PDPTW-DB: MILP-Based Offline Route Planning for PDPTW with Driver Breaks

Agrima Khanna
Vanderbilt University
Nashville, Tennessee, United States
agrima.khanna@vanderbilt.edu

Fangqi Liu
Vanderbilt University
Nashville, Tennessee, United States
fangqi.liu@vanderbilt.edu

Samir Gupta
Vanderbilt University
Nashville, Tennessee, United States
samir.a.gupta@vanderbilt.edu

Sophie Pavia
Vanderbilt University
Nashville, Tennessee, United States
sophie.r.pavia@vanderbilt.edu

Ayan Mukhopadhyay
Vanderbilt University
Nashville, Tennessee, United States
ayan.mukhopadhyay@vanderbilt.edu

Abhishek Dubey
Vanderbilt University
Nashville, Tennessee, United States
abhishek.dubey@vanderbilt.edu

Abstract

The Pickup and Delivery Problem with Time Windows (PDPTW) involves optimizing routes for vehicles to meet pickup and delivery requests within specific time constraints, a challenge commonly faced in logistics and transportation. Microtransit, a flexible and demand-responsive service using smaller vehicles within defined zones, can be effectively modeled as a PDPTW. Yet, the need for driver breaks—a key human constraint—is frequently overlooked in PDPTW solutions, despite being necessary for regulatory compliance. This study presents a novel mixed-integer linear programming formulation for the Pickup and Delivery Problem with Time Windows and Driver Breaks (PDPTW-DB). To the best of our knowledge this formulation is the first to consider mandatory periodic driver breaks within optimized Microtransit routes. The proposed model incorporates regulatory compliant break scheduling directly within the vehicle routing optimization framework. By considering driver break requirements as an integral component of the optimization process, rather than as a post-processing step, the model enables the generation of routes that respect hours of service regulations while minimizing operational costs. This integrated approach facilitates the generation of schedules that are operationally efficient and prioritize driver welfare through driver breaks. We work with a public transit agency from the southern USA, and highlight the specific nuances of driver break optimization, and present a Pickup and Delivery Problem with Time Windows formulation for optimizing Microtransit operations and scheduling driver breaks. We validate our approach using real-world data from the transit agency. Our results validate our formulation in producing cost-effective, and regulation-compliant solutions.

CCS Concepts

• **Applied computing** → **Transportation**.

ICDCN 2025, January 04–07, 2025, Hyderabad, India

ACM ISBN 979-8-4007-1062-9/25/01
https://doi.org/10.1145/3700838.3700854

Keywords

Microtransit, Vehicular Networks, Pickup and Delivery Problem with Time Windows, Driver Breaks, MILP, Optimization

ACM Reference Format:
Agrima Khanna, Fangqi Liu, Samir Gupta, Sophie Pavia, Ayan Mukhopadhyay, and Abhishek Dubey. 2025. PDPTW-DB: MILP-Based Offline Route Planning for PDPTW with Driver Breaks. In *26th International Conference on Distributed Computing and Networking (ICDCN 2025), January 04–07, 2025, Hyderabad, India.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3700838.3700854

1 Introduction

Between 2010 and 2020, intercity buses were involved in 11% of all fatal bus crashes on average, while school buses and transit buses accounted for 38% and 35% of these crashes, respectively [9]. The U.S. Federal Motor Carrier Safety Administration (FMCSA) identified the key contributing factors — fatigue, time pressure, and distraction [11], highlighting the need for stricter rest regulations and effective route planning to ensure road safety. To address these concerns, the FMCSA established Hours of Service (HOS) regulations, requiring passenger-carrying drivers to take a 30-minute break after 8 hours of driving [10]. The European Union enforces similar rules, including a daily driving limit of 9 hours (extendable to 10 hours twice a week) and a weekly limit of 56 hours. Drivers must also observe a daily rest of at least 11 hours (reducible to 9 hours up to three times a week) and take a 45-minute break after 4.5 hours of driving [8]. Although these regulations aim to ensure drivers get adequate rest, without proper route planning, drivers may struggle to find suitable opportunities for breaks or appropriate rest stops. This challenge is particularly pronounced in poorly scheduled passenger services, leading to potential violations of maximum working hours. Consequently, drivers might have to deviate from their routes or stop unexpectedly, delaying passenger delivery. In some instances, drivers may end up taking breaks in locations without adequate rest facilities or food supplies, which can exacerbate fatigue, reduce alertness, and compromise the safety of passengers and drivers.

We present a problem formulation and offer a solution method for the optimization of Microtransit systems. We focus on the Pickup and Delivery Problem with Time Windows (PDPTW), which has been used to represent Microtransit in previous studies [29]. PDPTWs involve coordinating the pickup and delivery of goods

or passengers within specified time windows while optimizing routes to minimize costs and meet scheduling constraints. The proposed approach incorporates considerations for driver break rules to address the associated challenges and improve overall system efficiency and compliance. Existing work in PDPTWs includes Kim et al., who developed a rolling horizon framework for Offline PDPTWs by decomposing the problem into manageable sub-problems to ensure smooth transitions between time windows [16]. Sivagnanam et al. addressed paratransit routing with flexible pickup windows by combining offline VRP complexity with online booking requirements using a learning-based policy [25]. Sartori and Buriol proposed a hybrid metaheuristic approach integrating Adaptive Guided Ejection Search (AGES), Large Neighborhood Search (LNS), and Set Partitioning (SP) to improve solutions and generate new problem instances [23].

In the context of PDPTWs considering driver breaks, Tuin et al. introduced a method for scheduling breaks during waiting periods using a modified Dijkstra's algorithm to optimize travel times [28]. Kok et al. tackled traffic congestion and driving hours regulations with a solution method for the Vehicle Routing Problem with Time Windows (VRPTW), balancing travel distance and duty time [17]. However, existing methods generally focus on breaks at specific times, such as lunch breaks, without addressing the need for periodic breaks throughout the route or considering the spatial requirements for break locations, which may impact the overall effectiveness of driver rest and route planning.

Considering the limitations of existing works, this paper aims to address PDPTWs by incorporating more flexible continuous time intervals for driver breaks and selecting suitable break locations that do not impact future delivery requests. Specifically, the main contributions of this paper include:

- A novel MILP formulation integrating periodic driver breaks into the Pickup and Delivery Problem with Time Windows (PDPTW-DB), balancing serviceability, travel cost, and break scheduling, with specific constraints and linearization techniques (Section 3).
- Implementation of the formulation and performance comparison of multiple optimization tools, including Gurobi, Google OR Tools, and Hexaly (formerly Localsolver).
- Experiment setup and evaluation of solution quality and impact on working hours using real-world Microtransit delivery statistics from the Public Transit Agency (PTA) (Section 4 and Section 5.1) .

2 Related Work

The Pickup and Delivery Problem with Time Windows (PDPTW) incorporating driver breaks can be derived from the Vehicle Routing Problem (VRP), which has been extensively studied, with various methodologies proposed to address challenges posed by different constraints. This section summarizes significant contributions to the field, highlighting their limitations and relevance to our problem.

Several studies have explored VRP or PDPTW with time-dependent driver breaks within specific time windows, such as lunch breaks. Kok et al. employed a dynamic programming heuristic to minimize driver duty time, accounting for time-dependent travel and EC regulations, while optimizing schedules and incorporating capacity constraints [17]. Braekers et al. combined a branch-and-cut method with meta-heuristics to optimize Dial-a-ride service routes, considering multiple vehicle types and depots. They recognized the importance of driver breaks and time-dependent travel times, planning to integrate these in future versions [4].

Su and Chen introduced a hybrid ant colony system to minimize delivery time, taking into account cargo capacity, battery capacity, and human energy constraints. They found that incorporating driver breaks did not significantly increase overall delivery time [26]. Prescott-Gagnon et al. employed a large neighborhood search methodology to minimize the number of vehicles and travel distance in VRPTWs. They considered weekly rest periods based n EU regulations but didn't account for rest periods within the schedules. [21].

Similarly, Goel employed a large neighborhood search strategy to optimize routing schedules in accordance with EU driver regulations. They considered 8-hour shifts with one break per shift and an 11-hour rest period [12]. However, their approach does not account for longer shifts that may require multiple breaks. Furthermore, the paper focuses on the Vehicle Routing Problem with Time Windows (VRPTW), which differs from the problem at hand involving passenger pickup and drop-off. A significant constraint in our problem is that drivers cannot take breaks when passengers are on board, as it would cause inconvenience. Therefore, routes must be scheduled while taking this restriction into account. Kopfer et al. proposed a distributed decision-making framework that aims to minimize vehicle usage and travel time while adhering to EC social legislation. Their research also focused on incorporating mandated rest periods throughout the week, rather than daily route schedule breaks, into the optimization process. [18]. Zhao et al. combined Hybrid Genetic Search with dynamic programming to minimize total travel time, addressing time-dependent travel and multi-trip constraints [31]. Their approach focused on maximizing idle time to mitigate driver fatigue.

Various studies have focused on different combinations of constraints within VRP formulations with breaks. Benjamin and Beasley utilized metaheuristics, including Tabu Search, VNS, and VNTS, to optimize routing with constraints such as time windows, multiple disposal facilities, and driver rest periods. Each vehicle has a driver rest period (associated with a lunch break during the working day [2]. Karademir et al. and Bazirha formulated the VRPTW with driving time-based breaks [1, 14]. Coelho et al. combined MILP and local search heuristics to minimize travel distance while addressing the classical VRPTW with lunch breaks, using VB.net and CPLEX for implementation [7]. Buhrkal et al. proposed their own formulation of the CVRPTW with lunch breaks [5]. Table 1 provides an overview of the research conducted on incorporating driver breaks into VRPs.

Modeling the problem as a Mixed-Integer Linear Programming (MILP) helps address complex and intertwined objectives. Yadav and Mukherjee demonstrated this in their work on charging and routing optimization for electric vehicles (EVs). MILP allows for the simultaneous consideration of limited battery capacity, varying charging rates, and congestion at charging stations while optimizing travel

and charging schedules. This approach ensures an even distribution of delivery locations among EVs, minimizing total travel and charging time in a smart grid environment [30]. Schiffer et al. studied the impact of synchronizing driver breaks with recharging operations while scheduling the routes of electric vehicles. Their investigation highlighted how Hours of Service (HOS) regulations influence the operational efficiency of Electric Commercial Vehicles (ECVs) compared to Internal Combustion Engine Vehicles (ICEVs) in mid-haul transportation. By integrating driver breaks with necessary recharging stops, they found that ECVs can become more competitive. The synchronization of breaks and recharging can lead to cost savings of up to 20% compared to ICEVs. [24].

Our formulation considers vehicle capacity, pickups and deliveries, time windows, and driving time-based driver breaks in the daily schedule, while also making specific spatial considerations by incorporating suitable break nodes and locations. We present a novel variant of the VRP, ensuring periodic driver breaks are integrated into the generated schedule, thereby improving the practical applicability and effectiveness of routing algorithms in Microtransit systems.

3 Problem Formulation

We formulate our PDPTW-DB problem based on existing PDPTW formulations [19] proposed by Pavia et al., which extend the framework originally presented by Toth and Vigo [27]. This section overviews our PDPTW-DB and illustrates the extension of constraints for incorporating driver break scheduling into the route plan. For clarity, Table 2 summarizes all notations used in the formulation.

3.1 Problem Description

Pickup-Delivery Request: In the PDPTW-DB problem, vehicles aim to serve n pickup and delivery requests. The set N represents all pickup and delivery locations, denoted as nodes, with the pickup nodes being $P = \{1, \ldots, n\}$ and the drop-off nodes being $D = \{n+1, \ldots, 2n\}$, such that $N = P \cup D$. Each request i involves a pickup node i and its corresponding delivery node $n + i$. The number of passengers is denoted by d_i, and s_i indicates the serving time at node i, which accounts for loading passengers or goods. Each request is constrained by time windows for both pickup and delivery. For each node i, the time window $[a_i, b_i]$ specifies the period within which the vehicle must arrive at that node to perform the pickup or delivery.

Vehicle's Route: Vehicles are represented by the set K, with each vehicle $k \in K$ assigned to serve a subset of requests. Each vehicle starts its route at the origin depot $o(k)$ and ends at the destination depot $d(k)$. The travel time between nodes i and j in N is denoted by τ_{ijk}, and the cost between nodes i and j is denoted by c_{ijk}. In this paper, we consider the cost based on travel distance (in kilometers). Each vehicle has an initial capacity C_k, representing its maximum passenger count, which changes as it serves requests. The load change after a vehicle visits node i is indicated by ℓ_i. For a pickup node i, $\ell_i = d_i$, and for a delivery node $n + i$, $\ell_i = -d_i$.

Driver Break Rules: Our driver break constraint stipulates that drivers can work a maximum of θ hours before taking a mandatory break. The additional time and cost for vehicle k taking a break between nodes i and j are denoted by τ^{break}_{ijk} and c^{break}_{ijk}, respectively. This includes the travel time from node i to the nearest break location, the break duration, and the travel time from the break location to node j. During route planning, we must ensure drivers have access to designated break facilities with food and rest areas, enabling them to resume service and meet subsequent requests on time.

Route Planning and Break Scheduling: Our problem involves planning routes for all vehicles to serve pickup and delivery requests within their respective time windows while also determining driver break times. To achieve this, we define a binary decision variable x_{ijk}, where $i, j \in N$ and $k \in K$. The variable x_{ijk} is defined as:

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

Additionally, we define a binary variable β_{ijk} for scheduling driver breaks, which is given by:

$$\beta_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ takes a break at node } i \text{ before going to node } j \\ 0 & \text{otherwise} \end{cases}$$

3.2 Objectives

Our PDPTW-DB aims to optimize route planning and break scheduling by minimizing vehicle workload, including travel and break times, while maximizing serviceability of requests. We incorporate serviceability into the objective function rather than as a constraint, acknowledging that some requests may not be feasible to fulfill in real-world scenarios. This is captured by decision variables x_{ijk} and β_{ijk}, as outlined in the objective function shown in Eq. (1).

$$\min \sum_{k \in K} \sum_{i,j \in A} (c_{ijk} x_{ijk} + \beta_{ijk} c^{break}_{ijk}) + \alpha \times (n - \sum_{i \in P} r_i) \quad (1)$$

In the equation, $\sum_{i,j \in A} (c_{ijk} x_{ijk} + \beta_{ijk} c^{break}_{ijk})$ represents the total travel time and additional time for breaks for vehicle k. The variable r_i indicates whether request i has been served, as computed by:

$$r_i = \sum_{k \in K} \sum_{j \in N \cup \{d(k)\}} x_{ijk} \qquad \forall i \in P \quad (2)$$

From Eq. (2), we can deduce that $r_i = 1$ if a vehicle visits the pickup node for request i; otherwise, $r_i = 0$. We include a penalty term for r_i in the objective function, using a large penalty coefficient α to penalize unserved requests, thereby maximizing serviceability.

3.3 Constraints for Pickup-Delivery Services

Next, we list all constraints of the PDPTW-DB formulation to ensure that vehicles are routed correctly and requests are effectively serviced.

To ensure that vehicles start and end their routes at predefined depots and that every visited node is subsequently left, we define the following constraints:

$$\sum_{j \in P \cup \{d(k)\}} x_{o(k),jk} = 1, \qquad \forall k \in K \quad (1)$$

$$\sum_{i \in D_k \cup \{o(k)\}} x_{i,d(k),k} = 1, \qquad \forall k \in K \quad (2)$$

Table 1: VRP with Driver Breaks - Summary

Reference	Methodology	Objective Function	Constraints
Kok et al. [17]	Dynamic Programming Heuristic	Minimize Driver Duty Time	Time-Dependent Travel, EC Regulations
Su and Chen [26]	Hybrid Ant Colony System	Minimize Delivery Time	Cargo Capacity, Battery Capacity, Human Energy
Prescott-Gagnon et al. [21]	Large Neighborhood Search	Minimize Number of Vehicles and Travel Distance	VRPTW, Weekly Rest periods based on EU regulations
Goel [12]	Large Neighborhood Search	Optimize Routing Schedules	CVRP, EU Driver Regulations, Fixed Break Periods
Kopfer et al. [18]	Distributed Decision-Making Framework	Minimize Vehicle Number and Travel Time	VRPTW, EC Social Legislation
Coelho et al. [7]	MILP and Local Search Heuristics	Minimize Travel Distance	Capacity, Time Windows, Lunch Breaks
Braekers et al. [4]	Branch-and-Cut and Meta-heuristics	Minimize Travel Time	Vehicle Type, Multiple Depots
Zhao et al. [31]	HGS and Dynamic Programming	Minimize Total Travel Time	Time-Dependent Travel, Multi-Trip
Rochat and Taillard [22]	Probabilistic Techniques for Local Search	Optimize Vehicle Routing	Simple VRP with Lunch Breaks

Table 2: Notation table for PDPDTW

Symbol	Description
n	Number of requests
N	Set of all nodes
P	Set of pickup nodes
D	Set of drop-off nodes
K	Set of vehicles
t_{ik}	Start time of service at node i by vehicle k
τ_{ijk}	Travel time between nodes i and j by vehicle k
x_{ijk}	1 if vehicle k travels from node i to j, else 0
r_i	1 if pickup node i is serviced, else 0
c_{ijk}	Travel cost from node i to j by vehicle k
y_i	Load of vehicle k after node i has been serviced
ℓ_i	Passenger load associated with node i
C_k	Capacity of vehicle k
s_i	Service time at node i
a_i, b_i	Start and end time window for node i
τ_{ijk}^{break}	Additional travel time between nodes i and j if β_{ijk}=1
c_{ijk}^{break}	Additional travel cost between nodes i and j if β_{ijk}=1
t_{ik}^{break}	Total driving time at last break node
t_{ik}^{acc}	Driving time accumulated since last break

$$\sum_{i \in N \cup \{o(k)\}} x_{ijk} = \sum_{i \in N \cup \{d(k)\}} x_{jik} \qquad \forall k \in K,\ j \in N \quad (3)$$

$$\sum_{j \in N} x_{ijk} = \sum_{j \in N} x_{j,n+i,k} \qquad \forall k \in K, i \in P \quad (4)$$

$$\sum_{k \in K} \sum_{j \in N \cup \{d(k)\}} x_{i,j,k} \leq r_{i-n}, \qquad \forall i \in D \quad (5)$$

where Constraint (1) and (2) ensure that each vehicle starts and ends its route at its origin and final depot. Constraint (3) ensures that if a vehicle travels from node i to node j, it must later leave node j, except when j is the final depot. Constraint (4) ensures that if vehicle k visits the pickup node i, it will subsequently visit the corresponding delivery node $i + n$. To ensure that each request i is served by the same vehicle, we define Constraint (5) to guarantee each serviced pickup request has a corresponding serviced delivery request:

Additionally, we define constraints to ensure that a vehicle's load never exceeds its capacity during its route. We introduce the intermediate load variable $y_{ik} \in \mathbb{R}_{\geq 0}$, where $k \in K$ and $i \in N$, to represent the load of vehicle k after serving at node i.

$$y_{o(k),k} = 0, \qquad \forall k \in K \quad (6)$$

$$y_{d(k),k} = 0, \qquad \forall k \in K \quad (7)$$

$$(x_{ijk} = 1) \Rightarrow y_{ik} + \ell_j = y_{jk}, \qquad \forall k \in K, i, j \in N \quad (8)$$

$$\ell_i \leq y_{ik} \leq C_k, \qquad \forall k \in K, i \in P \quad (9)$$

$$0 \leq y_{n+i,k} \leq C_k - \ell_i, \qquad \forall k \in K, n + i \in D \quad (10)$$

We first use Constraints (6) and (7) to ensure vehicles start with a load of 0 at their origin depots and return with a load of 0 at their final depots. Additionally, Constraint (8) updates the vehicle load y_{ik} based on the load change value ℓ_j, which indicates the number of passengers boarding or alighting at node j when vehicle k travels from node i to node j. Finally, Constraints (9) and (10) ensure that the vehicle load never exceeds its capacity C_k and is properly adjusted at both pickup and delivery nodes.

Furthermore, to ensure all requests are served within their required time windows, we introduce an intermediate variable $t_{ik} \in \mathbb{R}_{\geq 0}$, where $i \in N$ and $k \in K$, representing the start time of service

for vehicle k at node i. The following constraints are proposed:

$$t_{o(k),k} = 0, \quad \forall k \in K \tag{11}$$

$$(x_{ijk} = 1) \Rightarrow t_{ik} + s_i + \tau_{ijk} + \beta_{ijk}\tau^{break}_{ijk} \leq t_{jk}, \forall i, j \in N, \forall k \in K \tag{12}$$

$$a_i \leq t_{ik} \leq b_i, \quad \forall k \in K, i \in N \tag{13}$$

$$t_{ik} + \tau_{i,n+i,k} + \beta_{ijk}\tau^{break}_{ijk} \leq t_{n+i,k}, \quad \forall k \in K, i \in P \tag{14}$$

where Constraint (11) ensures that the start time at each vehicle's origin depot is zero. Constraint (12) mandates that if a vehicle travels from node i to node j, the start time at node j must occur after the service time at node i, including the service time at node i, travel time between nodes, and any additional break time if taken. Constraint (13) enforces that the start time t_{ik} at any node must fall within its designated time window $[a_i, b_i]$. If a vehicle arrives at node j before the earliest time window boundary a_i, it must wait until a_i to begin service. Finally, Constraint (14) ensures that each vehicle visits the pickup node before the corresponding delivery node by requiring the service time at the pickup node to precede that at the delivery node.

3.4 Constraints for Driver Breaks

We introduce constraints to ensure drivers take breaks at appropriate intervals during their routes. Specifically, drivers must take a break every θ time interval without disrupting service fulfillment. To implement this, we define two intermediate variables: $t^{break}_{ik} \in \mathbb{R}\geq 0$, which tracks the last break time taken by vehicle k at node i, and t^{acc}_{ik}, which maintains the accumulated driving time at node i since the last break. Next, we define constraints to update these parameters as follows:

$$t^{acc}_{o(k),k} = 0, \quad \forall k \in K \tag{15}$$

$$t^{break}_{o(k),k} = 0, \quad \forall k \in K \tag{16}$$

$$(x_{ijk} = 1) \wedge (\beta_{ijk} = 0) \Rightarrow t^{acc}_{jk} = t^{acc}_{ik} + t_{jk} - t_{ik}, \forall k \in K, i, j \in N \tag{17}$$

$$(x_{ijk} = 1) \wedge (\beta_{ijk} = 1) \Rightarrow t^{acc}_{jk} = 0, \quad \forall k \in K, i, j \in N \tag{18}$$

$$(x_{ijk} = 1) \wedge (\beta_{ijk} = 1) \Rightarrow t^{break}_{jk} = t_{jk}, \quad \forall k \in K, i, j \in N \tag{19}$$

$$(x_{ijk} = 1) \wedge (\beta_{ijk} = 0) \Rightarrow t^{break}_{jk} = t^{break}_{ik}, \quad \forall k \in K, i, j \in N \tag{20}$$

We define Constraints (15) and (16) to initialize the accumulated working time since the last break, t^{acc}_{ik}, and the last break time, t^{break}_{ik}, as 0 at the vehicle's origin depot. If vehicle k passes nodes i and j consecutively, i.e., $x_{ijk} = 1$, Constraint (17) updates t^{acc}_{jk} by adding the accumulated working time at node i, (t^{acc}_{ik}), and the travel time and service time gap between nodes i and j, assuming no break is taken between these nodes. If a break occurs between nodes i and j, Constraint (18) resets t^{acc}_{jk} to zero. Constraint (19) ensures that if a break is taken between nodes i and j, the last break time at node j: t^{break}_{jk} is updated to the time service began at node j and The constraint in (20) ensures that if no break is taken between nodes i and j, the timestamp of the last break remains unchanged and is carried over from node i to node j.

We further establish constraints to ensure vehicle route planning compliance with driver break rules:

$$a_i \leq t^{break}_{ik} + t^{acc}_{ik} \leq b_i, \quad \forall k \in K, i \in N \tag{21}$$

$$(x_{ijk} = 1) \Rightarrow t^{break}_{ik} + t^{acc}_{ik} + s_i + \tau_{ijk} \leq t^{break}_{jk} + t^{acc}_{jk}, \quad \forall i, j \in N, \forall k \in K \tag{22}$$

$$(\beta_{ijk} = 1) \Rightarrow y_{i,k} = 0, \quad \forall k \in K, i, j \in N \tag{23}$$

$$(t^{acc}_{ik} \geq \theta) \Rightarrow \beta_{ijk} = 1, \quad \forall k \in K, i, j \in N \tag{24}$$

We use Constraint (21) to ensure that the accumulated working time at node i is within its service window. Constraint (22) verifies that the sequence of the last break time and the accumulated time adheres to the specified time order in the routes. Constraint (23) guarantees that the vehicle's load is zero when the driver takes a break. Lastly, Constraint (24) ensures a break is taken if the working threshold θ is reached.

3.5 Constraint Linearization

To handle logical implications and conditional relationships in our MILP formulation, we linearize Constraints (8), (12), (17) – (19), and (22) – (24) using the Big-M linearization method. We introduce M as a large constant and consider variables $i, j \in N$ and $k \in K$.

Constraint (8) for updating the vehicle load at nodes where the vehicle passes with $x_{ijk} = 1$ is linearized as follows:

$$(y_{ik} + \ell_j - y_{jk}) \leq M \times (1 - x_{ijk}), \tag{8a}$$

Constraint (12) that enforces the update of the starting service time is linearized as follows:

$$t_{ik} + s_i + \tau_{ijk} + \beta_{ijk}\tau^{break}_{ijk} - t_{jk} <= M \times (1 - x_{ijk}). \tag{12a}$$

Constraints (17) and (18), which ensure the update of t^{acc}_{jk} based on if a break is taken (β_{ijk}) can be linearized by Eqs.(17a) and (17b), and Eqs.(18a) and (18b) respectively.

$$t^{acc}_{jk} \geq t^{acc}_{i,k} + (t_{jk} - t_{ik}) - M \times \beta_{ijk} - M \times (1 - x_{ijk}), \tag{17a}$$

$$t^{acc}_{jk} \leq t^{acc}_{ik} + (t_{jk} - t_{ik}) + M \times \beta_{ijk} + M \times (1 - x_{ijk}), \tag{17b}$$

$$t^{acc}_{jk} \geq (t_{jk} - t_{ik}) - M \times (1 - \beta_{ijk}) - M \times (1 - x_{ijk}), \tag{18a}$$

$$t^{acc}_{jk} \leq (t_{jk} - t_{ik}) + M \times (1 - \beta_{ijk}) + M \times (1 - x_{ijk}). \tag{18b}$$

Constraints (19) and (20) that update the last break time based on whether break is taken at the last node are linearized by Eqs.(19a) and (19b), and Eqs.(20a) and (20b).

$$t^{break}_{jk} \geq t_{ik} - M \times (1 - \beta_{ijk}) - M \times (1 - x_{ijk}), \tag{19a}$$

$$t^{break}_{jk} \leq t_{ik} + M \times (1 - \beta_{ijk}) + M \times (1 - x_{ijk}), \tag{19b}$$

$$t^{break}_{jk} \geq t^{break}_{ik} - M \times \beta_{ijk} - M \times (1 - x_{ijk}), \tag{20a}$$

$$t^{break}_{jk} \leq t^{break}_{ik} + M \times \beta_{ijk} + M \times (1 - x_{ijk}). \tag{20b}$$

Constraint (22), which ensures the correct sequence of t^{acc}_{ik} and t^{break}_{ik} based on the order of visited nodes, is linearized as follows:

$$t^{break}_{ik} + t^{acc}_{ik} + s_i + \tau_{ijk} - (t^{break}_{jk} + t^{acc}_{jk}) \leq M \times (1 - x_{ijk}). \tag{22a}$$

Constraint (23) that ensures the vehicle load equals 0 during a break, is linearized as follows:

$$y_{i,k} \leq M \times (1 - \beta_{ijk}). \tag{23a}$$

Constraint (24) that enforces taking a break after θ, is linearized by

$$t_{ik}^{acc} + \tau_{ijk} \leq \theta - \epsilon + M \times \beta_{i,j,k} \tag{24a}$$

$$t_{ik}^{acc} + \tau_{ijk} \geq \theta - M \times \beta_{i,j,k}. \tag{24b}$$

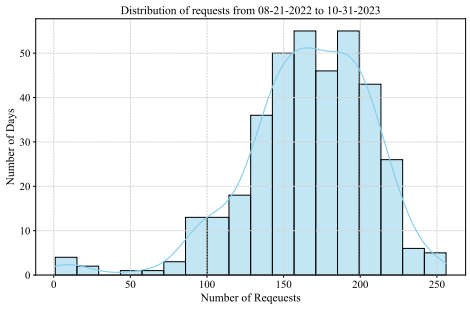


Figure 1: Distribution of Request Numbers

Data	
Number of Days	377
Number of Requests (Median)	168
Number of Bins	5
Size of Bin	50
Sample Size per Bin	10
Length of Time Window for Each Node	30 min
Number of Vehicles or Drivers	4
Driver Shifts	5:00 A.M. to 8:20 P.M.
Depot Time Window	[0,55200] (in seconds)
Break Rules	
European Union (EU)	θ = 4.5 hr, Break Duration = 45 min
FMCSA	θ = 8 hr, Break Duration = 30 min

Table 3: Summary of Data and Break Rules

4 Experiment Setup

The following section details the experimental setup, dataset, methodologies, and optimization tools used to address our proposed Pickup and Delivery Problem with Time Windows incorporating driver breaks (PDPTW-DB).

To conduct our analysis, we sourced user request data from our partner Public Transit Agency (PTA) (name hidden for blind review). This dataset included essential details such as latitude and longitude coordinates for pickup and dropoff locations, the number of passengers or customers per request, and precise pickup times (service times). To standardize the data, a dropoff time was set exactly one hour after the pickup for each request, and a 30-minute service window was established. Our study involved a fleet of four vehicles, with driver shifts scheduled from 5:00 AM to 8:20 PM, effectively determining the operating hours of the depot. To accurately assess distances and travel times between each pair of points, we leveraged data from the Open Source Routing Machine (OSRM), utilizing the provided latitude and longitude coordinates. The distance values obtained from OSRM were used to create the cost matrix, and the corresponding travel times were employed to generate the time matrix.

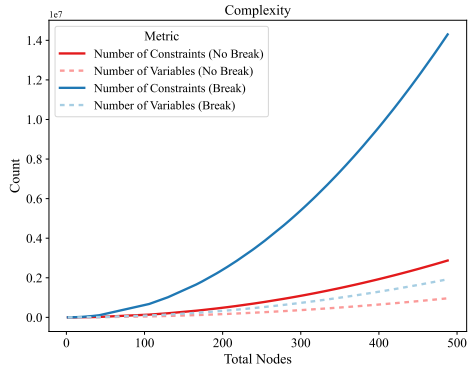


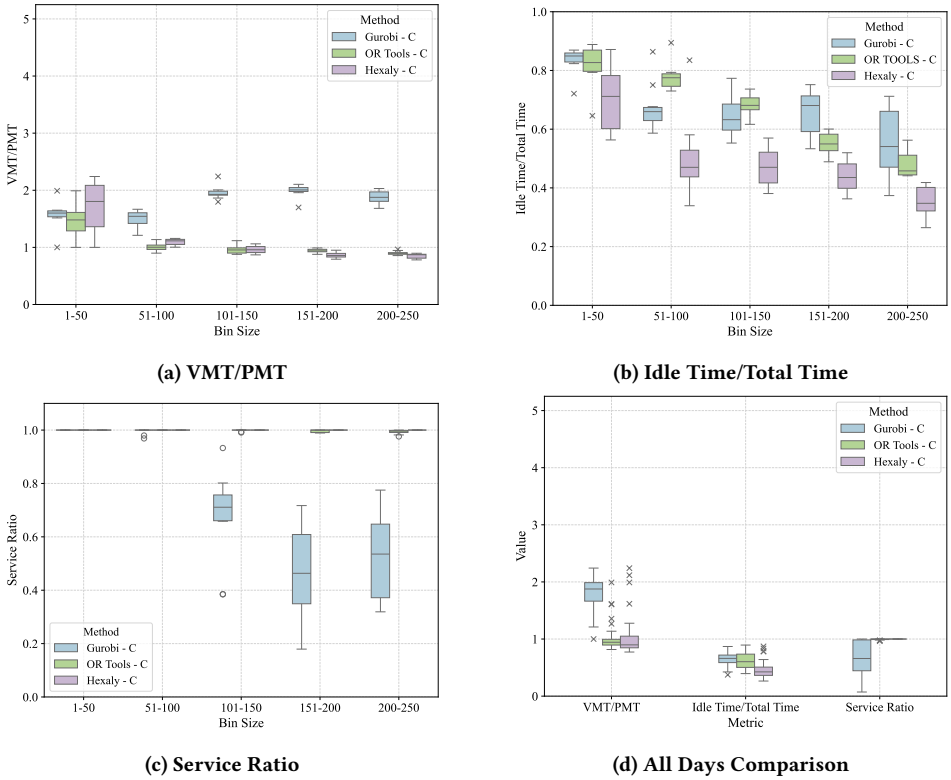
Figure 2: Impact of Driver Breaks on Problem Complexity

Daily Pickup-Dropoff Requests: The dataset spans 377 days from 2022-08-08 to 2023-10-31, with daily requests ranging from 1 to 250. The distribution of daily request numbers is shown in Figure (1). To evaluate solutions at different problem scales, samples were categorized into five bins based on daily request numbers: 1-50, 51-100, 101-150, 151-200, and 201-250. 10 days of data from each bin were sampled for experimentation. The dataset used in this study can be made available to researchers upon request, following the application of appropriate data anonymization procedures.

Break Rules: The break duration and the time after which a break must be taken were varied in our experiments. We conducted two sets of experiments to explore different regulatory frameworks. The first set adhered to the EU rule, which mandates a 45-minute break after 4.5 hours of driving [8]. The second set followed FMCSA regulations, requiring a 30-minute break after 8 hours of driving [10]. This allowed us to evaluate the impact of different break requirements on overall performance metrics. The break rules and data details have been summarized in Table 3.

Break Location Selection: Feasible break locations within the experiment region were identified using the Google Places API, based on specific location types, including but not limited to 'supermarkets', 'gas stations', and 'parks'. The closest break location that would minimize additional travel distance when a break was required were then determined using the OSRM server. For each pair of points, a 1.5 km radius was initially set to find potential break locations. If no suitable locations were found within this radius, the search radius was gradually expanded until a feasible break location was identified.

Solution Approaches: We employed off-the-shelf solvers to implement the base problem (PDPTW) and our formulation (PDPTW-DB) and evaluate performance across various metrics. The solvers used are as follows:



(a) VMT/PMT

(b) Idle Time/Total Time

(c) Service Ratio

(d) All Days Comparison

Figure 3: Comparison of Metrics between Gurobi PDPTW, Google OR-Tools PDPTW, and Hexaly PDPTW

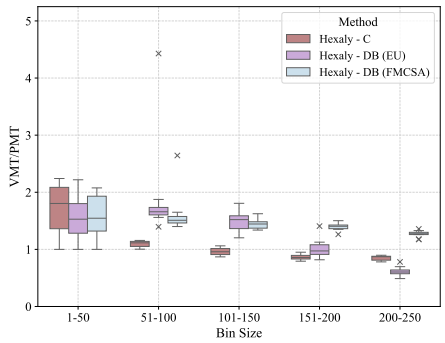
- **Gurobi**: A commercial solver for Mixed-Integer Linear Programming (MILP), known for its effectiveness in tackling large-scale optimization problems [13]. We configure its parameters with MIPFocus set to 1 to prioritize finding high-quality feasible solutions, Heuristics set to 0.5 to balance between exploration and exploitation, and CliqueCuts set to 2 to apply more aggressive cut generation for tighter formulations. The time limit on the search was set to 24 hours.
- **Google OR-Tools**: OR-Tools Routing, developed by Google, is a widely accessible Vehicle Routing Problem (VRP) solver. We implemented a VRP solver using the OR-Tools Routing API, employing the local cheapest insertion heuristic for the first solution and guided local search algorithm to optimize route plans. [20] The time limit on the search was set to 24 hours.
- **Hexaly**: A specialized solver designed for vehicle routing problems, particularly effective in managing complex constraints such as time windows and driver breaks and works with a combination of heuristic methods unique to the software. [3] We set the time limit on the search to 24 hours.

These approaches enabled us to assess the effectiveness and efficiency of different methodologies, particularly in managing the added complexity introduced by driver breaks.

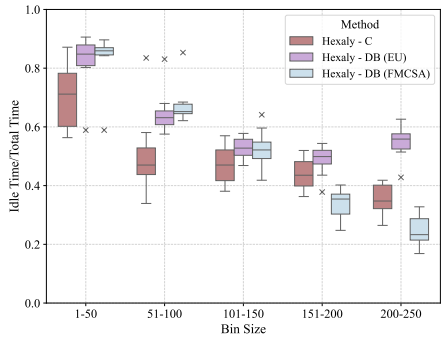
Metrics: The evaluation focused on key performance metrics:

- **Vehicle Miles Traveled per Passenger Mile Traveled (VMT/PMT)**: This metric assesses vehicle usage efficiency relative to passenger service. VMT represents the total vehicle miles traveled in a day, including miles with and without passengers on board. PMT is the sum of the shortest paths between each pickup node and delivery node pair in the dataset for a given day. Essentially, PMT indicates the vehicle miles required if each passenger were to travel directly between their origin and destination. A lower VMT/PMT ratio signifies more efficient vehicle use, minimizing unnecessary travel miles. [19]
- **Service Ratio**: The ratio of requests successfully serviced to the total number of requests. A higher service ratio indicates better fulfillment of user requests, which is crucial for the reliability of on-demand transit systems.
- **Idle Time**: The duration vehicles spend inactive, either waiting for requests or during breaks. By incorporating breaks, we aim to demonstrate that drivers experience a less demanding schedule. [6]

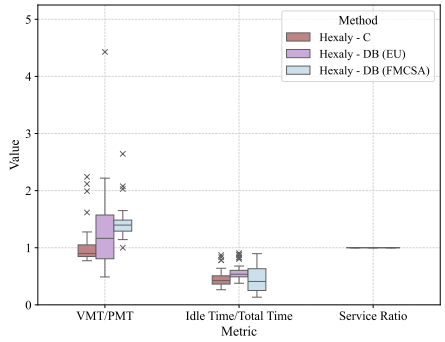
Comparing these metrics between the classical PDPTW and the version that includes driver breaks is essential for understanding the trade-offs involved in complying with regulatory requirements. While mandatory breaks may increase idle time and reduce service ratios, they are critical for ensuring driver safety and regulatory adherence. By quantifying the trade-offs between idle time and



(a) VMT/PMT



(b) Idle Time/Total Time



(c) All Days Comparison

Figure 4: Comparison of Metrics between Hexaly PDPTW, Hexaly PDPTW-DB (EU), Hexaly PDPTW-DB (FMCSA)

service ratio impacts, we can better understand the practical implications of different routing strategies and the importance of optimizing break schedules to mitigate negative effects on overall system performance.

5 Results

We compare our implementations of PDPTW and PDPTW-DB using various tools. Table 4 provides a comprehensive list of methods organized by the format "Tool - Problem Type (Driver Break Rules)." In this format, "$-C$" denotes implementations of the Classical PDPTW, while "$-D$" indicates implementations of PDPTW-DB.

Method	Implementation
Gurobi - C	Classical PDPTW using Gurobi
Gurobi - DB (EU)	PDPTW - DB using Gurobi ($\theta = 4.5$)
Gurobi - DB (FMCSA)	PDPTW - DB using Gurobi ($\theta = 8$)
OR Tools - C	Classical PDPTW using Google OR Tools
Hexaly - C	Classical PDPTW using Hexaly
Hexaly - DB (EU)	PDPTW - DB using Hexaly ($\theta = 4.5$)
Hexaly - DB (FMCSA)	PDPTW - DB using Hexaly ($\theta = 8$)

Table 4: Implementation Methods and Descriptions

5.1 Increased Complexity Due to Driver Breaks

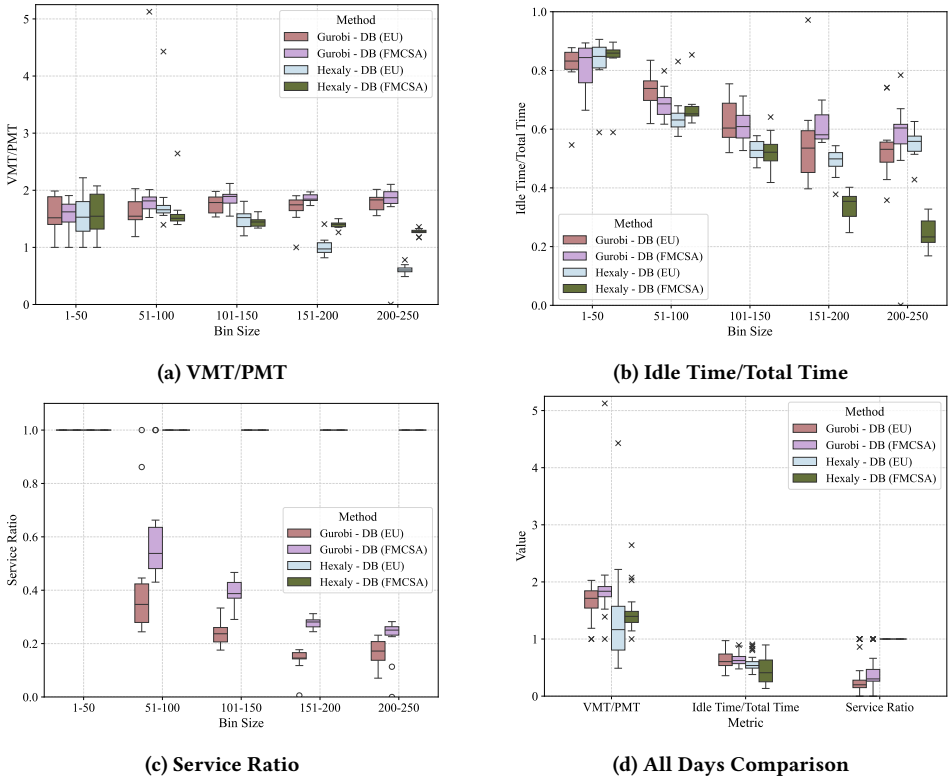
The classical Pickup and Delivery Problem with Time Windows (PDPTW) is known to be NP-hard, making it inherently challenging to solve efficiently for large instances. Introducing driver breaks further complicates the problem by increasing its complexity. Specifically, the number of variables expands to include scheduling these breaks, and the constraints multiply to ensure compliance with regulatory requirements and time windows for each driver's route. Figure (2) illustrates the increase in the number of variables and constraints for different problem sizes in both PDPTW and PDPTW-DB. It shows that incorporating breaks leads to a significant increase in complexity, with the number of variables and constraints approximately five times greater in PDPTW-DB when the node count reaches 500, compared to PDPTW without breaks.

5.2 Optimization Solver Comparison

Figure (3) compares the capabilities of three optimization solvers in solving the classic PDPTW. Figure (3a) highlights their performance on VMT/PMT, showing that Hexaly-C generally offers lower VMT/PMT, indicating better solutions with greater potential for ride-sharing within fixed time constraints. This advantage becomes more pronounced as the number of requests increases. For example, Hexaly-C achieves a VMT/PMT value that is 0.45 % of Gurobi-C's when the number of requests is between 200-250. Additionally, as the number of requests grows, both OR-Tools-C and Hexaly-C tend to provide solutions with lower VMT/PMT. This trend may be due to the increased likelihood of ride-sharing opportunities as more requests occur in close proximity. Similarly, Figure (3b) shows that Hexaly-C consistently results in a lower idle time ratio compared to the other two solvers, indicating that Hexaly-C is better at handling larger problems and providing superior solutions within a fixed time frame. Figure (3c) compares the service ratios of the three solvers, revealing that Gurobi-C struggles to maintain serviceability when the number of requests exceeds 100. Finally, Figure (3d) provides an overview of the performance of these three solvers across all metrics. It demonstrates that Hexaly-C generally offers lower VMT/PMT and idle time ratios, along with a higher service ratio, indicating higher quality solutions for PDPTW.

5.3 Performance comparison of Hexaly PDPTW and Hexaly PDPTW-DB

Figure (4) evaluates the solutions for classic PDPTW and PDPTW-DB using Hexaly, which provided the best performance among the three tools used. Hexaly-C represents the solution for standard



(a) VMT/PMT

(b) Idle Time/Total Time

(c) Service Ratio

(d) All Days Comparison

Figure 5: Comparison of Metrics between Hexaly - DB and Gurobi - DB

PDPTW, Hexaly-DB (EU) applies driver break rules set by the European Union, and Hexaly-DB (FMCSA) follows driver break rules set by the FMCSA. Figure (4a) shows the VMT/PMT ratio, which generally decreases as the number of requests increases. Solutions for PDPTW without driver break scheduling consistently offer the lowest VMT/PMT ratio compared to those with break scheduling. Among the methods with driver breaks, the FMCSA rule results in the highest VMT/PMT ratio, while the EU rule falls between the two. Figure (4b) indicates a decrease in idle time ratio as the number of requests increases for all methods. Hexaly-C exhibits the lowest idle time ratio across most request bins, while Hexaly-DB (EU) and Hexaly-DB (FMCSA) provide similar idle time ratios. Figure (4c) shows comparable service ratios for all three methods. The analysis suggests that the EU rule, with its shorter working duration, results in more flexible schedules, as indicated by the higher idle time. Although this may lead to slightly higher VMT/PMT values compared to other methods. Adhering to the EU rule ensures that all customers are served. This implies that the EU rule strikes a balance between driver break requirements and overall system efficiency while maintaining a high service level.

5.4 Driver Break Rules

Figure (5) explores the performance metrics of PDPTW-DB with different driver break rules: the FMCSA rule, which mandates an 8-hour driving time followed by a 30-minute break, and the EU rule, which requires a 4.5-hour driving time followed by a 45-minute break. The solutions from Gurobi and Hexaly are compared. Despite running the Gurobi solver for 24 hours, the results were suboptimal (see Figure (5c)) compared to Hexaly, which provides solutions for both driver break rules across scenarios with varying request numbers, achieving a service ratio of 1. Figure (5a) highlights that using the same solver, solutions under the EU rule (with a shorter work duration threshold) generally result in a lower VMT/PMT ratio compared to the FMCSA rule. This is due to the constraint that vehicles must have zero load during driver breaks, causing them to opt for routes with customers further apart to accommodate the 8-hour driving limit. This inference is supported by the lower idle time shown in Figure (5b) for the FMCSA rule, indicating that drivers spend more time travelling between service nodes. Therefore, it is more advantageous to follow the EU rule, which provides a break after 4.5 hours of driving. This approach ensures compliance with regulations and improves overall transportation system performance. Figure (5d) provides an overview, revealing that Hexaly outperforms Gurobi, delivering higher service ratios. Additionally, Hexaly with EU rules offers a lower VMT/PMT, while the FMCSA rule results in relatively lower idle time.

6 Conclusion

In this paper, we address a novel version of the Pickup and Delivery Problem with Time Windows (PDPTW) by incorporating driver

breaks into the vehicle routing process. We formulate the Pickup and Delivery Problem with Time Windows and Driver Breaks (PDPTW-DB) by extending the PDPTW model with additional constraints to account for periodic driver breaks and ensure that vehicles reach break locations with an empty load. We implement this formulation using multiple optimization tools, including Gurobi, Google OR-Tools, and Hexaly, to evaluate their performance across different problem scales and service request scenarios. We test these implementations using real-world data from a Microtransit system in a mid-sized southern U.S. city and compare the FMCSA rule (8-hour driving with a 30-minute break) with the EU rule (4.5-hour driving with a 45-minute break). Our experiments reveal that Hexaly outperforms both Gurobi and Google OR-Tools in solving the PDPTW-DB. The EU rule offers more flexible scheduling, resulting in higher idle times and although this may occasionally lead to slightly higher VMT/PMT values compared to other methods, the EU rule still ensures that all customers are served. We demonstrate the trade-off between service ratio, VMT/PMT, and idle time, showing that well-planned breaks can minimize the impact on VMT/PMT while maintaining efficiency in route planning. This insight also highlights the importance of considering multiple performance metrics when optimizing transportation systems. Our findings can serve as a reference for transportation authorities in resource allocation and route optimization for microtransit, paratransit, and public transit systems.

7 Acknoledgement

This material is based upon work sponsored by the National Science Foundation (NSF) under Award Number 1952011. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. Results presented in this paper were obtained using the Chameleon Testbed [15], supported by the NSF.

References

[1] Mohammed Bazirha. 2023. A novel MILP formulation and an efficient heuristic for the vehicle routing problem with lunch break. *Annals of Operations Research* (12 2023). https://doi.org/10.1007/s10479-023-05742-3

[2] A.M. Benjamin and J.E. Beasley. 2010. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers and Operations Research* 37, 12 (2010), 2270–2280. https://doi.org/10.1016/j.cor.2010.03.019

[3] Thierry Benoist, Bertrand Estellon, Frédéric Gardi, Romain Megel, and Karim Nouioua. 2011. LocalSolver 1.x: A black-box local-search solver for 0-1 programming. *4OR* 9 (09 2011), 299–316. https://doi.org/10.1007/s10288-011-0165-9

[4] Kris Braekers, An Caris, and Gerrit Janssens. 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological* 67 (09 2014), 166–186. https://doi.org/10.1016/j.trb.2014.05.007

[5] Katja Buhrkal, Allan Larsen, and Stefan Ropke. 2012. The Waste Collection Vehicle Routing Problem with Time Windows in a City Logistics Context. *Procedia - Social and Behavioral Sciences* 39 (12 2012), 241–254. https://doi.org/10.1016/j.sbspro.2012.03.105

[6] Ömer Çam and Hayrettin Sezen. 2020. Linear Programming Formulation For Vehicle Routing Problem Which Is Minimized Idle Time. *Decision Making: Applications in Management and Engineering* 3 (03 2020). https://doi.org/10.31181/dmame2003132h

[7] Leandro Coelho, Jean-Philippe Gagliardi, Jacques Renaud, and Angel Ruiz. 2015. Solving the vehicle routing problem with lunch break arising in the furniture delivery industry. *Journal of the Operational Research Society* 67 (2015). https://doi.org/10.1057/jors.2015.90

[8] European Commission. 2015. Driving time and rest periods. https://transport.ec.europa.eu/transport-modes/road/social-provisions/driving-time-and-rest-periods_en.

[9] Federal Motor Carrier Safety Administration. 2020. Large Truck and Bus Crash Facts 2020. https://www.fmcsa.dot.gov/safety/data-and-statistics/large-truck-and-bus-crash-facts-2020

[10] Federal Motor Carrier Safety Administration. 2022. Hours of Service (HOS). https://www.fmcsa.dot.gov/regulations/hours-of-service.

[11] Federal Motor Carrier Safety Administration (FMCSA). 2024. Driver Fatigue and Distraction Monitoring and Warning System, Phase I. https://www.fmcsa.dot.gov/safety/research-and-analysis/driver-fatigue-and-distraction-monitoring-and-warning-system.

[12] Asvin Goel. 2009. Vehicle Scheduling and Routing with Drivers' Working Hours. *Transportation Science* 43, 1 (February 2009), 17–26. https://doi.org/10.1287/trsc.1070.0226

[13] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[14] C. Karademir, B. Alves Beirigo, R.R. Negenborn, and B. Atasoy. 2022. Two-echelon Multi-trip Vehicle Routing Problem with Synchronization for An Integrated Water- and Land-based Transportation System. In *10th Symposium of the European Association for Research in Transportation.*

[15] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20).* USENIX Association.

[16] Youngseo Kim, Danushka Edirimanna, Michael Wilbur, Philip Pugliese, Aron Laszka, Abhishek Dubey, and Samitha Samaranayake. 2023. Rolling Horizon based Temporal Decomposition for the Offline Pickup and Delivery Problem with Time Windows. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023).*

[17] A. L. Kok, E. W. Hans, J. M. J. Schutten, and W. H. M. Zijm. 2011. A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks. *Flexible Services and Manufacturing Journal* 22, 1 (2011), 83–108. https://doi.org/10.1007/s10696-011-9077-4

[18] Herbert Kopfer, Christoph Meyer, , A. Kok, and Marco Schutten. 2011. *Distributed Decision Making in Combined Vehicle Routing and Break Scheduling.* 125–133. https://doi.org/10.1007/978-3-642-11996-5_12

[19] Sophie Pavia, David Rogers, Amutheezan Sivagnanam, Michael Wilbur, Danushka Edirimanna, Youngseo Kim, Philip Pugliese, Samitha Samaranayake, Aron Laszka, Ayan Mukhopadhyay, and Abhishek Dubey. 2024. Deploying Mobility-On-Demand for All by Optimizing Paratransit Services. In *International Joint Conferences on Artificial Intelligence 2024 (IJCAI '24).* 7430–7437.

[20] Laurent Perron and Vincent Furnon. 2024. *OR-Tools.* Google. https://developers.google.com/optimization/

[21] Eric Prescott-Gagnon, Guy Desaulniers, and Louis-Martin Rousseau. 2009. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks* 54 (12 2009), 190–204. https://doi.org/10.1002/net.20332

[22] Yves Rochat and Eric Taillard. 2015. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics* 1, 1 (September 2015), 147–167. https://doi.org/10.1007/BF02430370

[23] Carlo S. Sartori and Luciana S. Buriol. 2020. A study on the pickup and delivery problem with time windows: Matheuristics and new instances. *Computers and Operations Research* 124 (2020), 105065. https://doi.org/10.1016/j.cor.2020.105065

[24] Maximilian Schiffer, Gilbert Laporte, Michael Schneider, and Grit Walther. 2017. *The impact of synchronizing driver breaks and recharging operations for electric vehicles.* Technical Report G-2017-46. Les Cahiers du GERAD.

[25] Amutheezan Sivagnanam, Salah Kadir, Ayan Mukhopadhyay, Philip Pugliese, Abhishek Dubey, Samitha Samaranayake, and Aron Laszka. 2022. Offline Vehicle Routing Problem with Online Bookings: A Novel Problem Formulation with Applications to Paratransit. *Preprint submitted to Elsevier* (2022). Accepted for publication in the proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022).

[26] Zifei Su and Pingen Chen. 2024. Driver-Centric Capacitated Electric Vehicle Routing Problem: A Human Energy-Aware Approach. *SSRN* (2024).

[27] Paolo Toth and Daniele Vigo. 2002. *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515

[28] Marieke Tuin, Mathijs de Weerdt, and G. Veit Batz. 2018. Route Planning with Breaks and Truck Driving Bans Using Time-Dependent Contraction Hierarchies. In *Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018).* Association for the Advancement of Artificial Intelligence.

[29] Michael Wilbur, Maxime Coursey, Pravesh Koirala, Zakariyya Al-Quran, Philip Pugliese, and Abhishek Dubey. 2023. Mobility-On-Demand Transportation: A System for Microtransit and Paratransit Operations. In *ICCPS '23* (San Antonio, TX, USA) *(ICCPS '23).* Association for Computing Machinery, New York, NY, USA, 260–261. https://doi.org/10.1145/3576841.3589625

[30] Anoop Kumar Yadav and Joy Chandra Mukherjee. 2021. MILP-Based Charging and Route Selection of Electric Vehicles in Smart Grid. In *International Conference on Distributed Computing and Networking 2021 (ICDCN '21).* ACM. https://doi.org/10.1145/3427796.3427820

[31] Jingyi Zhao, Mark Poon, Vincent Tan, and Zhenzhen Zhang. 2024. A hybrid genetic search and dynamic programming-based split algorithm for the multi-trip time-dependent vehicle routing problem. *European Journal of Operational Research* 317 (04 2024). https://doi.org/10.1016/j.ejor.2024.04.011