmoid, tanh, selu, relu**, on the road segment $(0, 1)$ output features. Table 3 has the same information for a three-layer ANN with $S_1 = 5, 10, 25, 50$ neurons of the first layer, $S_2 = 5, 10, 20$ neurons of the second (hidden) layer, and activation functions $f_1$ and $f_2$ on the road segment $(0, 1)$ output features. The output layer uses activation function **relu** and has from 1 to 10 neurons in Tables 2 and 3. The training set is fixed at $\frac{2}{3}$ of the data set. For example, for arc $(0, 1)$: $\frac{2}{3} \times 430 \approx 288$. The results are the average RMSE over 50 consecutive runs of 50 epochs, where an epoch is a single forward and backward run over the entire training set.

Table 2 suggests that coupling the activation function **relu** with SGD gives, on average, better accuracy than **tanh** or **selu** for a two-layer ANN. This inference holds for the predicted output features and road network. The RMSE is lowest (0.6609) with SGD, $S_1 = 50$ and $S_2 = 10$. SGD performs better and is more stable than the other optimizers across all activation functions.

We conjecture that its stochastic component mitigated the large initial error gradients whereas the other optimizers were unable to overcome this erroneous random initialization.

Table 3 suggests that coupling the $f_1() :$**relu** and $f_2() :$**relu** activation functions with SGD generally provides the best accuracy for the three-layer ANN. The RMSE is lowest (0.4227) when $S_1 = 50$, $S_2 = 20$, and $S_3 = 10$ neurons. SGD is more stable than ADAM when $S_1$ varies from 5 to 50 and $S_2$ varies from 5 to 20 neurons. The activation functions $f_1() :$**tanh** and $f_2() :$**log-sigmoid** provide the largest RMSE on average. The best RMSE (0.6595) is obtained by the ADAM optimizer with $S_1 = 50$ and $S_2 = 20$ neurons. According to the results of the two-sample T-test, no particular architecture is significantly dominant. This result is also confirmed for the different predicted output features. Fig. 5 shows the actual versus predicted travel times with SGD, $f_1() = f_2() = f_3() :$**relu**, $S_1 = 256$, $S_2 = 128$ and $S_3 = 64$ nodes. This combination gives the lowest RMSE = 0.31806. In summary, architectures using an SGD optimizer with two or three layers of neurons dotted with relu activation functions provide the most accurate predictions.

## 5. Game theoretic multi-objective approach for DARPMTW

With the travel, and service times at hand, the problem becomes a static DARPMTW. It is herein modeled as a multi-objective game theoretic problem with three decision agents: the $n$ customers, the $m$ drivers, and the transport company. Each agent seeks to maximize its utility, under a set of constraints. The DARPMTW game allows the drivers and customers to choose each other if they so wish, and allows the transport company to sequence the stops of each driver while ensuring the feasibility of all routes. Once the customer-driver matching is solved, the transport company seeks the best way to accommodate the drivers' and customers' initial choices. This is a two-stage extensive game: Drivers and customers decide simultaneously during the first stage while the transport company decides in the second stage.

For this three-agent game, Section 5.1 proposes a mathematical model whose parameters and decision variables are given in Table 4. The binary condition on decision variables $x$, $y$, and $z$ is relaxed for modeling purposes. Section 5.2 shows that, under few assumptions on the agents' utility coefficients, any Nash equilibrium of the proposed game satisfies the binary nature of $x$, $y$, and $z$.

**Table 2**
RMSE of travel time forecasts using a two-layer ANN.

| | | $f_1()$: log-sigmoid | | | | |
|---|---|---|---|---|---|---|
| $S_1$ | $S_2$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 1 | 0.7619 | 1.8845 | 5.9605 | 2.1290 | 2.6840 |
| 10 | 2 | 0.7779 | 2.1016 | 6.0239 | 2.2453 | 2.7872 |
| 25 | 5 | 0.7781 | 1.9932 | 5.7287 | 2.2049 | 2.6762 |
| 50 | 10 | 0.7982 | 2.0297 | 5.3079 | 2.2845 | 2.6051 |
| | Avg. | 0.7790 | 2.0022 | 5.7552 | 2.2159 | 2.6881 |
| | | $f_1()$: tanh | | | | |
| $S_1$ | $S_2$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 1 | 0.7265 | 1.7430 | 5.8112 | 2.1774 | 2.6145 |
| 10 | 2 | 0.7315 | 1.5831 | 5.3674 | 1.8465 | 2.3821 |
| 25 | 5 | 0.7299 | 1.6973 | 4.9278 | 1.7996 | 2.2887 |
| 50 | 10 | 0.7517 | 1.7001 | 5.0110 | 2.0857 | 2.3871 |
| | Avg. | 0.7349 | 1.6809 | 5.2794 | 1.9773 | 2.4181 |
| | | $f_1()$: selu | | | | |
| $S_1$ | $S_2$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 1 | 0.7628 | 1.4445 | 5.6493 | 1.6198 | 2.3691 |
| 10 | 2 | 0.7487 | 1.9182 | 5.5104 | 1.9673 | 2.5362 |
| 25 | 5 | 0.7507 | 1.7651 | 4.8466 | 1.8726 | 2.3088 |
| 50 | 10 | 0.7603 | 1.7294 | 3.9477 | 1.6215 | 2.0147 |
| | Avg. | 0.7556 | 1.7143 | 4.9885 | 1.7703 | 2.3072 |
| | | $f_1()$: relu | | | | |
| $S_1$ | $S_2$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 1 | 0.7204 | 2.0262 | 6.0948 | 1.8979 | 2.6848 |
| 10 | 2 | 0.6803 | 1.5727 | 5.9224 | 1.8549 | 2.5076 |
| 25 | 5 | 0.6771 | 1.4832 | 5.6420 | 1.9471 | 2.4374 |
| 50 | 10 | **0.6609** | 1.7316 | 5.3504 | 1.6817 | 2.3562 |
| | Avg. | 0.6847 | 1.7034 | 5.7524 | 1.8454 | 2.4965 |

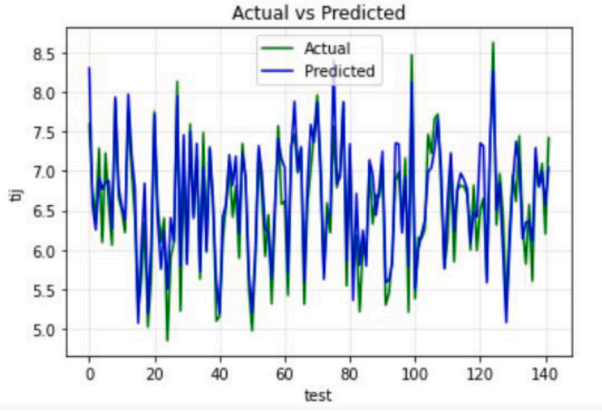


**Fig. 5.** Actual versus predicted travel times with SGD and $f_1() = f_2() = f_3()$ : relu.

### 5.1. Mathematical programs

Any DARP feasible solution sets $x_i^k = y_i^k$, even though the two decisions are determined independently by customer $i \in \mathcal{I}$ and driver $k \in \mathcal{K}$. Herein, these two decision variables are used for game modeling purposes to translate the interaction between customers and drivers, and to formulate two sets of independent mathematical programs. In addition, the interaction between customer $i$ and driver $k$ impacts their respective utility functions $\mathfrak{f}_i(x_i^k, y_i^k)$ and $\mathfrak{g}_k(x_i^k, y_i^k)$. A utility function translates the agent's perception of her gain.

Customers and drivers take part in the game with the intent of maximizing, individually, their respective utility functions. At any Nash equilibrium, all game participants maximize their respective utilities, simultaneously. Thus, no participant has any

**Table 3**
RMSE of travel time forecasts using a three-layer ANN.

| | | | $f_1()$: log-sigmoid; $f_2()$: selu | | | | |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_2$ | $S_3$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 2 | 1 | 0.772 | 1.9569 | 5.7671 | 2.2063 | 2.6756 |
| 10 | 5 | 2 | 0.7695 | 1.6658 | 4.4777 | 1.6279 | 2.1352 |
| 25 | 10 | 5 | 0.7493 | 0.9752 | 2.1198 | 1.0509 | 1.2238 |
| 50 | 25 | 10 | 0.7452 | 0.7317 | 0.7871 | 0.7637 | 0.7569 |
| | | Avg. | 0.759 | 1.3324 | 3.287925 | 1.4122 | 1.6979 |
| | | | $f_1()$: tanh; $f_2()$: selu | | | | |
| $S_1$ | $S_2$ | $S_3$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 2 | 1 | 0.7401 | 1.8111 | 5.522 | 1.8459 | 2.4798 |
| 10 | 5 | 2 | 0.7079 | 1.3616 | 3.847 | 1.2994 | 1.8040 |
| 25 | 10 | 5 | 0.6735 | 0.7267 | 2.0284 | 0.7936 | 1.0556 |
| 50 | 25 | 10 | 0.6653 | 0.6557 | 1.3772 | 0.6949 | 0.8483 |
| | | Avg. | 0.6967 | 1.1388 | 3.1937 | 1.1585 | 1.5469 |
| | | | $f_1()$: tanh; $f_2()$: log-sigmoid | | | | |
| $S_1$ | $S_2$ | $S_3$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 2 | 1 | 0.7557 | 2.0378 | 6.3373 | 2.1743 | 2.8263 |
| 10 | 5 | 2 | 0.7349 | 1.7355 | 5.8134 | 1.5562 | 2.4600 |
| 25 | 10 | 5 | 0.7159 | 1.0192 | 4.3961 | 0.9616 | 1.7732 |
| 50 | 20 | 10 | 0.7321 | 0.6595 | 2.2364 | 0.7001 | 1.0820 |
| | | Avg. | 0.7347 | 1.363 | 4.6958 | 1.3481 | 2.0354 |
| | | | $f_1()$: relu; $f_2()$: relu | | | | |
| $S_1$ | $S_2$ | $S_3$ | SGD | ADAM | AdaGrad | AdaMax | Avg. |
| 5 | 2 | 1 | 0.7381 | 1.8389 | 6.1883 | 2.2629 | 2.7571 |
| 10 | 5 | 2 | 0.6567 | 1.3032 | 5.6233 | 1.1182 | 2.1754 |
| 25 | 10 | 5 | 0.5261 | 0.7425 | 3.5542 | 0.8187 | 1.4104 |
| 50 | 20 | 10 | **0.4227** | 0.4303 | 1.7282 | 0.6144 | 0.7989 |
| | | Avg. | 0.5859 | 1.0787 | 4.2735 | 1.2036 | 1.7854 |

**Table 4**
Variables of DARPMTW game model.

| Variable | Type | Description |
|---|---|---|
| $x^k_{i^+}$ | Binary | = 1 if customer $i$ chooses to be picked by driver $k$, |
| $x^k_{i^-}$ | Binary | = 1 if customer $i$ chooses to be dropped by driver $k$, |
| $x^k_{i\cdot}$ | Binary | = $x^k_{i^+}$ or $x^k_{i^-}$, |
| $\mathbf{x}_i$ | Binary | = vector $\left(x^k_{i^+}, x^k_{i^-}\right)_{k\in\mathcal{K}}$, |
| $v^p_{i\cdot}$ | Binary | = 1 if node $i\cdot$ is served during time window $p$, |
| $y^k_{i^+}$ | Binary | = 1 if driver $k$ chooses to pick-up customer $i$, |
| $y^k_{i^-}$ | Binary | = 1 if driver $k$ chooses to drop customer $i$, |
| $y^k_{i\cdot}$ | Binary | = $y^k_{i^+}$ or $y^k_{i^-}$, |
| $r^k_{i\cdot}$ | Real | = waiting time of vehicle $k$ at stop $i\cdot$, |
| $\mathbf{y}_j$ | Binary | = vector $\left(y^k_{i^+}, y^k_{i^-}\right)_{k\in\mathcal{K}}$, |
| $a^k_{i\cdot}$ | Real | = arrival time of vehicle $k$ at node $i\cdot$, |
| $b^k_{i\cdot}$ | Real | = departure time of vehicle $k$ from node $i\cdot$, |
| $z^k_{i^+j^+}$ | Binary | = 1 if company decides driver $k$ serves $i^+$ immediately before $j^+$, |
| $z^k_{i^-j^-}$ | Binary | = 1 if company decides driver $k$ serves $i^-$ immediately before $j^-$, |
| $z^k_{i^+j^-}$ | Binary | = 1 if company decides driver $k$ serves $i^+$ immediately before $j^-$, |
| $z^k_{i^-j^+}$ | Binary | = 1 if company decides driver $k$ serves $i^-$ immediately before $j^+$, |
| $z^k_{i\cdot j\cdot}$ | Binary | = $z^k_{i^+j^+}$, $z^k_{i^+j^-}$, $z^k_{i^-j^+}$, or $z^k_{i^-j^-}$, |
| $q^k_{i\cdot j\cdot}$ | Real | = number of passengers riding vehicle $k$ from $i\cdot$ to $j\cdot$, |
| $\mathbf{z}$ | Binary | = vector $\left(z^k_{i\cdot j\cdot}\right)_{i\cdot\in\mathcal{N},j\cdot\in\mathcal{N},k\in\mathcal{K}}$. |

incentive to deviate from their equilibrium decision vector, i.e., customer $i \in \mathcal{I}$ from $\mathbf{x}_i = \left\{ (x^k_{i^+}, x^k_{i^-}),\ k \in \mathcal{K} \right\}$ and driver $k \in \mathcal{K}$ from $\mathbf{y}_k = \left\{ (y^k_{i^+}, y^k_{i^-}),\ i \in \mathcal{I} \right\}$ (provided the other participants do not deviate).

For a DARP game, the Nash equilibrium conditions follow.

**Definition 1.** A Nash equilibrium for a DARP game is a vector of strategies $(\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n, \hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_m, \hat{\mathbf{z}})$. Strategy vector $\hat{\mathbf{x}}_i$ of customer $i$, $i \in \mathcal{I}$, maximizes the linear program $LP_i$:

$$\hat{\mathbf{x}}_i \in \underset{x^k_{i\cdot}}{\operatorname{argmax}} \sum_{k=1}^{m} \mathfrak{f}_{ik}(x^k_{i\cdot}, \hat{y}^k_{i\cdot}) \tag{1}$$

$$\text{subject to} \quad \sum_{k=1}^{m} x^k_{i^+} \le 1, \tag{2}$$

$$\sum_{k=1}^{m} x^k_{i^-} \le 1, \tag{3}$$

$$x^k_{i^+} \le \hat{y}^k_{i^+}, \quad \forall\ k \in \mathcal{K}, \tag{4}$$

$$x^k_{i^-} \le \hat{y}^k_{i^-}, \quad \forall\ k \in \mathcal{K}, \tag{5}$$

$$x^k_{i^+} = x^k_{i^-}, \quad \forall\ k \in \mathcal{K}, \tag{6}$$

$$0 \le x^k_{i\cdot} \le 1, \ \forall\ k \in \mathcal{K}, \tag{7}$$

where $y^k_i = \hat{y}^k_i$ when $\hat{y}^k_i$ is known. Function (1) sets customer $i$'s objective, which is to maximize the sum of her utility across all drivers. Conditions (2) and (3) state that neither pick-up $i^+$ nor drop $i^-$ is served more than once. Conditions (4) and (5) ensure that customer $i$ can be picked up and dropped by driver $k$ if and only if $k$ decides to serve $i$. Condition (6) guarantees that the pick up $i^+$ and drop $i^-$ of customer $i$ are served by the same driver $k$. Finally, condition (7) relaxes the binary constraint on the $x^k_{i^+}$ and $x^k_{i^-}$ decision variables. This relaxation is only needed for primal–dual LP relations.

Strategy vector $\hat{\mathbf{y}}^k$ of driver $k$, $k \in \mathcal{K}$, maximizes $LP^k$:

$$\hat{\mathbf{y}}^k \in \underset{y^k_{\cdot}}{\operatorname{argmax}} \sum_{i=1}^{n} \mathfrak{g}_{ik}(\hat{x}^k_{i\cdot}, y^k_{i\cdot}) \tag{8}$$

$$\text{subject to} \quad y^k_{i^+} \le \hat{x}^k_{i^+}, \quad \forall\ i^+ \in \mathcal{N}, \tag{9}$$

$$y^k_{i^-} \le \hat{x}^k_{i^-}, \quad \forall\ i^- \in \mathcal{N}, \tag{10}$$

$$0 \le y^k_{i\cdot} \le 1, \ \forall\ i^- \in \mathcal{N}, \tag{11}$$

where $x^k_i = \hat{x}^k_i$ when $\hat{x}^k_i$ is known. Function (8) sets driver $k$'s objective, which maximizes the sum of $k$'s utility across all customers. Conditions (9) and (10) ensure that driver $k$ may pick up or drop customer $i$ if and only if $i$ decides to be served by $k$. Finally, condition (11) relaxes the binary condition on decision variables $y^k_{i^+}$ and $y^k_{i^-}$. This relaxation is only needed for primal–dual LP purposes, as detailed in Section 5.2.

The company' strategy is described by vector $\hat{\mathbf{z}} = (z^k_{i\cdot j\cdot})$, $i\cdot \in \mathcal{V}$, $j\cdot \in \mathcal{V}, k \in \mathcal{K}$, and $z^k_{i\cdot i\cdot} = 0$. $\hat{\mathbf{z}}$ is the solution of the following linear program $LP$

$$\hat{\mathbf{z}} \in \underset{z}{\operatorname{argmax}} \sum_{k\in\mathcal{K}} \sum_{(i\cdot,j\cdot)\in\mathcal{V}^2} t_{i\cdot j\cdot} z^k_{i\cdot j\cdot} + \left( \sum_{k\in\mathcal{K}} \sum_{i\cdot\in\mathcal{N}} r^k_{i\cdot} + \sum_{i\cdot\in\mathcal{N}} s_{i\cdot} \right) \tag{12}$$

$$\text{subject to} \sum_{j\cdot\in\mathcal{V}} z^k_{j\cdot i\cdot} = \sum_{j\cdot\in\mathcal{V}} z^k_{i\cdot j\cdot}, \forall\ i\cdot \in V, \ k \in \mathcal{K}, \tag{13}$$

$$2z^k_{i\cdot j\cdot} \le \hat{x}^k_{i\cdot} + \hat{y}^k_{j\cdot}, \forall\ i\cdot \in \mathcal{N}, \ j\cdot \in \mathcal{N}, \ k \in \mathcal{K}, \tag{14}$$

$$\sum_{k\in\mathcal{K}} \sum_{j\cdot\in\mathcal{V}} z^k_{i\cdot j\cdot} \le 1, \ \forall\ i\cdot \in \mathcal{V}, \tag{15}$$

$$\sum_{k\in\mathcal{K}} \sum_{j\cdot\in\mathcal{V}} z^k_{j\cdot i\cdot} \le 1, \ \forall\ i\cdot \in \mathcal{V}, \tag{16}$$

$$q^k_{i\cdot j\cdot} \le L_k z^k_{i\cdot j\cdot}, \ \forall\ i\cdot \in \mathcal{V}, \ j\cdot \in \mathcal{V}, \ k \in \mathcal{K}, \tag{17}$$

$$\sum_{j\cdot\in\mathcal{V}} q^k_{j\cdot i\cdot} - \sum_{j\cdot\in\mathcal{V}} q^k_{i\cdot j\cdot} \ge d_{i\cdot} \hat{x}^k_{i\cdot}, \ \forall i\cdot \in \mathcal{N}, \ k \in \mathcal{K}, \tag{18}$$

$$b^k_{i\cdot} \ge a^k_{i\cdot} + r^k_{i\cdot} + s_{i\cdot} - M(1 - \hat{y}^k_{i\cdot}), \ \forall\ i\cdot \in \mathcal{N}, \ k \in \mathcal{K}, \tag{19}$$

$$a^k_{j\cdot} \ge b^k_{i\cdot} + t_{i\cdot j\cdot} - M(1 - z^k_{i\cdot j\cdot}), \ \forall\ i\cdot \in \mathcal{V}, \ j\cdot \in \mathcal{V}, \ i\cdot \ne j\cdot, \ k \in \mathcal{K}, \tag{20}$$

$$a^k_{i\cdot} + r^k_{i\cdot} \geq l^p_{i\cdot} - M(1-\hat{x}^k_{i\cdot}) - M(1-v^p_{i\cdot}),$$
$$\forall\, i\cdot \in \mathcal{N},\ p \in P_{i\cdot},\ k \in \mathcal{K}, \tag{21}$$
$$a^k_{i\cdot} + r^k_{i\cdot} \leq u^p_{i\cdot} + M(1-\hat{x}^k_{i\cdot}) + M(1-v^p_{i\cdot}),$$
$$\forall\, i\cdot \in \mathcal{N},\ p \in P_{i\cdot},\ k \in \mathcal{K}, \tag{22}$$
$$\sum_{p=1}^{p_{i\cdot}} v^p_{i\cdot} = 1,\ \forall\, i\cdot \in \mathcal{V}, \tag{23}$$
$$r^k_{i\cdot},\ a^k_{i\cdot},\ b^k_{i\cdot} \geq 0,\ \forall\, i\cdot \in \mathcal{V},\ k \in \mathcal{K}, \tag{24}$$
$$q^k_{i\cdot j\cdot} \geq 0,\ \forall\, i\cdot \in \mathcal{V},\ j\cdot \in \mathcal{V},\ i\cdot \neq j\cdot,\ k \in \mathcal{K}, \tag{25}$$
$$0 \leq z^k_{i\cdot j\cdot} \leq 1,\ \forall\, i\cdot \in \mathcal{V},\ j\cdot \in \mathcal{V},\ i\cdot \neq j\cdot,\ k \in \mathcal{K}, \tag{26}$$
$$0 \leq v^p_{i\cdot} \leq 1,\ \forall\, i\cdot \in \mathcal{V},\ p \in P_{i\cdot}, \tag{27}$$

where $M$ is a large positive number. The function (12) sets the objective of the enterprise. It minimizes the total duration of all routes, where the duration of a route includes travel, waiting, and service times. Condition (13) is a flow conservation constraint that dictates that the number of arcs leaving a node $i\cdot \in \mathcal{V}$ is equal to the number of active arcs entering $i\cdot$. If $i\cdot$ is a depot, this condition causes the route of the vehicle $k$ to start and end in the depot. Condition (14) states that the arc $(i\cdot, j\cdot)$ can be traversed by vehicle $k$ only if $\hat{x}^k_{i\cdot}$ and $\hat{y}^k_{j\cdot}$ are both set to 1. Conditions (15) and (16) require that each node $i\cdot$ has one entering and one exiting arc. Condition (17) ensures that the number of customers traveling from $i$ to $j$ in vehicle $k$ cannot exceed the load capacity of vehicle $L_k$. Condition (18) states that the demand of each node $i\cdot$ is satisfied. Condition (19) calculates the departure time from node $i\cdot$, being greater than or equal to the sum of the arrival, waiting, and service times at $i\cdot$ when $i\cdot$ is served by vehicle $k$. Condition (20) calculates the arrival time at node $j\cdot$; it is set greater than or equal to the sum of the departure time from $i\cdot$ and the travel time $t_{i\cdot j\cdot}$ when vehicle $k$ traverses arc $(i\cdot, j\cdot)$. Conditions (21) and (22) state that serving $i\cdot$ during time window $\left[l^p_{i\cdot}, u^p_{i\cdot}\right]$ starts when vehicle $k$ serves $i\cdot$ during its time window $p$. Service starting time is the sum of the arrival and waiting times at $i\cdot$. Condition (23) imposes that service to $i\cdot$ take place during a single time window. Finally, conditions (24) to (27) define the nature of the decision variables.

For simplification purposes, we assume quadratic utility functions $\mathfrak{f}$ and $\mathfrak{g}$.

**Assumption 1.** The utility of a customer $i \in \mathcal{I}$ with respect to (w.r.t.) driver $k \in \mathcal{K}$ is $\mathfrak{f}_{ik}(x^k_{i\cdot}, \hat{y}^k_{i\cdot}) = (x^k_{i^+},\ x^k_{i^-}) F^k_i (\hat{y}^k_{i^+},\ \hat{y}^k_{i^-})^T$, where

$$F^k_i = \begin{pmatrix} f^k_{i^+} & 0 \\ 0 & f^k_{i^-} \end{pmatrix}$$

is a square diagonal matrix representing the expected utility of customer $i$ when served by driver $k$, and $(f^k_{i^+}, f^k_{i^-}) \in \mathbb{R}^2$.

The intuition behind Assumption 1 is that the utility of a pick-up $i^+$ or a drop $i^-$ w.r.t. a driver $k$ is null. Thus, the utility of customer $i$ w.r.t. driver $k$ is a linear function:

$$\begin{aligned} \mathfrak{f}_{ik}(x^k_{i\cdot}, \hat{y}^k_{i\cdot}) &= (x^k_{i^+},\ x^k_{i^-}) F^k_i (\hat{y}^k_{i^+},\ \hat{y}^k_{i^-})^T \\ &= x^k_{i^+} f^k_{i^+} \hat{y}^k_{i^+} + x^k_{i^-} f^k_{i^-} \hat{y}^k_{i^-}. \end{aligned}$$

Consequently, function (1) is linear, and $LP_i$ becomes:

$$\hat{\mathbf{x}}_i \in \underset{x^k_{i^+}, x^k_{i^-}}{\operatorname{argmax}} \sum_{k=1}^{m} (x^k_{i^+},\ x^k_{i^-}) F^k_i (\hat{y}^k_{i^+},\ \hat{y}^k_{i^-})^T \quad \text{subject to} \quad (2)\text{–}(7).$$

**Assumption 2.** The utility of driver $k \in \mathcal{K}$ w.r.t. a customer $i \in \mathcal{I}$ is $\mathfrak{g}_{ik}(\hat{x}^k_{i\cdot}, y^k_{i\cdot}) = (\hat{x}^k_{i^+},\ \hat{x}^k_{i^-}) G^k_i (y^k_{i^+},\ y^k_{i^-})^T$, where

$$G^k_i = \begin{pmatrix} g^k_{i^+} & 0 \\ 0 & g^k_{i^-} \end{pmatrix}$$

is a square diagonal matrix representing the expected utility of driver $k$ when served by customer $i$ and $(g^k_{i^+}, g^k_{i^-}) \in \mathbb{R}^2$.

Similarly, the intuition behind Assumption 2 is that driver $k$'s utility w.r.t. a drop $i^-$ or a pick-up $i^+$ is null. Thus, the utility of driver $k$ w.r.t. customer $i$ is a linear function:

$$\begin{aligned} \mathfrak{g}_{ik}(\hat{x}^k_{i\cdot}, y^k_{i\cdot}) &= (\hat{x}^k_{i^+},\ \hat{x}^k_{i^-}) G^k_i (y^k_{i^+},\ y^k_{i^-})^T \\ &= \hat{x}^k_{i^+} g^k_{i^+} y^k_{i^+} + \hat{x}^k_{i^-} g^k_{i^-} y^k_{i^-}. \end{aligned}$$

Consequently, function (8) is linear, and $LP^k$ becomes

$$\hat{\mathbf{y}}_{\cdot k} \in \underset{y^k_{i^+}, y^k_{i^-}}{\operatorname{argmax}} \sum_{i=1}^{n} (\hat{x}^k_{i^+},\ \hat{x}^k_{i^-}) G^k_i (y^k_{i^+},\ y^k_{i^-})^T \quad \text{subject to} \quad (9)\text{–}(11).$$

### 5.2. Nash equilibrium conditions

A Nash equilibrium for the DARP game satisfies the constraints of $LP_i,\ i \in \mathcal{I},\ LP^k,\ k \in \mathcal{K}$, and of their respective dual linear programs $DLP_i$ and $DLP^k$. For customer $i \in \mathcal{I},\ DLP_i$ is given by

$$\min_{\delta_{i^+},\delta_{i^-},\alpha^k_{i^+},\alpha^k_{i^-},\beta^k_i} \delta_{i^+} + \delta_{i^-} + \sum_{k=1}^{m} \alpha^k_{i^+} \hat{y}^k_{i^+} + \sum_{k=1}^{m} \alpha^k_{i^-} \hat{y}^k_{i^-} \tag{28}$$

$$\text{subject to} \quad \delta_{i^+} + \alpha^k_{i^+} + \beta^k_i \geq f^k_{i^+} \hat{y}^k_{i^+},\ \forall\ k \in \mathcal{K}, \tag{29}$$

$$\delta_{i^-} + \alpha^k_{i^-} - \beta^k_i \geq f^k_{i^-} \hat{y}^k_{i^-},\ \forall\ k \in \mathcal{K}, \tag{30}$$

$$\delta_{i^+}, \delta_{i^-}, \alpha^k_{i^+}, \alpha^k_{i^-} \geq 0,\ \forall\ k \in \mathcal{K}, \tag{31}$$

$$\beta^k_i \in \mathbb{R},\ \forall\ k \in \mathcal{K}. \tag{32}$$

Similarly, for a driver $k \in \mathcal{K},\ DLP^k$ is given by:

$$\min_{\lambda^k_{i^+},\lambda^k_{i^-}} \sum_{i=1}^{n} \hat{x}^k_{i^+} \lambda^k_{i^+} + \sum_{i=1}^{n} \hat{x}^k_{i^-} \lambda^k_{i^-} \tag{33}$$

$$\text{subject to} \quad \lambda^k_{i^+} \geq \hat{x}^k_{i^+} g^k_{i^+},\ \forall\ i^+ \in \mathcal{N}, \tag{34}$$

$$\lambda^k_{i^-} \geq \hat{x}^k_{i^-} g^k_{i^-},\ \forall\ i^- \in \mathcal{N}, \tag{35}$$

$$\lambda^k_{i^+}, \lambda^k_{i^-} \geq 0,\ \forall\ i^+,\ i^- \in \mathcal{N}. \tag{36}$$

**Definition 2.** Given a DARP game, for every Nash equilibrium $(\hat{x}_{\cdot_1}, \ldots, \hat{x}_{\cdot_n}, \hat{y}^1_{\cdot}, \ldots, \hat{y}^m_{\cdot})$, there exist dual variables $\hat{\delta}_{i\cdot} \geq 0$, $\hat{\alpha}^k_{i\cdot} \geq 0$, $\hat{\beta}^k_i$ free, and $\hat{\lambda}^k_{i\cdot} \geq 0$ that minimize $DLP_i$ and $DLP^k$, $\forall i \in \mathcal{N}$ and $\forall k \in \mathcal{K}$.

The primal–dual optimality conditions for customer $i$ and driver $k$ are

$$(\delta_{i^+} + \alpha^k_{i^+} + \beta^k_i - f^k_{i^+} y^k_{i^+}) x^k_{i^+} = 0,\ \forall\ k \in \mathcal{K}, \tag{37}$$

$$(\delta_{i^-} + \alpha^k_{i^-} - \beta^k_i - f^k_{i^-} y^k_{i^-}) x^k_{i^-} = 0,\ \forall\ k \in \mathcal{K}, \tag{38}$$

and

$$(\lambda^k_{i^+} - x^k_{i^+} g^k_{i^+}) y^k_{i^+} = 0,\ \forall\ i^+ \in \mathcal{N}, \tag{39}$$

$$(\lambda^k_{i^-} - x^k_{i^-} g^k_{i^-}) y^k_{i^-} = 0,\ \forall\ i^- \in \mathcal{N}. \tag{40}$$

Eqs. (37) and (38) are linearized via binary variables $u^k_{i\cdot}$; thus, for a customer $i$,

$$\delta_{i^+} + \alpha^k_{i^+} + \beta^k_i - f^k_{i^+} y^k_{i^+} \leq M u^k_{i^+},\ \forall\ k \in \mathcal{K}, \tag{41}$$

$$\delta_{i^-} + \alpha^k_{i^-} - \beta^k_i - f^k_{i^-} y^k_{i^-} \leq M u^k_{i^-},\ \forall\ k \in \mathcal{K}, \tag{42}$$

$$x^k_{i^+} + u^k_{i^+} \leq 1,\ \forall\ k \in \mathcal{K}, \tag{43}$$

$$x^k_{i^-} + u^k_{i^-} \leq 1,\ \forall\ k \in \mathcal{K}, \tag{44}$$

$$u^k_{i^+}, u^k_{i^-} \in 0, 1,\ \forall\ k \in \mathcal{K}. \tag{45}$$

When $u^k_{i^+} = 1$, Eq. (43) sets $x^k_{i^+} = 0$; thus, Eq. (37) is satisfied. Meanwhile, when $u^k_{i^+} = 0$, the right-hand side of Eq. (41) is zero; forcing its left-hand side to be zero, because conditions (29) must be satisfied. Similarly, with $u^k_{i^-} = 1$ or $u^k_{i^-} = 0$, Eq. (38) is satisfied. In all cases, the primal–dual conditions for customer $i$ are satisfied.

Similarly, Eqs. (39) and (40) are linearized via binary variables $o^k_{i\cdot}$; thus, for each driver $k \in \mathcal{K}$,

$$\lambda^k_{i^+} - x^k_{i^+} g^k_{i^+} \leq M o^k_{i^+},\ \forall\ i^+ \in \mathcal{N}, \tag{46}$$

$$\lambda^k_{i^-} - x^k_{i^-} g^k_{i^-} \leq M o^k_{i^-},\ \forall\ i^- \in \mathcal{N}, \tag{47}$$

$$y^k_{i^+} + o^k_{i^+} \leq 1,\ \forall\ i^+ \in \mathcal{N}, \tag{48}$$

$$y^k_{i^-} + o^k_{i^-} \leq 1,\ \forall\ i^- \in \mathcal{N}, \tag{49}$$

$$o^k_{i^+}, o^k_{i^-} \in 0, 1,\ \forall\ i^+,\ i^- \in \mathcal{N}. \tag{50}$$

When $o^k_{i^+} = 1$, Eq. (48) sets $y^k_{i^+} = 0$; thus, Eq. (39) is satisfied. Meanwhile, when $o^k_{i^+} = 0$, the right-hand side of Eq. (46) is zero; forcing its left-hand side to be zero because of conditions (34). Similarly, with $o^k_{i^-} = 1$ or $o^k_{i^-} = 0$, Eq. (40) is satisfied. Hence, all primal–dual conditions for driver $k \in \mathcal{K}$ are satisfied. Proposition 1 compiles all the above Nash equilibrium conditions.

**Proposition 1.** *For a DARP game, any Nash equilibrium satisfies conditions (2)–(7), (9)–(11), (29)–(32), (34)–(36) and (41)–(50).*

Proposition 2, develops the assumptions on the utility matrices $F_i^k$ and $G_i^k$.

**Proposition 2.** *When the payoff matrices $F_i^k$ and $G_i^k$, $\forall i \in \mathcal{I}$ and $\forall k \in \mathcal{K}$, are non-negative, and there exists a binary solution $(X^*, Y^*) = ((x_{i\cdot}^{k^*}), (y_{i\cdot}^{k^*}))_{i\cdot\in\mathcal{N},k\in\mathcal{K}}$ (i.e., that satisfies all customers' requests), then $(X^*, Y^*)$ is a Nash equilibrium for the DARP game.*

Proofs of Propositions 1 and 2 are provided in Appendix A.

Proposition 2 suggests that a feasible solution to DARPMTW is a Nash equilibrium for the DARP game when the entries of the payoff matrices $F_i^k$ and $G_i^k$ are all non-negative. As a result, it is assumed that all elements of the payoff matrices $F_i^k$ and $G_i^k$ are non-negative.

## 6. Multi-objective DD-HALNS for DARPMTW

This section describes how the objectives of DARPMTW game agents are optimized. For the company, minimizing the total travel duration

- reduces the overall work schedules of the drivers, who spend less time waiting and more time productively, and
- ensures a better quality of service to the customers, who spend less time in transit.

For a solution $X = (X_1, \ldots, X_m)$, defined by its set of $m$ routes $X_k$, $k \in \mathcal{K}$, the total travel duration of $X$ is

$$T(X) = \sum_{k\in\mathcal{K}} T(X_k),$$

where $T(X_k)$ is the total duration of vehicle $k$'s route $X_k$.

For the drivers, maximizing the minimum driver utility provides better overall driver satisfaction. It ensures that the less fortunate driver reaches the largest possible level of satisfaction $G(X)$. This is given by:

$$\max_j \; G(X)$$

Subject to

$$G(X) \le G_j,$$
$$G_j = \sum_{k=1}^{m} (\hat{x}_{i^+}^k, \; \hat{x}_{i^-}^k) G_i^k (y_{i^+}^k, \; y_{i^-}^k)^T,$$
$$G_j \ge 0.$$

For the customers, maximizing the minimum customer utility enhances overall customer satisfaction. It guarantees a minimum level of service $F(X)$ to all customers. This is given by:

$$\max_i \; F(X)$$

Subject to

$$F(X) \le F_i,$$
$$F_i = \sum_{k=1}^{m} (\hat{x}_{i^+}^k, \; \hat{x}_{i^-}^k) F_i^k (y_{i^+}^k, \; y_{i^-}^k)^T,$$
$$F_i \ge 0.$$

By balancing these three objectives, multi-objective DD-HALNS reaches quasi-optimal routing solutions. Clusters and insertions are used by DD-HALNS to construct a solution to DARPMTW. A sequence of local and global search operators is then intensively applied around the incumbent in its neighboring regions. In addition, it diversifies this exploration to avoid local minima. By using data-driven operators, it can improve solutions (local search) and determine areas that contain the best solutions (global search). DD-HALNS relies on new operators and simulated annealing (SA) for intensification. For diversification strategies, DD-HALNS relies on evolutionary pools of best parent solutions in cases where intensification may fail to ameliorate the incumbent. As DD-HALNS searches, it generates a pool of best solutions $\mathcal{L}$, and stores them in dynamic tables. At later stages of the search, this tabu list may also be re-explored.

The entries $F_i^k$ and $G_i^k$ of the utility matrices $F$ and $G$ are set to values that satisfy the conditions of Proposition 2. The customers' and the drivers' parameters are used for this purpose. We define $F_i^k = G_i^k = \frac{|d_{i\cdot}|}{1+s_i+t_i^k}$, where $t_i^k$ is the travel time from the depot of driver $k$ to customer $i$. For the customers, the chosen utility creates a positive correlation between their utilities and demand, while also creating a negative correlation between their utilities and their service and travel times from the driver's depot. Conversely, for the driver, this choice penalizes customers who have longer service or travel times from their depot. This viewpoint differs from that of the customer's perspective.

The logic structure diagram of the multi-objective DD-HALNS is illustrated in Fig. 6.

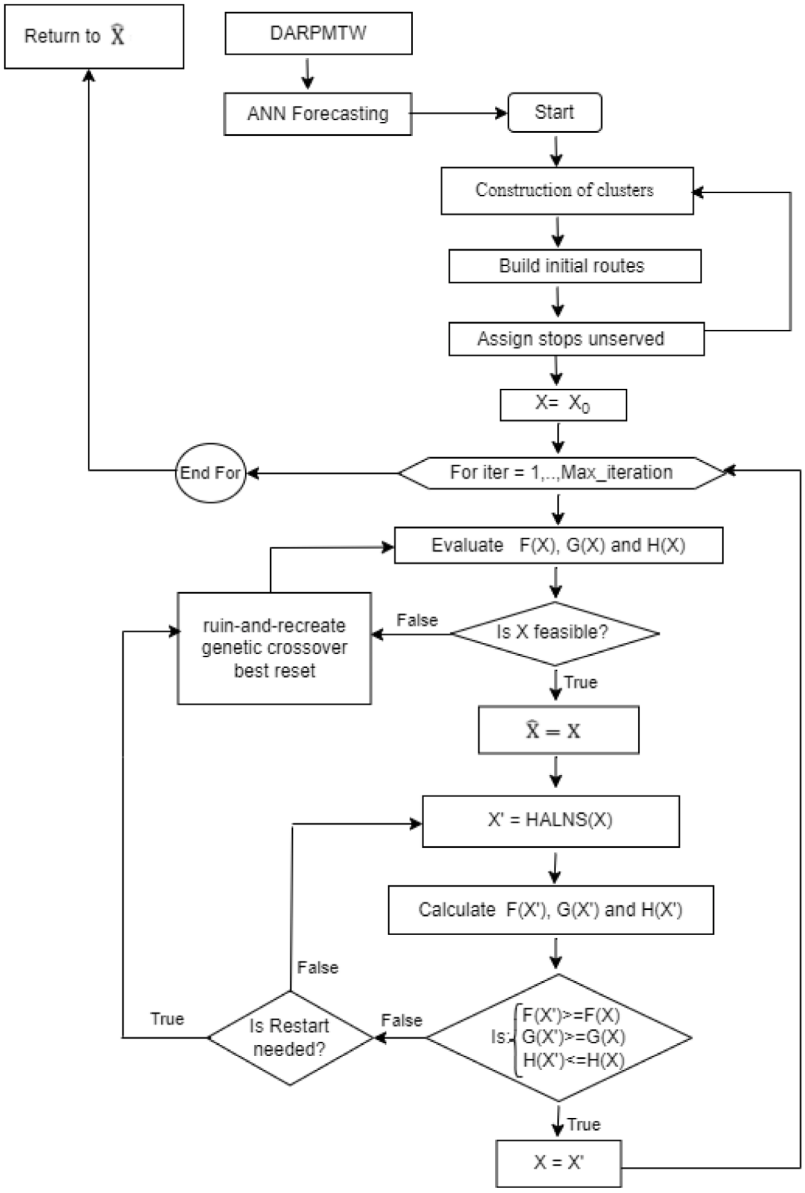


**Fig. 6.** DD-HALNS logic structure diagram.

### 6.1. Initialization

Diana and Dessouky (2004) propose a regret insertion heuristic for DARPTW. They introduce a novel method for initializing routes that considers both the spatial and temporal aspects of the problem and also includes regret insertion for fulfilling remaining requests. Similarly, DD-HALNS utilizes a three-step approach to determine its incumbents: First, it clusters customers into circles with the same radius; second, it builds a route using some or all of the stops in each cluster; and third, it inserts stops left unserved into existing routes or creates new routes, as some stops may be left unassigned to a route after the second step. The subsequent section provides further information on these three steps.

#### 6.1.1. Construction of clusters

The cluster construction process begins by generating a random permutation $\Pi = (\pi_1, \ldots, \pi_{2n})$ of the pickups and the drops for the $n$ customers. The stop served in position $\zeta$ is represented by $\pi_\zeta$. The stops of $\Pi$ are then assigned sequentially, in the order they appear, to clusters of radius $R$ (as shown in Fig. 7). The construction process of the first cluster sets the focal point of the cluster to $\pi_1$ and initializes the number of clusters $c$ to 1. Additionally, the radius $R$ of the cluster is also set.

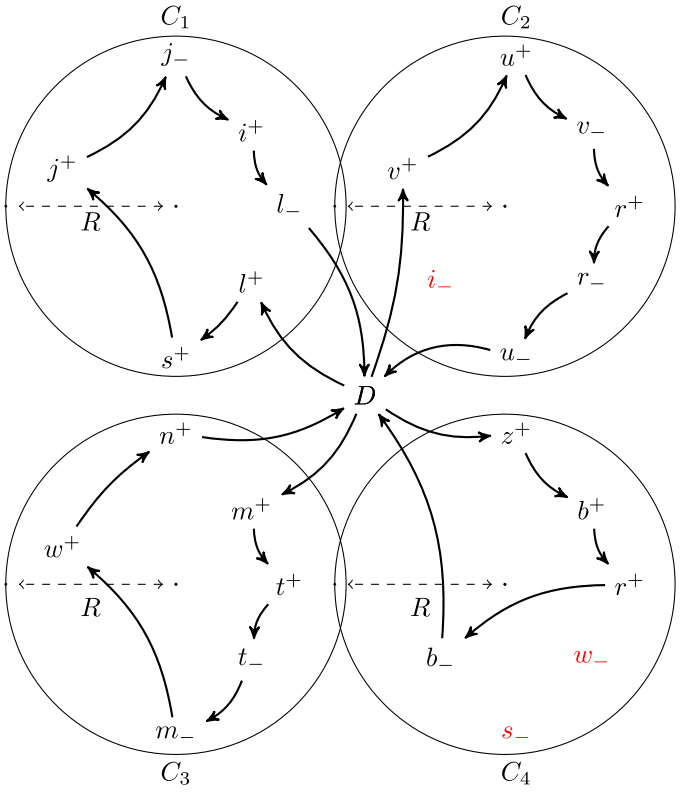


**Fig. 7.** Cluster assignment.

During the construction process, each stop in the set $\Pi$ is checked to see if it can be assigned to an existing cluster. This is done by sequentially checking the distance between the stop $\pi_\zeta$ and the focal point of each existing cluster $c'$, where $c' = 1, \ldots, c$, to see if it is less than or equal to the radius $R$. The process of constructing clusters goes through each stop in $\Pi$ until all stops have been assigned to a cluster or a new cluster has been created to serve any remaining stops. This is achieved by checking if a stop $\pi_\zeta$ falls within a circle of radius $R$ centered at the focal point of a cluster $c'$. If such a cluster exists, $\pi_\zeta$ is assigned to it, and the focal point of the cluster is updated to the median position of all stops in the cluster. The median point is calculated based on the coordinates of the stops already assigned to the cluster. If no cluster can accommodate $\pi_\zeta$, a new cluster is created. In this case, the algorithm increases the value of $c$ by 1, adds $\pi_\zeta$ to the new cluster, and sets the focal point of the new cluster as $\pi_\zeta$. The set of pickups and drop-offs for cluster $c'$ is represented as $C_{c'}$, where $c' = 1, \ldots, c$.

The number of clusters, denoted by $c$, is determined by the radius $R$ used in the clustering process. If $R$ is large, $c$ will be small, and if $R$ is small, $c$ will be large. The aim is to choose an appropriate $R$ such that $c$ is close to the number of available vehicles $m$. To begin with, the median position of all the stops is computed and set as the focal point $i_0 = (x_0, y_0)$. The maximum distance from any stop $i_\cdot$ in the set $\mathcal{N}$ to $i_0$ is then computed and denoted by $\overline{R}$, which is given by:

$$\overline{R} = \max_{i_\cdot \in \mathcal{N}} = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}.$$

If $R$ is set to $\overline{R}$, all the stops will be grouped into a single cluster. Alternatively, if $R$ is set to $\frac{\overline{R}}{2}$, it will result in a larger number of smaller clusters. Finally, if all $m$ vehicles must be used, $R$ is set to $R = \frac{\overline{R}}{m}$.

### 6.1.2. Construction of routes

The route construction process aims to create an initial solution $X_0$ by utilizing the $c$ clusters available to allocate a certain number of stops from cluster $c' = k$ to each vehicle $k \in \mathcal{K}$. At the beginning of the process, all vehicles are located at the depot. The construction process initializes $k$ to 1 and iterates through each vehicle. For each vehicle, it selects the customer pickup $i^+$, from the set of available customers $C_k$, that results in the minimum increase in the cost of the vehicle's route. While making sure that they do not exceed the capacity $q_k$ and duration $t_k$, it updates the route of vehicle $k$ by adding both $i^+$ and its corresponding drop $i^-$, where $i^-$ follows $i^+$ in the route of $k$, but not necessarily immediately after it. Following this, it removes $i^+$ and $i^-$ from their clusters, with the caveat that $i^-$ may not belong to cluster $c'$. It halts the process of assigning customers to the route of vehicle $k$ when either the cluster $C_k$ becomes empty or no $i^+ \in C_k$ can be added without breaching the capacity and the duration constraints of vehicle

$k$. When this occurs, the construction process proceeds to the next vehicle. The process terminates when either all customers have been assigned to a route or none of the $m$ vehicles in $K$ can accommodate any additional customers.

#### 6.1.3. Insertion of unassigned stops

The solution obtained may not cover all stops, but this can be addressed by using insertion operations to repair the solution. If any stops remain unassigned, the construction process tries to insert them at every position in every route and selects the one that minimizes the impact on the solution's fitness. Once this insertion step is complete, the resulting solution becomes the current solution $X$ and the current best solution $X^*$ of DD-HALNS. Additionally, the construction process sets $\mathcal{L} = X$.

### 6.2. Maximum minimum utilities and fitness function

The fitness of a solution $X$ is evaluated based on its ability to adhere to the constraints imposed by the TWs', vehicles' capacity, and vehicles' time-out. It is measured using the fitness function $H(X)$, which considers both the total travel duration of $X$ (i.e., $T(X)$) and the penalties incurred due to violations of the constraints. These penalties are calculated by summing over all nodes in the network $\mathcal{N}$ and finding the minimum travel time to reach each node within its corresponding time window $W_{i\cdot}$, as well as summing over all vehicles in the fleet $\mathcal{K}$ for both capacity and time-out violations:

$$\sum_{i\cdot\in\mathcal{N}} \tau^{\rho}_{pi\cdot} + \sum_{k\in\mathcal{K}} \upsilon^{\rho}_{k} + \sum_{k\in\mathcal{K}} \omega^{\rho}_{k}. \tag{51}$$

The initial term in the expression computes the sum of penalties for all stops $i\cdot \in \mathcal{N}$ in the solution $X$ that violates the corresponding TW conditions. Specifically, for a given stop $i\cdot$ that is serviced during a TW $p \in W_{i\cdot}$, the penalty incurred is:

$$\tau_{i\cdot} = \min_{p_{i\cdot}}\{|a_{i\cdot} - l_{p_{i\cdot}}|, |a_{i\cdot} - u_{p_{i\cdot}}|\}.$$

The second term in the equation calculates the sum of load capacity violations, denoted as $\upsilon_k$, for all vehicles $k \in \mathcal{K}$. The value of $\upsilon_k$ is obtained by computing the maximum of zero and the difference between the total demand accumulated by the vehicle while serving a subset of stops $I_k$, up to a certain stop $i\cdot \in I_k$, and the maximum load capacity of the vehicle $L_k$. This is expressed as:

$$\upsilon_k = \max_{i\cdot\in I_k}\{0, (\sum_{j\cdot\in I_k}^{i\cdot} d_{j\cdot}) - L_k\}.$$

The third and final term in the equation adds up the time-out-of-depot violations, denoted as $\omega_k$, for all vehicles $k \in \mathcal{K}$. The value of $\omega_k$ is obtained by taking the maximum of zero and the difference between the actual departure time $d^k_D$ of the vehicle $k$ from the depot and the latest allowable departure time $a^k_D$, minus the total travel time $T_k$ of the vehicle $k$. This is expressed as: $\omega_k = \max\{0, (a^k_D - d^k_D) - T_k\}$. All three types of penalties are weighted by an exponent $\rho \geq 1$. Therefore, the fitness of a solution $X$ is given by the sum of the three penalty terms:

$$H(X) = T(X) + \sum_{i\in I} \min_{p\in W_i} \tau^{\rho}_{i} + \sum_{k\in\mathcal{K}} \upsilon^{\rho}_{k} + \sum_{k\in\mathcal{K}} \omega^{\rho}_{k}. \tag{52}$$

A solution $X$ is feasible only when it satisfies all the constraints and incurs no penalties, resulting in $H(X) = T(X)$. Conversely, if the solution $X$ violates any of the constraints, it becomes infeasible, which is reflected by a fitness value higher than the total travel duration $T(X)$. In other words, if $H(X) > T(X)$, then the solution $X$ is infeasible.

### 6.3. Multi-objective DD-HALNS

The multi-objective DD-HALNS algorithm aims to find a Pareto-dominating solution for three objectives, namely, total travel duration, customer satisfaction, and driver satisfaction. To achieve this, multi-objective DD-HALNS explores the neighborhood of the current solution $X$ to find a new solution $X'$ that improves upon the incumbent solution $\hat{X}$. A solution $X'$ is considered an improvement if it results in better values for all three objective functions, i.e., if $F(X') \geq F(\hat{X})$, $G(X') \geq G(\hat{X})$, and $H(X') \leq H(\hat{X})$. Here, $F$, $G$, and $H$ denote the objective functions for total travel duration, customer satisfaction, and driver satisfaction, respectively. The inequality signs indicate that the new solution must be at least as good as the incumbent solution for the first two objectives and must be better for the third objective. By continuously searching for and updating the incumbent solution with improving solutions, the multi-objective DD-HALNS algorithm aims to converge to a Pareto-dominating solution that achieves the best trade-off among the three objectives.

When $X'$ improves $\hat{X}$, it replaces both the current solution $X$ and the incumbent $\hat{X}$ and is added to the pool of best solutions, denoted as $\mathcal{L}$. If $\mathcal{L}$ is not yet full, $X'$ is simply appended to the list. However, if $\mathcal{L}$ is already full, the worst solution in the list is replaced with $X'$. This process allows $\mathcal{L}$ to be dynamically updated.

During the first phase of DD-HALNS, an intensification phase is applied to the first solution $X$ in $\mathcal{L}$. If this phase results in a better solution $X'$ than $X$, the search focus is shifted to $X'$ and the counter $\iota$ is reset to zero. However, if the intensification phase fails to improve $X$, $\iota$ is incremented by one, and the diversification phase is invoked. DD-HALNS iterates through these two phases for a fixed time limit $\bar{t}$. It is important to note that $\mathcal{L}$ plays a crucial role in both the intensification and diversification phases of DD-HALNS.

*6.3.1. Intensification*

Intensification is a technique used to search for an improved solution to a problem by exploring the neighborhood of the current solution, using a standard two-parameter SA. If $X'$ improves upon the current solution $X$, SA employs an initial temperature $T_0$, an initial solution $X_0$, and a cooling rate $\alpha \in (0,1)$ to transition from $X$ to $X'$ by exploring neighboring solutions. In cases where $X'$ does not improve upon $X$ but has a fitness of $H(X') < (1+\varsigma)H(X)$, where $\varsigma$ falls between 0.10 and 0.15 and $T$ represents the current temperature, SA uses a probability of $e^{-\frac{[H(X')-H(X)]}{T}}$ to determine whether to transition to $X'$. This ensures that $X'$ has a probability greater than $\frac{1}{2}$ of being accepted. SA sets a lower bound for $T_0$ when $H(X') - H(X_0) \leq (1+\varsigma)H(X_0)$ by using the inequality $e^{-\frac{[H(X')-H(X_0)]}{T_0}} \geq \frac{1}{2}$, which indicates that $T_0 \geq \frac{[H(X')-H(X_0)]}{ln(2)}$. Additionally, SA establishes a new temperature cycle every 1000 iterations and assigns a new cooling temperature $T$ to each cycle $\iota$ based on $T = T_0\alpha^{\iota}$, which provides a reasonable cooling schedule.

SA employs operators to generate neighboring solutions that aim to minimize the infeasibility of TWs, vehicle capacity, and vehicle time-out constraints in the current solution $X$. To achieve this, the approach incorporates "destroy" and "repair" operators, represented by $\Psi$, which eliminate random route sequences from $X$ and construct a new solution $X'$ by reintegrating the removed stops back into $X$. The selection of operator $\psi$, where $\psi = 1,\ldots,\Psi$, is determined using a uniform probability distribution $U(\psi) \in [0,1]$, which considers the success rate $\tau_\psi$ of the operator. The logarithmic scheme outlined in Algorithm 1 is applied to update the success rate of each operator.

At the beginning of the algorithm, all success rates $\tau_\psi$, where $\psi = 1,\ldots,\Psi$, are initialized to $ln(100)$, and all selection probabilities $U(\psi)$ are set to $\frac{\tau_\psi}{\tau}$, where $\tau$ represents the total weight of all $\Psi$ operators. Whenever an operator $\psi$ is selected to generate a neighboring solution $X'$ that improves upon $X$, its success rate $\tau_\psi$ is adjusted accordingly. In this case, the logarithmic scheme sets $\tau_\psi = ln(e^{\tau_\psi}+1)$. This process continues iteratively until the stopping condition is met. Using the logarithmic update method for the success rates of the DD-HALNS operators results in a smoother growth of the success rates of the $\Psi$ operators and a more moderate growth of their selection probability compared to using regular exponential or linear schemes.

**Algorithm 1** Multi-Objective DD-HALNS DARPMTW Logarithmic Scheme.

Step 0: Initialize;
**for** $\psi = 1,\ldots,\Psi$ **do**
  Set $\tau_\psi = ln(100)$;
**end for**
Step 1: Select;
Set $\tau = \sum_{\psi=1}^{\Psi} \tau_\psi$;
Select $\psi \in 1,\ldots,\Psi$ with $U(\psi) = \frac{\tau_\psi}{\tau}$;
Step 2: Apply & Update;
Get $X' = \psi(X)$;
**if** $F(X') \geq F(X)$ & $G(X') \geq G(X)$ & $H(X') \leq H(X)$ **then**
  $\tau_\psi = ln(e^{\tau_\psi}+1)$;
  $X = X'$;
**end if**;
**if** $F(X) \geq F(\hat{X})$ & $G(X) \geq G(\hat{X})$ & $H(X) \leq H(\hat{X})$ **then**
  $\hat{X} = X$;
**end if**;
**if** Stopping condition is not satisfied **then**
  go to Step 1;
**end if**

Herein, $\Psi = 6$ operators are applied.

- The destroy and repair-relocate (DRR) operator ($\psi = 1$) (Fig. 8), the destroy and repair-node (DRN) operator ($\psi = 2$) (Fig. 9), and then destroy and repair-arc (DRA) operator ($\psi = 3$) (Fig. 10) are single-route operators that are applied to each vehicle's route. These operators aim to reduce the infeasibility of a given route $k \in \mathcal{K}$ by identifying every drop $i^- \in I_k$ that is served before its matching pick-up $i^+ \in I_k$ and attempting to schedule $i^-$ after $i^+$ along the route of $k$. This can be achieved by either inserting $i^+$ after $i^-$, reversing the order of $i^-$ and $i^+$, or reversing their incident arcs. As all stops are eligible for relocation, these operators can make a given route more cost-effective and even feasible.
- The destroy–repair 2-route relocate (DR2R) operator ($\psi = 4$) (Fig. 11), the destroy–repair 2-route swap (DR2S) operator ($\psi = 5$) (Fig. 12), and the destroy–repair 2-route cross (DR2C) operator ($\psi = 6$) (Fig. 13) are operators that are applied to pairs of routes. These operators aim to eliminate infeasible drops.

To summarize, the destroy–repair operators and destroy–repair 2-route operators are heuristics commonly used in vehicle routing problems to improve the solution quality. These operators aim to remove infeasible drops and relocate stops to improve the overall

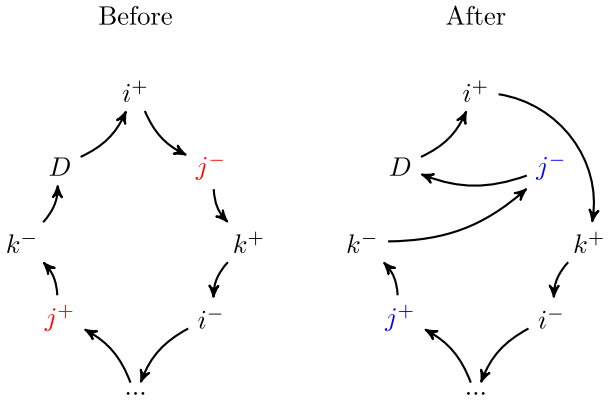


**Fig. 8.** Destroy–repair relocate operator (DRR).

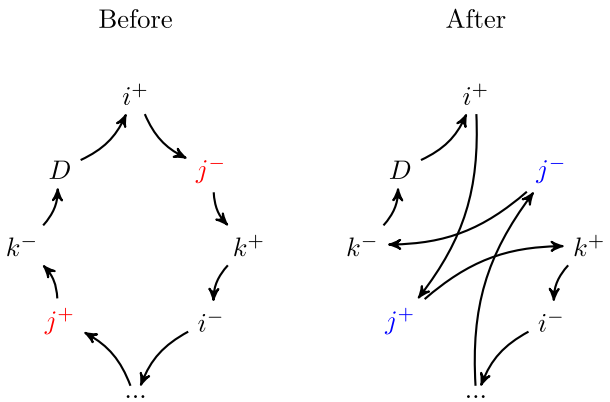


**Fig. 9.** Destroy–repair node operator (DRN).

fitness of the solution. The destroy–repair operators are applied to a single route, while the destroy–repair 2-route operators are applied to pairs of routes.

The destroy–repair operators include the destroy and repair-relocate (DRR) operator, the destroy–repair node (DRN) operator, and the destroy–repair arc (DRA) operator. DRR attempts to insert a drop after its matching pick-up along the same route, while DRN interchanges the positions of the drop and its matching pick-up in the route. DRA interchanges the order of the arcs incident to the drop and its matching pick-up. These operators have a complexity of $O(4mn^2)$.

The destroy–repair 2-route operators include the destroy–repair 2-route relocate (DR2R) operator, the destroy–repair 2-route swap (DR2S) operator, and the destroy–repair 2-route cross (DR2C) operator. DR2R attempts to relocate a drop to a different route or a pick-up to the current route, while DR2S exchanges drops and their matching pick-ups between two different routes, and DR2C attempts to move groups of stops to the same route. These operators have a complexity of $O(4m^2n^2)$.

### 6.3.2. Diversification

DD-HALNS uses the *best reset* operator whenever it fails to improve $X^*$ during $\iota$ consecutive temperature cycles, where $\iota$ is greater than the threshold used for *genetic crossover*. The *best reset* operator resets the current solution to the best solution recorded in $\mathcal{L}$, which is the pool of best solutions obtained during the search. The *best reset* operator can help to escape from local optima by moving the search to a completely different area of the search space, where the current solution has not yet been explored.

In summary, DD-HALNS uses large neighborhood diversification techniques, such as *ruin-and-recreate*, *genetic crossover*, and *best reset*, to escape from sub-optimal valleys and explore new regions of the search space. These techniques move the search to neighborhoods that lie far from the local neighborhoods explored during the intensification phase, which can help to avoid getting trapped in local optima and find better solutions (see Fig. 14). .

Whenever the genetic crossover fails to enhance the optimal solution $X^*$ after a predetermined number of iterations, DD-HALNS employs the "best reset" operator, which involves replacing the current solution $X$ with one of the solutions in $\mathcal{L}$. The size of $\mathcal{L}$ is selected to be large enough to enable DD-HALNS to escape local optima and prevent cycling, but small enough to permit revisiting the neighborhoods nearest to the global Pareto optimal solution.

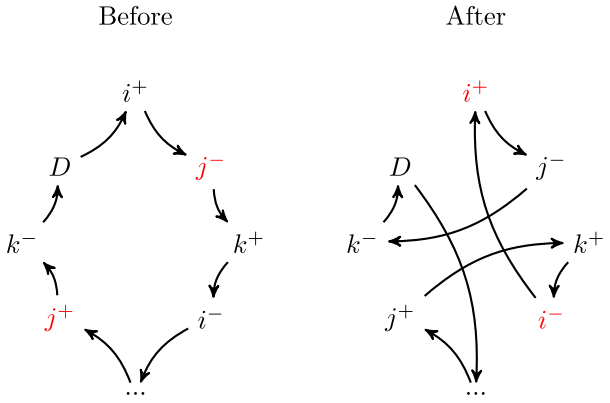


**Fig. 10.** Destroy–repair arc operator (DRA).

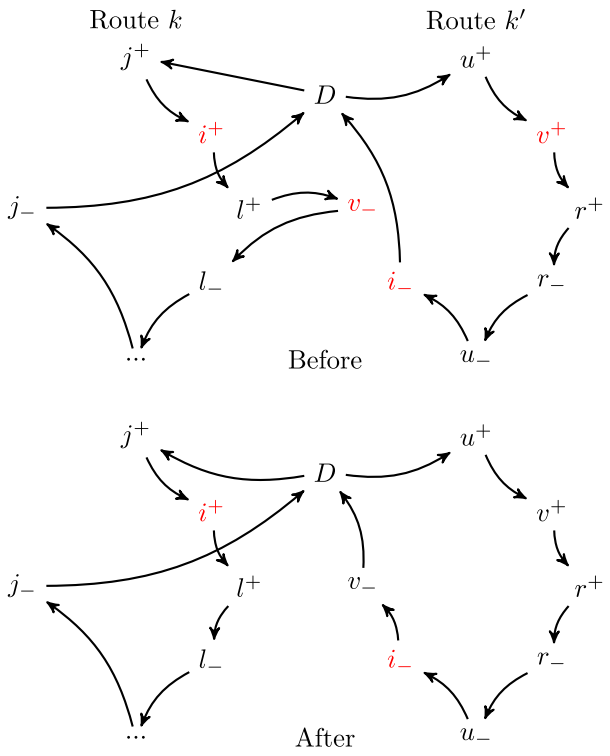


**Fig. 11.** Destroy–repair 2-route relocate (DR2R).

## 7. Computational results

The implementation of DD-HALNS was carried out using MS Visual C++ and executed on 64-bit workstations with 11th Gen Intel(R) Core(TM) i9-11900H at 2.50 GHz under MS Windows 11 pro. The performance of DD-HALNS was evaluated on both real-world scenarios and new benchmark instances.

### *7.1. Real-life instances*

RouteGenius provided real-life instances consisting of two sets of ten consecutive days each, i.e. two weeks, in Vancouver and surrounding suburbs, including Surrey, Coquitlam, North Vancouver, and Richmond. The data includes the geo-coded addresses of all pickups and drop-offs, converted into latitude and longitude coordinates using Google Cloud API services. The travel distances and times between each pair of points were obtained from these geo-codes. The instances range from 30 to 100 stops and contain

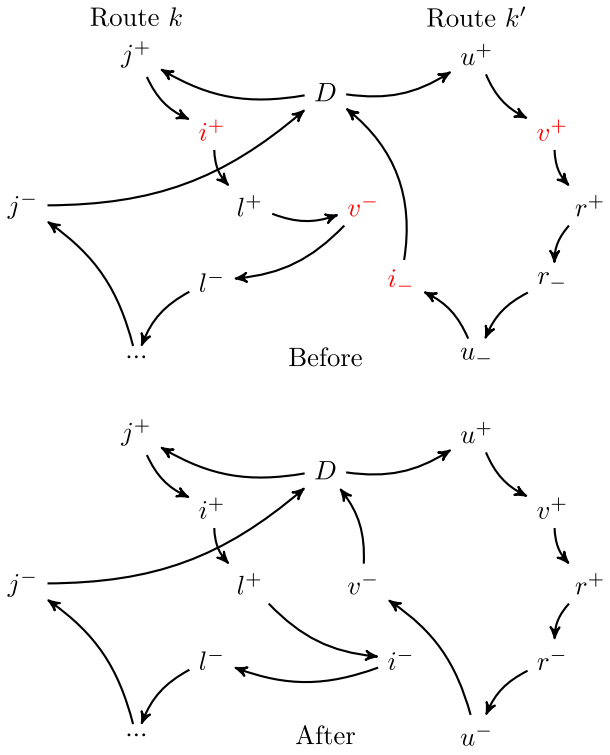


**Fig. 12.** Destroy–repair inter-route swap (DR2S).

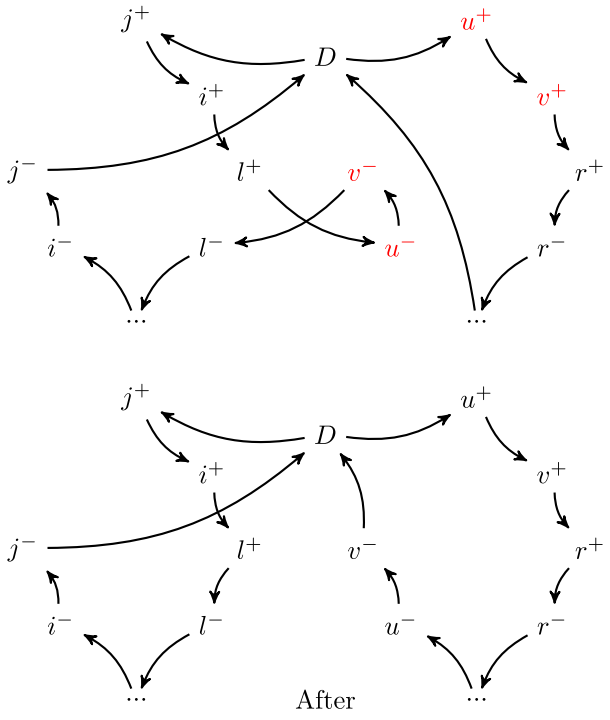


**Fig. 13.** Destroy–repair inter-route cross (DR2C).

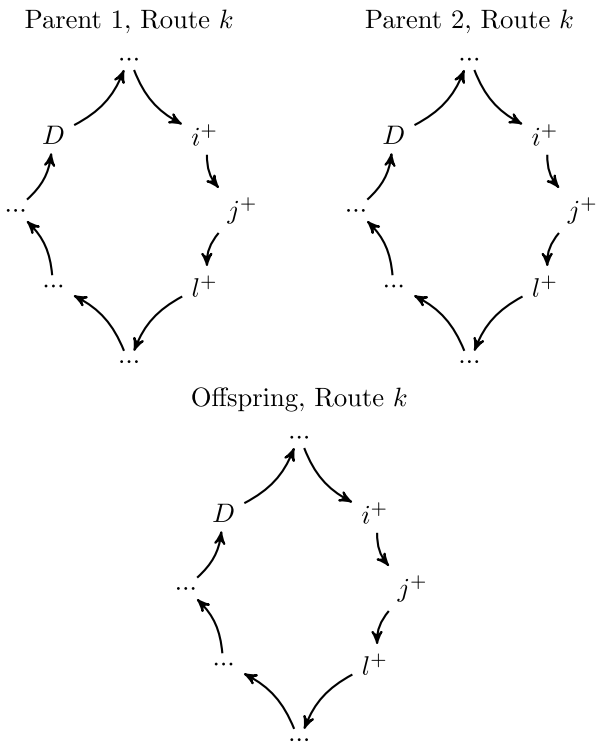


**Fig. 14.** Two parents crossover.

demands for impaired transport by a public–private service company, with 25 to 35 service requests per business day. The peak demand is typically on Thursdays and the lowest on Fridays.

The company has 12 to 15 medium-sized shuttle buses that are adapted to reduced mobility customers with standard passage corridors between seats and wheelchairs storage spaces. This limits the maximum capacity of each shuttle to eight passengers. A given route cannot exceed a maximum duration of 360 min, as a shuttle driver cannot remain on duty for more than six consecutive hours. The TW intervals for the pickups are closed and non-overlapping, while the drop-offs have tight TWs with no more than one hour span. Each customer's maximum transit time is 90 min, and the service time at each stop is 5 to 10 min. The first $n$ stops represent the pickups, and the last $n$ stops represent their corresponding drop-offs.

Table 5 presents the results of the experiments conducted on these instances. The "Day" column indicates the day of the instance, while the "$n$" and "$m$" columns show the number of customers and vehicles used, respectively. The "$H(X)$" column indicates the total duration in minutes of the routes scheduled by the company's current optimization software. The "$H(X^*)$" and "Avg" columns show the best and average solution values obtained by ALNS (without GA) and H-ALNS, respectively, along with the average execution times (in seconds). The "Time" column reports the average execution time for DD-HALNS, and the "$\%\delta_H$" column shows the percentage gap between the best solution obtained with DD-HALNS and the company's overall route duration.

The results demonstrate that DD-HALNS outperforms ALNS and H-ALNS in terms of solution quality and general results. DD-HALNS saves around 0.46% on average routing duration compared to the business solution currently deployed and obtained by expert human operators. These results were achieved in an average execution time of about 36 s, providing the company with significant flexibility to adapt to changes in circumstances. Fig. 15 illustrates the routing optimization solution obtained on Monday of Week 2.

The results presented in Table 6 demonstrate the average values of the Pareto frontier objectives, namely $H(\hat{X})$, $G(\hat{X})$, and $F(\hat{X})$. These results show that there is an average deterioration of the total duration objective $H(\hat{X})$ by 11.4% compared to the optimal solution $H(X^*)$. However, this deterioration allows for an average improvement of the minimum driver utility $G(\hat{X})$ by 229.5% compared to $G(X^*)$. The minimum customer utility $F(X^*)$ is the same as $F(\hat{X})$ because the customer utility is based on the distance to the unique depot for the drivers.

The Pareto frontier in Fig. 16 represents the trade-off between the two conflicting objectives: minimizing the total duration of the routes and maximizing the driver utility. The Break Point represents the point on the Pareto frontier where further improvements in the driver's utility come at a high cost in terms of the total duration of the routes. The Pareto solutions to the left of the break point have a lower total duration but lower driver utility, while the solutions to the right have a higher total duration but higher driver utility. There is a significant improvement in driver utility as we move toward the break point from the left. However, after

**Table 5**
Results on real-life instances.

| Week | Instance | | | ALNS | H-ALNS | DD-HALNS | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Day | $n$ | $m$ | $H(X^*)$ | $H(X^*)$ | $H(X^*)$ | $G(X^*)$ | $F(X^*)$ | $Avg_H$ | Time | $\%\delta_H$ |
| 1 | Mon | 25 | 7 | 1578.18 | 1561.42 | 1561.42 | 0.066 | 0.008 | 1597.66 | 14 | 0.00 |
| | Tue | 24 | 7 | 1629.88 | 1586.15 | 1566.85 | 0.028 | 0.008 | 1641.23 | 41 | −1.22 |
| | Wed | 24 | 7 | 1587.15 | 1574.33 | 1570.59 | 0.109 | 0.010 | 1651.93 | 59 | −0.24 |
| | Thu | 37 | 9 | 2286.73 | 2213.38 | 2175.80 | 0.073 | 0.013 | 2310.33 | 64 | −1.70 |
| | Fri | 19 | 4 | 1291.68 | 1291.68 | 1291.68 | 0.113 | 0.010 | 1308.66 | 11 | 0.00 |
| 2 | Mon | 24 | 4 | 1170.07 | 1164.79 | 1164.79 | 0.154 | 0.015 | 1182.39 | 27 | 0.00 |
| | Tue | 26 | 5 | 1461.20 | 1451.55 | 1439.77 | 0.031 | 0.015 | 1504.73 | 26 | −0.82 |
| | Wed | 28 | 6 | 1734.75 | 1710.61 | 1710.61 | 0.091 | 0.009 | 1765.11 | 27 | 0.00 |
| | Thu | 40 | 10 | 2619.03 | 2518.58 | 2502.55 | 0.089 | 0.009 | 2626.74 | 80 | −0.64 |
| | Fri | 15 | 5 | 1079.74 | 1079.74 | 1079.74 | 0.095 | 0.011 | 1089.56 | 13 | 0.00 |
| | Avg | 26.2 | 6.4 | 1643.84 | 1615.23 | 1606.38 | 0.085 | 0.0108 | 1667.81 | 36.2 | −0.46 |

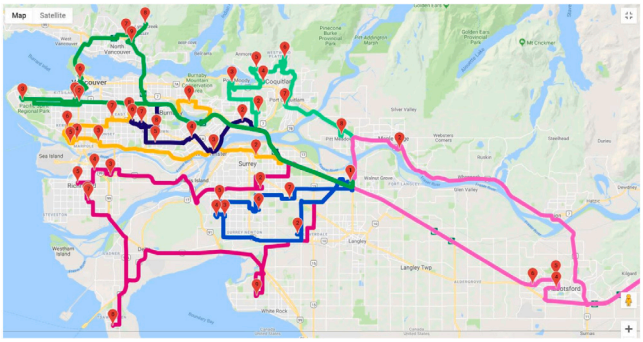


**Fig. 15.** Vancouver & region routing, monday, week 2.

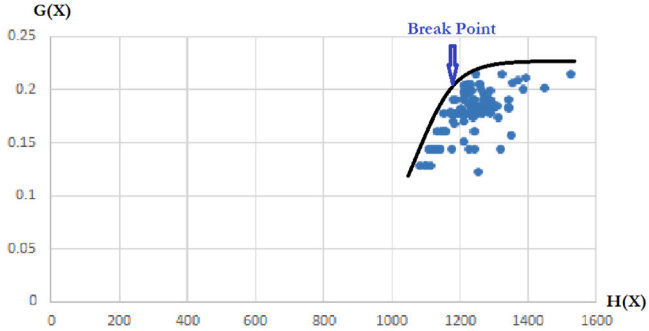


**Fig. 16.** Pareto frontier, friday, week 2.

reaching the Break Point, there is little improvement in driver utility, even if we are willing to accept a significant deterioration in the total duration of the routes. Therefore, it is advisable to choose a solution that lies around the Break Point to achieve a good balance between the two objectives. Overall, the Pareto frontier analysis provides a useful tool to visualize the trade-off between different objectives and to choose a solution that meets the requirements of all stakeholders.

### *7.2. New DARPMTW benchmark instances*

The new benchmark set consists of 20 DARPMTW instances obtained from the original DARPTW instances proposed by Cordeau and Laporte (Cordeau and Laporte, 2003). These instances have $n = 24$ to $n = 144$ customers. The new instances were obtained by splitting each stop time window randomly into 1 to 4 time windows, and random width between 200 and 700 time units.

In Table 7, column "Inst". indicates the name of the new instance, column "$n$" indicates the number of customers, and column "$m$" indicates the number of vehicles used. The columns "$D^*$" and "$Avg_D$" indicate the minimum and average total distance found by DD-HALNS. The columns "$H^*$" and "$Avg_H$" indicate the minimum and average route duration. The columns "$G^*$" and "$Avg_G$"

**Table 6**
DD-HALNS multi-objective results.

| 1 | Day | $H(\hat{X})$ | $G(\hat{X})$ | $F(\hat{X})$ | $\Delta_H$ | $\Delta_G$ |
|---|---|---|---|---|---|---|
| | Mon | 1591.97 | 0.110 | 0.008 | 1.96 | 66.26 |
| | Tue | 1614.2 | 0.245 | 0.008 | 3.02 | 763.66 |
| | Wed | 1643.03 | 0.237 | 0.010 | 4.61 | 115.52 |
| | Thu | 2408.44 | 0.194 | 0.013 | 10.69 | 166.85 |
| | Fri | 1403.73 | 0.142 | 0.010 | 8.67 | 25.31 |
| 2 | Day | $H(\hat{X})$ | $G(\hat{X})$ | $F(\hat{X})$ | $\Delta_H$ | $\Delta_G$ |
| | Mon | 1209.97 | 0.175 | 0.015 | 3.88 | 13.60 |
| | Tue | 1839.82 | 0.258 | 0.015 | 27.79 | 744.43 |
| | Wed | 2137.46 | 0.259 | 0.009 | 24.95 | 183.92 |
| | Thu | 2953.08 | 0.198 | 0.009 | 18.00 | 121.41 |
| | Fri | 1191.96 | 0.184 | 0.011 | 10.39 | 94.10 |
| | Avg | 1799.37 | 0.200 | 0.011 | 11.40 | 229.50 |

**Table 7**
Results on new DARPMTW Instances.

| Inst. | $n$ | $m$ | $D^*$ | $Avg_D$ | $H^*$ | $Avg_H$ | $G^*$ | $Avg_G$ | $H(\hat{X})$ | $G(\hat{X})$ | $F(\hat{X})$ | Time | $\Delta_H$ | $\Delta_G$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| prmtw01 | 24 | 2 | 130.73 | 150.92 | 610.73 | 630.92 | 1.17432 | 0.90851 | 625.89 | 1.02031 | 0.04687 | 4.5 | 2.48 | −13.11 |
| prmtw02 | 48 | 4 | 248.83 | 283.21 | 1208.83 | 1243.21 | 1.23645 | 1.00967 | 1247.06 | 1.15449 | 0.04236 | 15.4 | 3.16 | −6.63 |
| prmtw03 | 72 | 6 | 490.26 | 550.19 | 1936.26 | 1996.19 | 1.14188 | 1.01251 | 1993.13 | 1.11783 | 0.04029 | 45.3 | 2.94 | −2.11 |
| prmtw04 | 96 | 7 | 576.83 | 641.00 | 2496.83 | 2561.00 | 1.19564 | 1.01486 | 2573.50 | 1.19339 | 0.04530 | 65.3 | 3.07 | −0.19 |
| prmtw05 | 120 | 10 | 650.70 | 709.77 | 3070.61 | 3119.77 | 1.38661 | 1.09530 | 3148.51 | 1.29149 | 0.04147 | 107.2 | 2.54 | −6.86 |
| prmtw06 | 144 | 12 | 844.02 | 930.79 | 3727.24 | 3810.79 | 1.23485 | 1.021502 | 3834.14 | 1.18879 | 0.04217 | 156.8 | 2.87 | −3.73 |
| prmtw07 | 36 | 3 | 209.71 | 245.41 | 929.71 | 965.41 | 1.41381 | 0.966111 | 956.86 | 1.06020 | 0.04274 | 11.0 | 2.92 | −25.01 |
| prmtw08 | 72 | 6 | 435.96 | 507.89 | 1904.15 | 1947.89 | 1.47398 | 1.28518 | 1939.51 | 1.35439 | 0.04520 | 46.0 | 1.86 | −8.11 |
| prmtw09 | 108 | 8 | 666.24 | 735.09 | 2885.76 | 2895.09 | 1.61837 | 1.48785 | 2902.81 | 1.56521 | 0.04292 | 110.1 | 0.59 | −3.28 |
| prmtw10 | 144 | 10 | 915.86 | 984.66 | 3848.41 | 3864.66 | 1.74311 | 1.64858 | 3950.64 | 1.71749 | 0.03944 | 210.6 | 2.66 | −1.47 |
| prmtw11 | 24 | 2 | 126.54 | 148.44 | 606.54 | 628.44 | 1.22907 | 0.92632 | 627.03 | 1.03245 | 0.04687 | 4.5 | 3.38 | −16.00 |
| prmtw12 | 48 | 4 | 253.78 | 287.46 | 1213.78 | 1247.46 | 1.23833 | 1.00915 | 1242.12 | 1.17679 | 0.04236 | 20.9 | 2.33 | −4.97 |
| prmtw13 | 72 | 5 | 494.18 | 546.29 | 1934.18 | 1986.29 | 1.44247 | 1.27495 | 1958.96 | 1.27841 | 0.04029 | 49.9 | 1.28 | −11.37 |
| prmtw14 | 96 | 9 | 602.78 | 649.57 | 2522.78 | 2569.57 | 1.29799 | 1.02478 | 2595.04 | 1.29799 | 0.04530 | 75.5 | 2.86 | 0.00 |
| prmtw15 | 120 | 10 | 668.09 | 712.41 | 3068.09 | 3112.41 | 1.45892 | 1.25157 | 3106.67 | 1.45892 | 0.04147 | 116.9 | 1.26 | 0.00 |
| prmtw16 | 144 | 11 | 848.57 | 918.53 | 3728.58 | 3798.53 | 1.52248 | 1.38446 | 3832.03 | 1.50535 | 0.04217 | 191.8 | 2.77 | −1.13 |
| prmtw17 | 36 | 3 | 213.66 | 246.23 | 933.66 | 966.23 | 1.24217 | 0.93045 | 962.01 | 1.05645 | 0.04274 | 10.2 | 3.04 | −14.95 |
| prmtw18 | 72 | 6 | 432.53 | 507.36 | 1872.53 | 1947.36 | 1.6682 | 1.30252 | 1943.95 | 1.39844 | 0.04520 | 43.2 | 3.81 | −16.17 |
| prmtw19 | 108 | 8 | 674.66 | 728.14 | 2867.43 | 2888.14 | 1.63146 | 1.48506 | 2885.90 | 1.52143 | 0.04292 | 110.6 | 0.64 | −6.74 |
| prmtw20 | 144 | 10 | 925.68 | 978.60 | 3805.68 | 3858.60 | 1.8072 | 1.62602 | 3874.36 | 1.54584 | 0.03944 | 196.5 | 1.80 | −14.46 |
| Avg | 86.4 | 6.80 | 520.48 | 573.10 | 2258.59 | 2301.90 | 1.40787 | 1.18327 | 2310.01 | 1.29678 | 0.04288 | 79.6 | 2.41 | −7.82 |

indicate the maximum and average driver utility. Columns "$H(\hat{X})$", "$G(\hat{X})$" and "$F(\hat{X})$" indicate the total duration, the minimum driver utility and the minimum customer utility of the chosen break point non-dominated Nash equilibrium routing solution $X^*$. The column "$Time$" indicates the average execution time in seconds on 30 consecutive runs of multi-objective DD-HALNS. Finally, the columns $\Delta_H$ and $\Delta_G$ display the gap (in percentage) between the break point objectives and $H^*$ and $G^*$, respectively. It should also be noted that the minimum customer utility $F^*$ is the same as $F(\hat{X})$ because the customer utility is based on the distance to the unique depot for the drivers.

Table 7 shows that the average computation time is around 79.6 s for an average number of 170 requests using an average number of 6.8 vehicles. It also shows that the chosen break point objectives, namely $H(\hat{X})$, $G(\hat{X})$, and $F(\hat{X})$ are on average 2.41% and 7.82% from the single objective best duration minimization or driver utility maximization solutions.

## 8. Conclusion

In summary, this paper proposes a comprehensive data-driven approach to solving the DARPMTW problem. It starts with predicting the problem's uncertain parameters, such as travel and service times, using artificial neural networks. Then, it applies a multi-objective framework to minimize the total route duration, maximize the minimal drivers' satisfaction, and maximize the minimal customers' satisfaction. It shows that non-dominated Pareto feasible solutions satisfy the Nash equilibrium conditions of the company-drivers-customers' game theoretic model. Finally, it proposes a Data-Driven Hybrid Adaptive Large Neighborhood Search algorithm, which selects local search operators based on their success rates and uses hybridization techniques to improve performance. The experimental results show that the proposed algorithm is effective in reducing costs and improving driver satisfaction. The paper exemplified a way to estimate travel and service times, and showed how data from other cities could be analyzed to obtain those estimates. Future research will further investigate its use on real-time dispatching, its transferability and scalability.

**Acknowledgments**

This work was partly funded by King Fahd University of Petroleum and Minerals under the Interdisciplinary Research Center for Smart Mobility and Logistics research grant INML2107. This support is gratefully acknowledged. The data obtained from RouteGenius Canada was anonymized and contained no personal information that could identify the riders.

## Appendix A. Proof of Proposition 1

Every Nash equilibrium satisfies

- primal feasibility for every customer $i \in \mathcal{I}$ and for every driver $k \in \mathcal{K}$; that is, conditions (2)–(7) and (9)–(11), respectively;
- dual feasibility given by conditions (28)–(32) and (33)–(36) for every customer $i \in \mathcal{I}$ and every driver $k \in \mathcal{K}$, respectively; and
- complementary slackness conditions (41)–(50).

Therefore, every Nash equilibrium satisfies the conditions of Proposition 1. ■

## Appendix B. Proof of Proposition 2

Let $(X^*, Y^*) = ((x_{i\cdot}^{k^*}), (y_{i\cdot}^{k^*}))_{i\cdot \in \mathcal{N}, k \in \mathcal{K}}$ be a vector of the proportions $x_{i\cdot}^k$, $i\cdot$ and $y_{i\cdot}^k$, $i\cdot \in \mathcal{N}$, $k \in \mathcal{K}$, that request $i\cdot$ receives from driver $k$, such that

- $X^*$ satisfies conditions (2)–(7) for every customer $i \in \mathcal{I}$,
- $Y^*$ satisfies conditions (9)–(11) for every driver $k \in \mathcal{K}$, and
- $x_{i\cdot}^{k^*}$ and $y_{i\cdot}^{k^*}$ equal zero or one.

Herein, we detail a three-part construction proof that shows the existence of feasible dual variables such that $(X^*, Y^*)$ is a Nash equilibrium of the DARP game. The first part proves that there are always feasible dual variables $\delta_{i\cdot}$, $\alpha_{i\cdot}^k$ and $\beta_i^k$ when matrices $F_i^k$ are non-negative. The second part proves that there are always feasible dual variables $\lambda_{i\cdot}^k$ when matrices $G_i^k$ are non-negative. The third and last part concludes that $(X^*, Y^*)$ is a Nash equilibrium.

**Part** 1. Let $k^* \in \mathcal{K}$ be the index of the driver satisfying the request of customer $i$, $i \in \mathcal{I}$, and $k^0 \in \mathcal{K} \setminus \{k^*\}$ be the index of any other driver; that is, $x_{i^+}^{k^*} = x_{i^-}^{k^*} = 1$, and $x_{i^+}^{k^0} = x_{i^-}^{k^0} = 0$.

- The complementary slackness conditions (37) and (38) are satisfied $\forall\ k^0 \in \mathcal{K} \setminus \{k^*\}$ because $x_{i^+}^{k^0} = x_{i^-}^{k^0} = 0$.
- When $k = k^*$, $x_{i^+}^{k^*} = y_{i^+}^{k^*} = 1$, and $x_{i^-}^{k^*} = y_{i^-}^{k^*} = 1$. Thus, the satisfaction of the complementary slackness conditions given by Eqs. (37) and (38) requires that

  $$(\delta_{i^+} + \alpha_{i^+}^k + \beta_i^k - f_{i^+}^k y_{i^+}^k) = 0,$$

  and

  $$(\delta_{i^-} + \alpha_{i^-}^k - \beta_i^k - f_{i^-}^k y_{i^-}^k) = 0.$$

  These two conditions are equivalent to

  $$\delta_{i^+} + \alpha_{i^+}^k + \beta_i^k = f_{i^+}^k y_{i^+}^k,$$

  and

  $$\delta_{i^-} + \alpha_{i^-}^k - \beta_i^k = f_{i^-}^k y_{i^-}^k.$$

  Because $f_{i^+}^{k^*} \geq 0$ and $f_{i^-}^{k^*} \geq 0$, one feasible solution would set

  $$\alpha_{i^+}^{k^*} = f_{i^+}^{k^*},\ \alpha_{i^-}^{k^*} = f_{i^-}^{k^*},\ \delta_{i^+} = \delta_{i^-} = \beta_i^{k^*} = 0.$$

- Substituting these dual variable values in conditions (29) and (30) for $k = k^0$ yields

  $$\delta_{i^+} + \alpha_{i^+}^{k^0} + \beta_i^{k^0} \geq 0,$$

  and

  $$\delta_{i^-} + \alpha_{i^-}^{k^0} - \beta_i^{k^0} \geq 0$$

  (because $y_{i^+}^{k^0} = y_{i^-}^{k^0} = 0$). Any $\beta_i^{k^0} \geq 0$ satisfies these two conditions. For example, setting $\alpha_{i^+}^{k^0} = \alpha_{i^-}^{k^0} = \beta_i^{k^0} = 0$ gives a feasible solution (because $\delta_{i^+}$ and $\delta_{i^-}$ are already set to 0).

**Part** 2. Let $i^* \in \mathcal{I}_k$ be the index of any customer $i$ whose request is fully satisfied by driver $k$, $k \in \mathcal{K}$, and $i^0 \in \mathcal{I} \setminus \mathcal{I}_k$ be the index of any other customer who is not served by $k$. Hence, $y_{i^{*+}}^k = y_{i^{*-}}^k = 1$, and $y_{i^{0+}}^k = y_{i^{0-}}^k = 0$.

- The complementary slackness conditions (39) and (40) are automatically satisfied for $i = i^0 \in \mathcal{I} \setminus \mathcal{I}_k$ because $y^k_{i^{0+}} = y^k_{i^{0-}} = 0$.
- For $i^* \in \mathcal{I}_k$, $y^k_{i^{*+}} = y^k_{i^{*-}} = 1$. Thus, the complementary slackness conditions (39) and (40) reduce to

$$\lambda^k_{i^+} - x^k_{i^+} g^k_{i^+} = 0,$$

and

$$\lambda^k_{i^-} - x^k_{i^-} g^k_{i^-} = 0.$$

Because $g^k_{i^{*+}} \geq 0$ and $g^k_{i^{*-}} \geq 0$, a feasible solution is $\lambda^k_{i^{*+}} = g^k_{i^{*+}}$ and $\lambda^k_{i^{*-}} = g^k_{i^{*-}}$.
- Substituting these dual variables values in conditions (34) and (35) for $i = i^0 \in \mathcal{I} \setminus \mathcal{I}_k$ (i.e., for $x^k_{i^{0+}} = x^k_{i^{0-}} = 0$) reveals that any solution where $\lambda^k_{i^{0+}} \geq 0$ and $\lambda^k_{i^{0-}} = 0$ is feasible.

**Part** 3. The solution $(X^*, Y^*)$ satisfies primal feasibility, dual feasibility, and complementary slackness conditions. Therefore, $(X^*, Y^*)$ is a Nash equilibrium of the DARP game, with possibly:

$$\begin{array}{lll}
x^{k^*}_{i^+} = y^{k^*}_{i^+}, & x^{k^*}_{i^-} = y^{k^*}_{i^-} & \\
\alpha^{k^*}_{i^+} = f^{k^*}_{i^+}, & \alpha^{k^*}_{i^-} = f^{k^*}_{i^-}, & \beta^{k^*}_{i} = 0, \\
\alpha^{k^0}_{i^+} = 0, & \alpha^{k^0}_{i^-} = 0, & \beta^{k^0}_{i} = 0, \\
\delta_{i^+} = 0, & \delta_{i^-} = 0, & \\
\lambda^k_{i^{*+}} = g^k_{i^{*+}}, & \lambda^k_{i^{*-}} = g^k_{i^{*-}}, & \\
\lambda^k_{i^{0+}} \geq 0 & \lambda^k_{i^{0-}} = 0. &
\end{array}$$

■

## References


Beheshti, A.K., Hejazi, S.R., Alinaghian, M., 2015. The vehicle routing problem with multiple prioritized time windows: A case study. Comput. Ind. Eng. 90, 402–413.

Belhaiza, S., A data driven hybrid heuristic for the dial-a-ride problem with time windows. In: IEEE Symposium Series on Computational Intelligence SSCI 2017, IEEE SSCI Proceedings. pp. 1–8.

Belhaiza, S., 2018. A game theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows. IEEE Syst. J. 12, 1251–1262.

Belhaiza, S., 2019. A hybrid adaptive large neighborhood heuristic for a real-life dial-a-ride problem. Algorithms 12, 39.

Belhaiza, S., Hansen, P., Laporte, G., 2014. Hybrid variable neighborhood Tabu search heuristic for the vehicle routing problem with multiple time windows. Comput. Oper. Res. 52, 269–281.

Belhaiza, S., M'Hallah, R., Ben Brahim, G., Laporte, G., 2019. Three multi-start data-driven evolutionary heuristics for the vehicle routing problem with multiple time windows. J. Heuristics 25, 485–515.

Chevrier, R., Liefooghe, A., Jourdan, L., Dhaenens, C., 2012. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport. Appl. Soft Comput. 12, 1247–1258.

Cordeau, J.F., Laporte, G., 2003. A Tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. B 37, 579–594.

Diana, M., Dessouky, M.M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transp. Res. B 38, 539–557.

Drakoulis, R., Bellotti, F., Bakas, I., Berta, R., Paranthaman, P.K., Dange, G.R., Lytrivis, P., Pagle, K., De Gloaria, A., Admitis, A., 2018. A gamified flexible transport service for on-demand public transport. IEEE Trans. Intell. Transp. Syst. 19, 921–933.

Favaretto, D., Moretti, E., Pellegrini, P., 2007. Ant colony system for a VRP with multiple time windows and multiple visits. J. Interdiscip. Math. 10, 263–284.

Fu, L., 2002. A simulation model for evaluating advanced dial-a-ride paratransit systems. Transp. Res. A 36, 291–307.

Fu, L., 2003. Analytical model for paratransit capacity and quality-of-service analysis. Transp. Res. Rec. 1841, 81–89.

Guerriero, F., Pezzella, F., Pisacane, O., Trollinin, L., 2014. Multi-objective optimization in dial-a-ride public transport. Transp. Res. Procedia 3, 299–308.

Hagan, M.T., Demuth, H.B., Beale, M.H., Jesus, O.D., 2018. Neural Network Design, second ed. Amazon.

Hoogeboom, M., Dullaert, W., 2019. Vehicle routing with arrival time diversification. European J. Oper. Res. 275 (1), 93–107.

Hu, T.Y., Zheng G.C, T.Y., 2017. A multi-objective model for dial-a-ride problems with service quality and eco-efficiency. In: IEEE 20th International Conference on Intelligent Transport Systems. ITSC.

Léhudé, F., Masson, R., Parragh, S.N., Péton, O., Tricoire, F., 2014. A multi-criteria large neighbourhood search for the transport of disabled people. J. Oper. Res. Soc. 65, 983–1000.

Madsen, O.B.G., Raven, H.F., Rygaard, J.M., 2015. A branch-and-cut algorithm for a realistic dial-a-ride problem. Transp. Res. B 81, 267–288.

Marković, N., Kim, M.E., Schonfeld, P., 2016. Statistical and machine learning approach for planning dial-a-ride systems. Transp. Res. A 89, 41–55.

Marković, N., Milinković, S., Schonfeld, P., Drobnjak, Z., 2013. Planning dial-a-ride services: Statistical and meta-modeling approach. Transp. Res. Rec. 2352, 120–127.

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. Transp. Res. E 96, 60–80.

Masson, R., Léhudé, F., Péton, O., 2013. The dial-a-ride problem with transfer. Comput. Oper. Res. 41, 12–23.

M'Hallah, R., 2020. Combining exact methods to construct effective hybrid approaches to vehicle routing. In: Smith, A.-E. (Ed.), Women in Industrial and Systems Engineering. Women in Engineering and Science. Springer, Cham.

Molenbruch, Y., Braekers, K., Caris, A., Berghe, G.V., 2017. Multi-directional local search for a bi-objective dial-a-ride problem in patient transport. Comput. Oper. Res. 77, 58–71.

Muelas, S., LaTorre, A., Pena, J.M., 2015. A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios. Transp. Res. C 54, 110–130.

Neto, A.F., Cardoso, P.A., 2017. Dynamic vehicle programming and routing system applied to wheelchair transport. IEEE Latin Am. Trans. 15, 317–323.

Noda, I., Ohta, M., Shinoda, K., Kumada, Y., Nakashima, H., 2003. Evaluation of usability of dial-a-ride systems by social simulation. In: Fourth International Workshop on Multi-Agent-Based Simulation. MABS, pp. 167–181.

Parragh, S., Doerner, K., Hartl, R., 2010. Variable neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 37, 1129–1138.

Parragh, S., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 40, 490–497.

Quadrifoglio, L., Dessouky, M., Ordóñez, F., 2008. A simulation study of demand responsive transit system design. Transp. Res. A 42, 718–737.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Schilde, M., Doerner, K.F., Hartl, R.F., 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. European J. Oper. Res. 238, 18–30.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming), Vol. 1520. pp. 417–431.

Zidi, I., Mesghouni, K., Zidi, K., Ghedira, K., 2012. A multi-objective simulated annealing for the multi-criteria dial a ride problem. Eng. Appl. Artif. Intell. 25, 1121–1131.