Multi-Pickup and Multi-Delivery of Restaurant Orders: A Capacity-Constrained K-means Clustering Approach

Elson Cibaku, Sanchoy Das, SangWoo Park

***Abstract*—The Multi-Pickup and Multi-Delivery Problem with Time Window (MPMDPTW) is an essential optimization problem in transportation and logistics, with significant real-world applications in today's fast-paced environment. This research addresses the challenging task of optimizing multiple restaurant order pickups and deliveries through a novel application of a capacity-constrained K-means clustering approach. By employing advanced clustering techniques, this study aims to address the dynamic and complex logistics within the food service industry. The primary goal is to enhance service delivery efficiency, mitigate time lags, and reduce operational expenses. Our approach organizes orders into efficient clusters, effectively managing the distribution of tasks among available delivery resources. This method not only optimizes route planning but also adapts to varying logistical demands, ensuring more responsive and cost-effective delivery operations. This study redefines the utilization of clustering algorithms in managing food service logistics, marking a significant advancement in intelligent and automated delivery systems. We demonstrate the practical benefits and improvements in transportation and logistics systems through empirical analysis and real-world data application.**

***Index Terms*—Multi-pickup and multi-delivery problem with time window, vehicle routing problem, logistics optimization, time window constraints, capacity-constrained clustering, k-means, minimum cost flow, pickup and delivery routing, parallel optimization.**

I. Introduction

As the world becomes increasingly interconnected, the efficient transport of products and services has become a cornerstone of modern society, driving the need to address complex logistics problems such as the Multi-Pickup and Multi-Delivery Problem with Time Window (MPMDPTW). The MPMDPTW is an extension of the well-known Vehicle Routing Problem (VRP), a classic combinatorial optimization problem that aims to find the optimal routes for a fleet of vehicles to serve a set of customers with known demands while minimizing the total distance traveled [1]–[6]. The MPMDPTW introduces additional complexities by incorporating time window constraints for pickups and deliveries, as well as vehicle capacity limitations.

Efficiently solving the MPMDPTW can result in significant cost savings, improved customer satisfaction, and reduced environmental impact. Its applications span a wide range of industries, including e-commerce [7]–[9], healthcare [10], [11], urban mobility and transportation planning [12]–[14]. For instance, in e-commerce, the MPMDPTW arises in managing last-mile delivery, and in healthcare, the problem arises in managing medical supply channels, where the timely distribution of medical equipment, drugs, and other supplies is essential for patient care.

Elson Cibaku, Sanchoy Das, and SangWoo Park are in the Department of Mechanical and Industrial Engineering at New Jersey Institute of Technology, Newark, NJ, USA. Emails: ec426@njit.edu, sanchoy.das@njit.edu, sangwoo.park@njit.edu

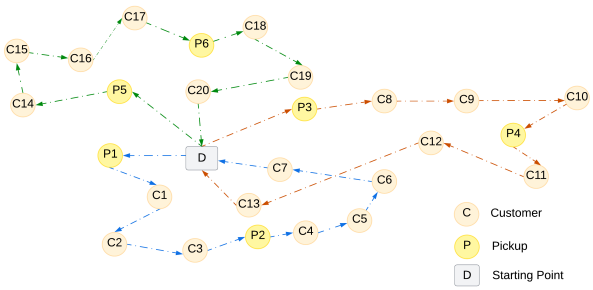


Fig. 1: Illustration of a pickup and delivery route showcasing drivers' routes, starting point (D), pickup points (P), and customers (C). Arrows indicate the flow of goods from pickup locations to customers, with different colors representing distinct routes and connections within the route.

The MPMDPTW is particularly salient in the rapidly expanding sector of on-demand food delivery, exemplified by platforms such as Uber Eats and DoorDash. These services must continuously coordinate a fleet of drivers to collect meals from multiple restaurants and deliver them to multiple customers within strictly defined time windows. Figure 1 illustrates the optimal solution of an MPMDPTW instance. Effective MPMDPTW solutions can reduce travel distances, driver workload, and operational costs while ensuring timely deliveries, thereby enhancing customer satisfaction and loyalty.

However, MPMDPTW inherits the NP-hardness of the VRP, making it difficult to find an optimal solution within a reasonable time, especially for large-scale problems [15]–[17]. Note that the MPMDPTW also introduces added complexity arising from the time window constraints, the many-to-many pickup-to-delivery mapping, and the vehicle capacity constraints. In fact, the MPMDPTW's trade-offs are complex and multifaceted. For instance, a trade-off exists between minimizing total distance or travel time and maximizing vehicle utilization. In addition, there is a trade-off between minimizing the required number of vehicles and ensuring that each delivery is made within the allotted time frame. It can be challenging to strike an equilibrium between these trade-offs, as achieving one objective may necessitate sacrificing another.

In this paper, we propose an accurate and computationally efficient approach for solving the MPMDPTW problem, grounded in a novel capacity-constrained clustering technique. Our contributions can be summarized as follows:

- To the best of our knowledge, we present, for the first time, a comprehensive mathematical model for the MP-MDPTW that accommodates multiple depots, enforces pickup–delivery precedence and vehicle capacities, and integrates earliness/lateness penalties, thereby providing a more realistic and operationally relevant representation of the problem.
- We develop a new capacity-constrained clustering algorithm that is efficient and essentially boils down to solving a linear programming problem formulated as an equivalent minimum cost flow problem, ensuring scalability and computational tractability.
- By combining clustering-based decomposition with parallelization, we utilize existing commercial solvers to enhance both the speed and accuracy of the overall solution process.
- We generate new datasets exhibiting varied spatial and temporal characteristics and perform extensive simulations, demonstrating the efficiency and adaptability of our approach under diverse conditions.

The remainder of this paper is organized as follows. Section II presents a literature review. Section III details the mathematical model formulation of MPMDPTW, followed by Section IV, which elaborates on the implementation of the proposed solution algorithm. Section V provides numerical results and analysis of the solution procedure in different scenarios, including different input parameter values. Finally, Section VI offers concluding remarks.

II. Literature Review

Pickup and Delivery Problems (PDP), Pickup and Delivery Problems with Time Windows (PDPTW), and Multi-Pickup and Delivery Traveling Salesman Problems with Time Windows (MPDPTW) represent variations of the Vehicle Routing Problem (VRP). These complex challenges have garnered considerable interest in operations research and transportation science. This literature review endeavors to succinctly expound upon the progression of solution methodologies for PDP and its variants.

Early research on the PDP and its time-window variant (PDPTW) focused on exact methods like branch-and-bound, branch-and-cut, and column-generation algorithms. The authors in [1] introduced the PDPTW with a branch-and-cut algorithm. Building on this, authors in [3] enhanced computational efficiency with a branch-and-price approach. [18] further improved branch-and-cut methods for PDPTW, incorporating valid inequalities. The authors in [19] applied column-generation techniques for the MPDPTW. The authors in [20] proposed an adaptive large neighborhood search (ALNS) for PDP, incorporating branch-and-price algorithms. Stochastic methods emerged to address uncertainties like varying travel times and demands, with contributions by [21] and [22]. Despite the accuracy of exact approaches, their scalability limitations led to the exploration of alternative methodologies.

To overcome the limitations of exact methods, researchers developed heuristics and metaheuristics for solving PDP and its variants. [23] proposed a tabu search heuristic for PDPTW, employing strategic oscillation and path relinking strategies to solve large-scale problems. Genetic algorithms were introduced by [24] for PDPTW, utilizing genetic operators like crossover, mutation, and selection. This approach demonstrated the potential of genetic algorithms in complex routing with time constraints. The authors in [25] showcased the efficiency of simulated annealing for large-scale instances of the PDPTW, integrating local search and acceptance criteria.

Meanwhile, multi-objective optimization addresses uncertainties and conflicting objectives in PDP and their variants. Multi-objective optimization aims to balance objectives such as minimizing operational costs, reducing travel times, and improving service quality. [26] proposed a Pareto-based multi-objective local search for the bi-objective PDPTW, minimizing travel distance and balancing vehicle workloads. In [27], the authors proposed a multi-objective genetic algorithm for PDP with Time Windows, considering cost and time objectives through a Pareto-based ranking mechanism. These approaches demonstrate the efficacy of multi-objective optimization in complex routing scenarios.

The rising interest in artificial intelligence and machine learning has led researchers to explore their applicability in this field. Neural networks help learn representations of problem instances, generating initial solutions refined by local search or metaheuristics. Reinforcement learning, including Q-learning and actor-critic methods, has been used for routing policies. [28] introduced the Pointer Network, leveraging attention mechanisms for combinatorial optimization, including PDP. Notably, the authors in [29] proposed a sequence-to-sequence model using recurrent neural networks (RNNs) to predict node sequences for feasible routes. These approaches demonstrate the potential of deep learning in addressing complex routing problems. Reinforcement learning (RL) approaches [30], [31] have also been applied to PDPTW and MPMDPTW, offering promising avenues for solving these problems by learning decision-making policies. In [32], the authors proposed an RL-based approach for the Capacitated Vehicle Routing Problem (CVRP) using a deep Q-network to learn routing policies. The authors in [33] introduced a deep reinforcement learning (DRL) approach for PDP, combining a graph neural network (GNN) with an RL algorithm. This combination allowed the model to learn both the representation of problem instances and optimal decision policies, demonstrating the potential of integrating deep learning and RL for complex routing challenges.

Due to their ability to naturally decompose large problems into smaller instances, which can be solved much faster in parallel, clustering techniques like K-means have been widely applied to PDPs and VRPs. The authors in [34], [35] demonstrated the simplicity and efficiency of K-means in partitioning data based on proximity. In [36], the authors used K-means for customer assignment in last-mile delivery, reducing travel distance. The authors in [37] employed K-means to partition delivery areas, balancing workload and optimizing costs. In cold chain logistics, K-means helped reduce costs and wastage [38]. Hybrid methods combining K-means and genetic algo-

rithms addressed green delivery and multi-depot PDP [39], [40]. Clustering also optimized distribution center locations [41], vehicle routing with time windows [42], and two-echelon routing with multi-commodity goods [43]. Existing clustering approaches, while successful in reducing computational complexity by grouping spatially proximate nodes, are generally designed for conventional VRPs and often overlook the intricate operational constraints inherent in MPMDPTW settings. For instance, standard K-means or related methods do not naturally account for vehicle capacity limitations, order-specific time windows, or the required precedence relationships between pickups and deliveries. As a result, directly applying these traditional clustering strategies to MPMDPTW can lead to infeasible solutions or require extensive post-processing to enforce operational constraints. In contrast, our approach integrates capacity directly into the clustering phase via a minimum cost flow formulation, ensuring that each cluster remains feasible under vehicle load limitations. This capacity-aware grouping significantly reduces the post-processing required to handle time windows and pickup-delivery precedence in subsequent routing steps, yielding a more efficient method for solving MPMDPTW instances.

III. Problem Formulation

In this section, we present the mathematical problem formulations for the MPMDPTW. We define the studied MPMDPTW over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes in $\mathcal{V}$ represent locations, and edges in $\mathcal{E}$ represent trips between locations. The set $\mathcal{V}$ is the union of three mutually exclusive sets: $\mathcal{P}$, which is the set of pickup locations, $\mathcal{D}$ the set of delivery locations, and $\mathcal{S}$ the set of depot locations.

A. Notation

We introduce the notations used in the mathematical model of the MPMDPTW:

Sets and Parameters

- $\mathcal{V} = \{1, \ldots, n\}$: Set of nodes representing locations.
- $\mathcal{E} = \{(i,j) | i, j \in V, i \neq j\}$: Set of edges between nodes.
- $\mathcal{K} = \{1, \ldots, m\}$: Set of dispatchable vehicles.
- $\mathcal{S} \subseteq \mathcal{V}$: Set of depot locations, $\mathcal{P} \subseteq \mathcal{V}$: Set of pickup locations, $\mathcal{D} \subseteq \mathcal{V}$: Set of delivery locations.
- $\mathcal{R} = \{(p_o, d_o) | o = 1, \ldots, n_r\}$: Set of customer requests, where p_o and d_o are the pickup and delivery nodes of order o, respectively.
- Q_k: Capacity of vehicle k.
- T_{ijk}: Travel time between nodes i and j for vehicle k.
- S_i: Service time at node i.
- q_i: Demand of node i.
- $TW_i = [e_i, l_i]$: Time window of node i, where e_i and l_i are the earliest and latest service times at node i, respectively.
- $S(k) : (\mathcal{K} \to \mathcal{S})$ a function that maps each vehicle $k \in \mathcal{K}$ to its designated starting depot in $\mathcal{S}$.
- $\mathcal{N}(i)$ is the set of nodes that are adjacent to node i in the graph. In other words, this is the set of nodes that can be traveled from node i.

Decision Variables

- $x_{ijk} \in \{0, 1\}$: Binary variable that takes the value 1 if vehicle k travels on arc $(i,j) \in \mathcal{E}$, and 0 otherwise.
- $b_{ik} \in \mathbb{R}^+$: Continuous variable representing the time when vehicle k starts servicing node i.
- $q_{ik} \in \mathbb{Z}^+$: Integer variable representing the load of vehicle k when leaving node i.
- $t_{ik} \in \mathbb{R}^+$: Continuous variable representing the tardiness at node i for vehicle k.
- $s_{ik} \in \{0, 1\}$: Binary variable that takes the value 1 if node i is serviced by vehicle k within the time window, and 0 otherwise.
- $r_{ik} \in \mathbb{R}^+$: Continuous variable representing the earliness at node i for vehicle k.

B. Objective Function

The objective is to minimize the total operational cost for all vehicles, which includes the sum of the travel times expended by each vehicle on each arc, along with penalties for earliness and tardiness at pickup and delivery locations. The mathematical formulation of the objective function is as follows:

$$\min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{E}} T_{ijk} \cdot x_{ijk} + \rho \sum_{i \in \mathcal{P} \cup \mathcal{D}} \sum_{k \in \mathcal{K}} (t_{ik} + r_{ik}) \quad (1)$$

C. Constraints

Every pickup and delivery must be served exactly once. This ensures that each customer's request is satisfied by a single vehicle.

$$\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{V}} x_{ijk} = 1, \quad \forall i \in \mathcal{P} \cup \mathcal{D} \quad (2)$$

If dispatched, vehicle k starts at the designated depot, $S(k)$, and must directly go to a pickup location (i.e., cannot go directly to a delivery location). A vehicle may not leave at a non-designated depot.

$$\sum_{j \in \mathcal{P}} x_{S(k)jk} \leq 1, \quad \forall k \in \mathcal{K} \quad (3)$$

$$\sum_{j \in \mathcal{D}} x_{S(k)jk} = 0, \quad \forall k \in \mathcal{K} \quad (4)$$

$$\sum_{i \in \mathcal{S}/S(k)} \sum_{j \in \mathcal{V}} x_{ijk} = 0, \quad \forall k \in \mathcal{K} \quad (5)$$

Similarly, if dispatched, a vehicle ends its route at the designated depot, $S(k)$ after serving its last delivery (i.e., cannot come directly from a pickup location). A vehicle may enter a non-designated depot.

$$\sum_{i \in \mathcal{D}} x_{iS(k)k} \leq 1, \quad \forall k \in \mathcal{K} \quad (6)$$

$$\sum_{i \in \mathcal{P}} x_{iS(k)k} = 0, \quad \forall k \in \mathcal{K} \quad (7)$$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{S}/S(k)} x_{ijk} = 0, \quad \forall k \in \mathcal{K} \quad (8)$$

Next, the flow conservation constraints ensure that if a vehicle k enters a node i, it should also leave it:

$$\sum_{j \in \mathcal{V}} x_{ijk} = \sum_{j \in \mathcal{V}} x_{jik}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K} \quad (9)$$

The following condition uses the big-M method to ensure that if a vehicle k goes from node i to node j (i.e., $x_{ijk} = 1$), the starting service time at node j for vehicle k must be greater than or equal to the time vehicle k finishes service at node i and travels to node j. This effectively ensures the proper sequencing of nodes and eliminates the possibility of subtours.

$$b_{jk} + (1 - x_{ijk})M \geq b_{ik} + S_i + T_{ijk}, \ \forall (i,j,k) \in \mathcal{E} \times \mathcal{K} \tag{10}$$

The parameter M is a large constant that renders the inequality irrelevant if $x_{ijk} = 0$. To initialize the reference time, we also need a constraint ensuring that each vehicle k starts its service at its designated depot $S(k)$ at time 0:

$$b_{S(k)k} = 0, \quad \forall k \in \mathcal{K}. \tag{11}$$

Another important aspect of the problem is the load constraint, which guarantees that a vehicle's load respects the pickup and delivery actions and never exceeds its capacity throughout its tour.

$$q_{jk} + (1 - x_{ijk})M \geq q_{ik} + q_j, \ \forall (i,j,k) \in \mathcal{E} \times \mathcal{K} \tag{12}$$

$$q_{ik} \leq Q_k, \ \forall (i,k) \in \mathcal{V} \times \mathcal{K} \tag{13}$$

$$q_{S(k)k} = 0, \ \forall k \in \mathcal{K} \tag{14}$$

$$q_{ik} \geq 0, \ \forall (i,k) \in \mathcal{V} \times \mathcal{K} \tag{15}$$

The first constraint is the load balancing constraint, which ensures that if vehicle k goes from node i to node j (i.e., $x_{ijk} = 1$), the load of vehicle k when leaving node j is at least the load of vehicle k when leaving node i plus the demand at node j. Note that for a pickup location, the demand takes on a positive value, while for a delivery location, the demand takes on a negative value. If the vehicle does not go from node i to node j (i.e., $x_{ijk} = 0$), then the constraint is not binding due to the big-M method. In our implementation, we set the value of M to be equal to maximum vehicle capacity. The second equation enforces the vehicle capacity limit, the third enforces non-negativity, and the fourth ensures that the load of any vehicle k starting at its depot $S(k)$ is zero, as the vehicle has not picked up any orders yet.

The following time window constraints ensure that each delivery operation respects the given time windows, avoiding early arrivals or late services that could cause discomfort for the customers:

$$\begin{aligned} e_i &\leq b_{ik}, \quad \forall i \in \mathcal{D}, \forall k \in \mathcal{K} \\ b_{ik} + S_i &\leq l_i, \quad \forall i \in \mathcal{D}, \forall k \in \mathcal{K} \end{aligned} \tag{16}$$

While the above constraints require strict compliance with the given time windows, this is oftentimes not feasible. Therefore, we relax the constraints and penalize the violations by introducing the tardiness and earliness terms. Tardiness is represented by the difference between the actual service completion time and the upper end of the time window:

$$t_{ik} \geq b_{ik} + S_i - l_i, \quad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall k \in \mathcal{K} \tag{17}$$

If the service is completed within the time window, the tardiness is zero. Similarly, earliness is represented by the difference between the service start time and the lower end of the time window:

$$r_{ik} \geq e_i - b_{ik}, \quad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall k \in \mathcal{K} \tag{18}$$

If the service is started within the time window, the earliness is zero.

To guarantee that both the pickup and delivery of a single request are completed by the same vehicle, we include the following constraints in the model:

$$\sum_{j \in \mathcal{V}} x_{p_o jk} \leq \sum_{j \in \mathcal{V}} x_{d_o jk}, \quad \forall (p_o, d_o) \in \mathcal{R}, \forall k \in \mathcal{K} \tag{19}$$

The following constraints ensure that for each request *i*, the delivery task must happen after the corresponding pickup task has happened for each vehicle *k*.

$$b_{p_o k} \leq b_{d_o k}, \quad \forall ((p_o, d_o), k) \in \mathcal{R} \times \mathcal{K} \tag{20}$$

This constraint complements constraint (10), ensuring that the pickup precedes the delivery in its route.

IV. Solution Methodology

The exact problem formulation described in Section III is computationally challenging to solve, especially for large systems. In this section, we provide a detailed overview of our proposed algorithm designed to solve the MPMDPTW. The algorithm is structured into two primary steps: (i) clustering of orders and assigning clusters to vehicles and (ii) finding the optimal route for each vehicle (which may cover multiple clusters). As mentioned previously, existing clustering techniques cannot be used in our problem context because of the vehicle capacity constraints. The proposed clustering-based decomposition also enables a parallel and distributed computing paradigm. By treating each cluster as an independent subproblem, multiple processors or computational nodes can address them concurrently. This parallelization substantially reduces total computation times and enhances scalability, making the approach well-suited for large-scale, real-time applications.

Capacity-Constrained K-means Clustering and Assignment. The work [44] proposes an adaption of k-means clustering in order to compute clusters that respect the capacity of vehicles. However, the method is computationally inefficient in the sense that it resets the centroids randomly every time the clusters fail to satisfy the capacity constraint. Furthermore, the method assumes that all vehicles are dispatched and start at the same depot, which is not the case for many real-world problems.

Motivated by the work of [45], we propose a novel alternating clustering algorithm that is made computationally efficient by utilizing the equivalent minimum cost network flow formulation. Given an initial randomly generated set of centroids, where the number of centroids matches the number of available vehicles, m, (1) assign clusters to vehicles (many-to-one matching), (2) create clusters by assigning order requests to the closest centroid while respecting vehicle capacity constraints, (3) update centroid for each cluster and return to step 1. A key challenge of this approach is that the cluster-to-vehicle assignment requires order-to-cluster information and vice versa. Alternating

between the two processes until convergence helps resolve this issue.

1. Cluster-to-Vehicle Assignment

Let V^k denote the 2-d coordinates of vehicle k. Then, we define the distance between vehicle k and centroid c to be $C_{kc} = \|V^k - C^c\|$. Based on these distances, the following optimization problem can be formulated to find the cluster-to-vehicle assignment that minimizes the total distance between vehicles and clusters. Note that the formulation allows (i) a vehicle to not be dispatched and (ii) multiple clusters to be assigned to a single vehicle. Also, note that $\sum_{r\in\mathcal{R}(c)} q_r$ is a parameter computed based on the order-to-cluster assignment of the previous iteration and represents the total load of orders assigned to cluster c.

In this formulation, z_{ck} is a binary decision variable, where $z_{ck} = 1$ if cluster c is assigned to vehicle k, and $z_{ck} = 0$ otherwise. The variable z_{ck} ensures that each cluster is assigned to exactly one vehicle while also allowing the model to determine the optimal assignment configuration that respects vehicle capacity constraints.

$$\begin{aligned}\min_{z\in\{0,1\}^{m\times m}} \quad & \sum_{(i,j)\in\mathcal{E}} C_{ck} z_{ck} \\ \text{s.t.} \quad & \sum_{k\in\mathcal{K}} z_{ck} = 1, && \forall c \in \mathcal{C} \\ & \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}(c)} q_r \right) z_{ck} \le Q_k, && \forall k \in \mathcal{K}\end{aligned} \tag{P1}$$

Equivalent Minimum Cost Flow (MCF) Formulation: In general, an MCF problem requires an underlying graph structure. Let each cluster centroid $c \in \mathcal{C}$ correspond to a supply node with supply = 1 ($\omega_c = 1$). Also, let each vehicle $k \in \mathcal{K}$ correspond to a demand node with $\omega_k = -Q_k$. Finally, let node a correspond to an artificial sink node. Then, the equivalent MCF formulation is given by the following linear program (LP):

$$\begin{aligned}\min_{\substack{y\in\mathbb{R}^{m\times m}\\ \hat{y}\in\mathbb{R}^{m\times 1}}} \quad & \sum_{k\in\mathcal{K}}\sum_{c\in\mathcal{C}} C_{ck} y_{ck} + \sigma \sum_{k\in\mathcal{K}} \max(\hat{y}_{ka}, 0) \\ \text{s.t.} \quad & \sum_{k\in\mathcal{K}} y_{ck} = 1 && \forall c \in \mathcal{C} \\ & \hat{y}_{ka} = \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}(c)} q_r \right) y_{ck} - Q_k && \forall k \in \mathcal{K}\end{aligned} \tag{P1'}$$

A figure that helps understand this equivalent formulation is provided in the Appendix.

2. Order-to-Cluster Assignment This step involves assigning orders to a nearby centroid while considering vehicle capacity constraints. Due to the capacity constraints, simply identifying the closest cluster centroid will not lead to optimal solutions. Similar to before, we can define C_{rc} to be the distance between order r and the centroid of cluster c. Then, the capacity-constrained k-means clustering can be formulated as the following optimization problem:

$$\begin{aligned}\min_{z\in\{0,1\}^{m\times m}} \quad & \sum_{r\in\mathcal{R}}\sum_{c\in\mathcal{C}} C_{rc} z_{rc} \\ \text{s.t.} \quad & \sum_{c} z_{rc} = 1 && \forall r \in \mathcal{R} \\ & \sum_{c\in\mathcal{C}(k)}\sum_{r\in\mathcal{R}} q_r z_{rc} \le Q_k && \forall k \in \mathcal{K}\end{aligned} \tag{P2}$$

Equivalent Minimum Cost Flow Formulation: The underlying graph structure for this MCF is a bit more complex than the previous one. Let each order $r \in \mathcal{R}$ correspond to a supply node with supply = 1 ($\omega_r = 1$). Let each cluster $c \in \mathcal{C}$ correspond to an intermediate node with no demand or supply. Let each vehicle correspond to a demand node, with demand equal to its capacity ($\omega_k = -Q_k$). There exists an arc between a cluster node and a vehicle node only if the cluster is previously assigned to that vehicle. All other edges between layers are connected. Finally, let node a represent an artificial sink node. Then, the equivalent MCF formulation is given by the following linear program (LP). Note that the distance between clusters and vehicles is not minimized because that relationship is fixed at this stage.

Algorithm 1: Constrained K-means Algorithm

Input: Delivery requests, $\mathcal{R}$, vehicle locations, $\{V^k\}$, initial centroids, $\{C^c\}$
Output: Clustering of orders and assignment of vehicles.
Parameter Settings: *move* = true

Assign each order $r \in \mathcal{R}$ to the nearest centroid without considering vehicle capacity constraints. $\rightarrow \mathcal{R}(c) = \{r \mid \text{order } r \text{ assigned to centroid } c\}$
while *move* = true **do**
  Assign each cluster $c \in \mathcal{C}$ to a nearby vehicle by **solving P1'** $\rightarrow$ $\mathcal{C}(k) = \{c \mid \text{centroid } c \text{ assigned to vehicle } k\}$.
  Assign each order $r \in \mathcal{R}$ to a nearby centroid while considering vehicle capacity constraints by **solving P2'** $\rightarrow$ update $\mathcal{R}(c)$.
  Update centroids using the clustered order locations:
  for $c \leftarrow 1$ **to** $|\mathcal{C}|$ **do**
    $N_c \leftarrow \sum_{r\in\mathcal{R}(c)} z_{rc}$
    $$\widetilde{C}^c = \begin{cases} \frac{1}{N_c}\left(\sum_{r\in\mathcal{R}(c)} z_{rc} X^i\right) & \text{if } N_c > 0 \\ C^c & \text{otherwise} \end{cases}$$
  $C^c \leftarrow \widetilde{C}^c \quad \forall c \in \mathcal{C}$
  Check for convergence:
  if *no point has moved to another cluster* **then**
    move $\leftarrow$ *false*

$$\begin{aligned}
\min_{\substack{y\in\mathbb{R}^{m\times m}\\ \hat{y}\in\mathbb{R}^{m\times 1}}} \quad & \sum_{r\in\mathcal{R}}\sum_{c\in\mathcal{C}} C_{rc}y_{rc} + \sigma\sum_{k\in\mathcal{K}}\max(\hat{y}_{ka},0) \\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}} y_{rc} = 1 && \forall r\in\mathcal{R} \\
& \hat{y}_{ka} = \sum_{c\in\mathcal{C}(k)}\sum_{r\in\mathcal{R}} q_r y_{rc} - Q_k && \forall k\in\mathcal{K}
\end{aligned} \tag{P2'}$$

A figure that helps understand this equivalent formulation is provided in the Appendix.

Parallel Computing for Optimal Routes Once orders are grouped into capacity-constrained clusters and assigned to vehicles, each cluster can be solved independently as a smaller MPMDPTW subproblem using the exact mathematical formulation presented in Section III. For each cluster, we extract the relevant subsets of tasks, vehicles, and constraints and construct an individual MPMDPTW instance. We then employ Gurobi's Mixed-Integer Programming (MIP) solver to optimize each subproblem. This clustering-based decomposition naturally facilitates concurrent optimization. This lends itself well to distributed computing paradigms: each cluster's route optimization can be offloaded to a separate processor or computational node. Modern parallelization frameworks, such as Python's *ProcessPoolExecutor*, execute these cluster-level optimizations concurrently, thereby significantly reducing overall computation time. As the number of clusters grows, practitioners can scale the computing resources accordingly, making the approach flexible, efficient, and better suited for large-scale, real-time applications.

V. Numerical Simulations

To evaluate the efficacy of our proposed algorithm for MPMDPTW, we devised an experimental setup that compares our clustering-based approach - where we use Gurobi optimizer to solve the resulting subproblems after orders are grouped into capacity-feasible clusters - against a direct application of Gurobi to the entire problem instance without clustering. This two-step strategy is contrasted with running Gurobi alone on the entire instance, allowing us to clearly demonstrate the advantages in computational efficiency and solution quality gained by incorporating the clustering step. Our datasets are based on the 100-task instances from Li & Lim's benchmark problems [46]. To generate a variety of test cases with different operational characteristics, we introduced controlled modifications to these base instances. For example, we advise our data generation tool to pick orders with time windows to vary their widths and overlaps, thereby influencing temporal scatteredness. We also altered vehicle capacities and the spatial placement of pickup and delivery points to create scenarios with differing levels of spatial density and complexity. Through these systematic variations, we obtained a spectrum of problem instances that differ not only in size but also in key structural properties.

We employ three metrics to capture the structural properties of the MPMDPTW problem instances: (i) spatial scatteredness (i.e., the density of requests), (ii) temporal scatteredness, and (iii) the degree of mixing of pickups and deliveries. Spatial scatteredness metric captures the geographical distribution of pickup and delivery tasks by computing the average pairwise distance among all $i, j \in \mathcal{V}$:

$$\text{Spatial Scatteredness} \;=\; \frac{2}{|\mathcal{V}|\,(|\mathcal{V}|-1)} \sum_{\substack{i,j\in\mathcal{V},\\ i<j}} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{21}$$

where $\|\mathbf{x}_i - \mathbf{x}_j\|$ is the Euclidean distance between nodes i and j. As illustrated in Figure 3, larger values indicate more spatially dispersed tasks.

Temporal Scatteredness metric measures the variance between the earliest and latest service times across orders, computed as follows:

$$\text{Temporal Scatteredness} \;=\; \max_{i\in\mathcal{V}}\big(l_i\big) \;-\; \min_{i\in\mathcal{V}}\big(e_i\big) \tag{22}$$

A larger gap indicates greater variation in service windows across the orders, as shown in Figure 2. The degree of mixing metric gauges how interspersed pickups and deliveries are based on their spatial proximity. For the set of requests $\mathcal{R}$, where each order o is composed of a pickup node $p_o \in \mathcal{P}$ and a delivery node $d_o \in \mathcal{D}$, we define:

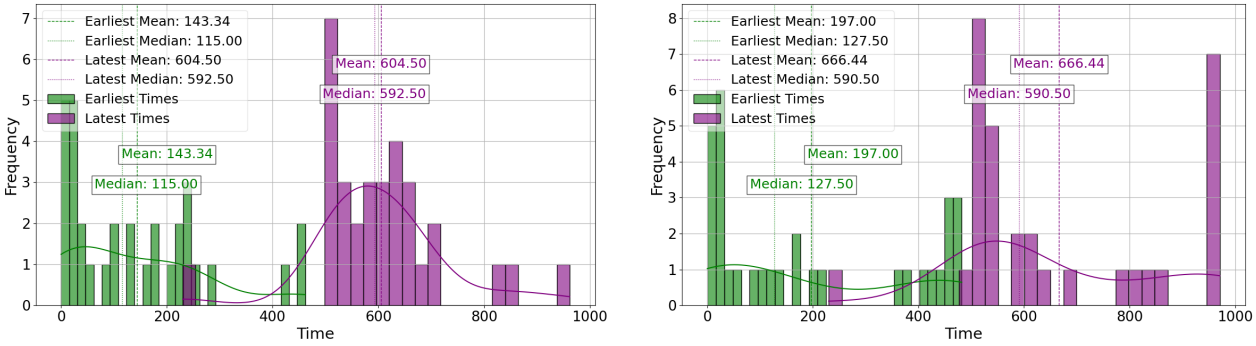
$$\text{Mixing Degree} \;=\; \frac{1}{|\mathcal{R}|} \sum_{(p_o,\,d_o)\in\mathcal{R}} \mathbf{1}\Big(\|\mathbf{x}_{p_o} - \mathbf{x}_{d_o}\| \;\le\; \delta\Big), \tag{23}$$

where $\mathbf{x}_{p_o}$ and $\mathbf{x}_{d_o}$ are the coordinates of the pickup and delivery locations for order o, respectively, and $\|\cdot\|$ denotes the Euclidean norm. The threshold δ parameter specifies a maximum allowable spatial distance between p_o and d_o. Higher MixingDegree values indicate a greater proportion of requests whose pickup and delivery locations lie sufficiently close to each other in space, thereby reflecting a more interwoven physical layout of the tasks.

These metrics enable the evaluation of a solution methodology's performance under varying spatial and temporal distributions of orders.

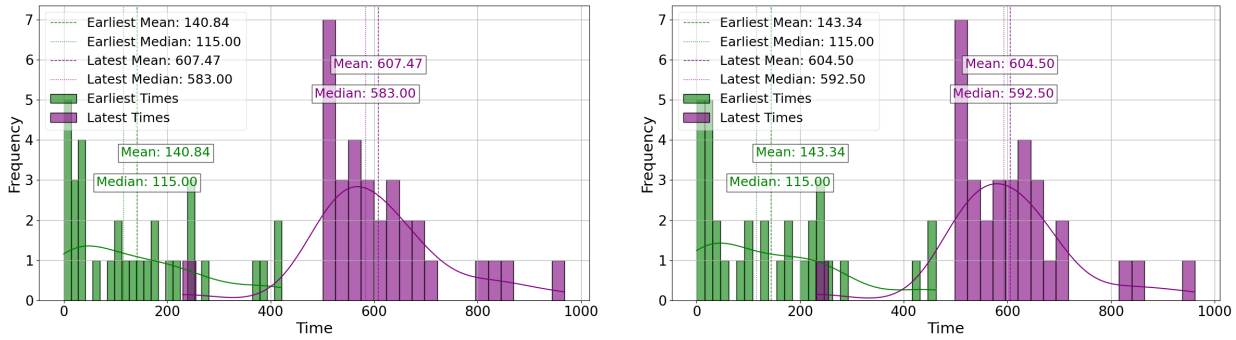
A. Numerical Results

Our numerical experiments compare the proposed Capacity-Constrained Clustering + Gurobi (C3-Gurobi) algorithm against the solutions obtained through Gurobi optimizer without clustering (Baseline Gurobi), by running extensive simulations on the aforementioned dataset. We compute the total travel time, the travel time gap, the MIP gap, and the runtime of algorithms for each test case. The travel time gap computes how much worse the C3-Gurobi solution compares against the Baseline Gurobi solution. The results illustrate the trade-offs between computational efficiency and solution quality specific to each scenario. Each dataset consists of 30 tasks (15 pickups and 15 deliveries) serviced by two drivers starting from different locations. Each driver had a capacity of 100 units, and the orders were distributed across multiple locations with time window constraints such as earliest start time, latest end time, and service durations, derived from Li & Lim's benchmark problems [46]. The computational results reveal



(a) Low temporal scatteredness (orders have similar time windows)

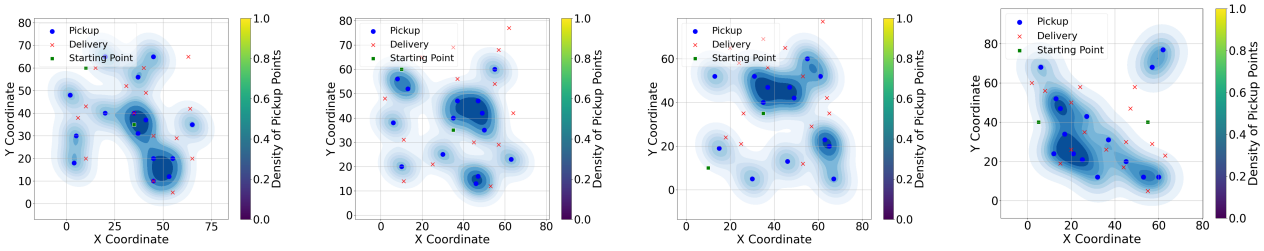
(b) Moderate temporal scatteredness



(c) Slightly more spread in time windows

(d) Widely varying time windows (high temporal scatteredness)

Fig. 2: These panels depict varying degrees of temporal scatteredness within scheduling scenarios, based on the 100-task instances from Li & Lim's benchmark problems [46]. From tasks with closely aligned time windows to those with widely varying windows, each scenario illustrates the complexity and dynamic nature of scheduling under different temporal constraints.



(a) Low spatial scatteredness (0.1)

(b) Moderate spatial density (0.3)

(c) Medium spatial density (0.5)

(d) High spatial density (0.8)

Fig. 3: These heatmaps display the density of pickup points across different scenarios, illustrating the range from low to high density. Each image is generated from Li & Lim's PDPTW benchmark data, reflecting how spatial planning varies with pickup point concentration.

the relationship between the three structural properties of data and the performance of both the Baseline Gurobi and the C3-Gurobi algorithms. With Gurobi constrained by a 60-minute time limit across all datasets, its ability to find optimal solutions was sometimes hindered, as indicated by higher MIP gap values in certain cases.

Spatial Density (see Figure 4): For experiments focused on spatial density, in the low spatial density scenario, Baseline Gurobi achieves a lower travel time cost of 499.7 minutes compared to C3-Gurobi's 511.2 minutes, resulting in a travel time gap of 2.24%. However, C3-Gurobi computed its solution with a minimal MIP gap of 0.94% in just 39.7 seconds, significantly faster than Baseline Gurobi's full runtime of 3600 seconds and MIP Gap of 3.66%. As the spatial density level increased to moderate (0.3), Baseline Gurobi maintained a lower travel time of 485.2 minutes versus C3-Gurobi's 512.1 minutes, leading to a travel time gap of 5.26%. Despite Baseline Gurobi utilizing the entire time limit, C3-Gurobi maintained a minimal MIP gap of 0.98% with a runtime of 33.5 seconds. In the medium spatial density case, Baseline Gurobi had a better travel time cost of 472.1 minutes compared to C3-Gurobi's 490.6 minutes, resulting in a travel time gap of 3.78%. C3-Gurobi, on the other hand, provided its solution in 20.3 seconds, with a negligible MIP gap of 0.93%. Under high spatial density, Baseline Gurobi achieved a lower travel time of 461.2 minutes compared to C3-Gurobi's 488.5 minutes, leading to a substantial travel time gap of 5.59%,

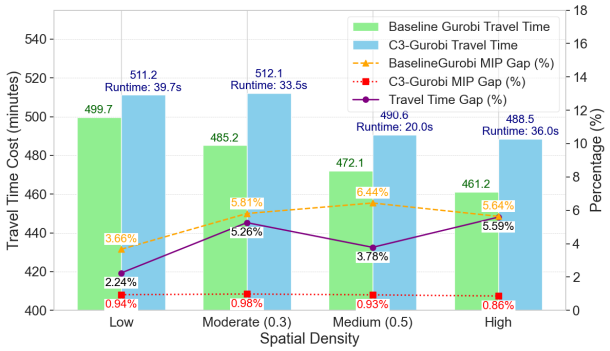


Fig. 4: Comparison of travel time and performance between Baseline Gurobi and C3-Gurobi across spatial density levels.

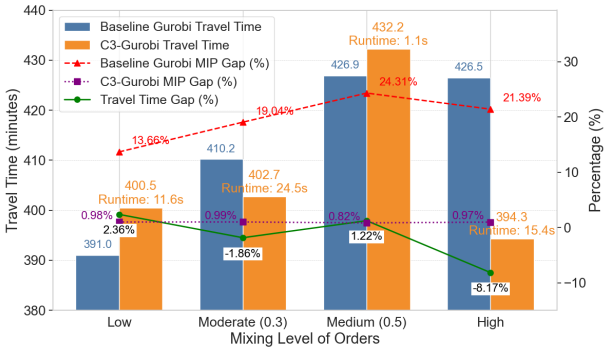


Fig. 5: Comparison of travel time cost, travel time discrepancy, MIP optimality gap, and runtime for the C3-Gurobi algorithm versus the exact approach in the mixed pickups and deliveries test case.

and C3-Gurobi delivered its solution in 36.0 seconds with a minimal MIP gap of 0.86%.

Mixing Pickup and Delivery Points: As shown in Figure 5, in the low temporal mixing scenario, Baseline Gurobi achieves a lower travel time cost of 391.0 minutes compared to C3-Gurobi's 400.5 minutes, resulting in a travel time gap of 2.36%. However, C3-Gurobi computed its solution in just 11.6 seconds, significantly faster than Baseline Gurobi's full runtime. As the mixing level increased to moderate, C3-Gurobi achieved a lower travel time of 402.7 minutes compared to Baseline Gurobi's 410.2 minutes, leading to a negative travel time gap of -1.86%. Despite Baseline Gurobi utilizing the entire time limit, it exhibited a higher MIP Gap of 19.04%, while C3-Gurobi maintained a minimal MIP gap of 0.98% with a runtime of 24.5 seconds. In the medium mixing case, Baseline Gurobi had a slightly better travel time cost of 426.9 minutes compared to C3-Gurobi's 432.2 minutes, resulting in a travel time gap of 1.22%. Yet, Baseline Gurobi's MIP gap increased to 24.31%, and it reached the time limit without further improvements. C3-Gurobi, on the other hand, provided its solution in approximately 1.1 seconds with a negligible MIP gap of 0.82%. Under high mixing, C3-Gurobi achieved a lower travel time of 394.3 minutes compared to Baseline Gurobi's 426.5 minutes, leading to a substantial negative travel time gap of -8.17% due to Baseline Gurobi's higher MIP gap of 21.39% and full runtime indicating difficulty in finding better solutions under the time constraint, whereas C3-Gurobi delivered its solution in 15.4 seconds.

Temporal scatteredness: The computational results, as shown in Figure 6, reveal that varying levels of temporal scatteredness in pickup and delivery points impact the performance of both the Baseline Gurobi solver and the C3-Gurobi algorithm, especially under Baseline Gurobi's (3600-second) runtime limit. In the low temporal scatteredness scenario, C3-Gurobi serves the orders in a lower travel time cost of 432.3 minutes compared to Baseline Gurobi's 443.7 minutes, resulting in a travel time gap of -2.63%, while completing the computation in just 24.9 seconds versus Baseline Gurobi's full runtime. As temporal scatteredness increases, C3-Gurobi continued to serve the orders with a travel time of 590.0 minutes versus 599.7 minutes, leading to a negative travel time gap of -1.63%. In

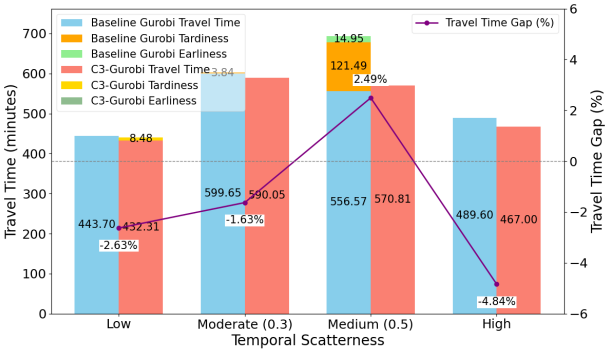


Fig. 6: Temporal scatteredness metric: comparison of travel time cost, travel time discrepancy, earliness, and tardiness for C3-Gurobi across various scenarios.

the medium temporal scatteredness case, Baseline Gurobi completed the orders with a slightly better travel time of 556.6 minutes compared to C3-Gurobi's 570.8 minutes, resulting in a travel time gap of 1.22%; however, Baseline Gurobi incurred significant tardiness and earliness, leading to a higher total cost of 693.0 minutes, while C3-Gurobi maintained a negligible MIP gap of 1.66% with a runtime of 135.4 seconds. Under high temporal scatteredness, C3-Gurobi reduced the travel time to 467.0 minutes compared to Baseline Gurobi's 489.6 minutes, achieving a substantial negative travel time gap of -4.84%, and delivered its solution faster compared to Baseline Gurobi's full runtime.

Overall, our experimental results demonstrate that the proposed C3-Gurobi algorithm significantly enhances computational efficiency compared to the Baseline Gurobi (without clustering), consistently delivering solutions within seconds versus monolithic Gurobi's hour-long runtime. While Baseline Gurobi occasionally achieved slightly lower travel time costs in low spatial density scenarios, C3-Gurobi maintained competitive travel time gaps and minimal MIP gaps across all test cases. This efficiency arises from the clustering approach, where each cluster is solved using exact methods. However, as cluster sizes grow, the runtime of these exact solvers within each cluster may increase, potentially impacting overall performance. Figure 7 exemplifies a typical routing solution, demonstrating how the clustering approach can streamline the

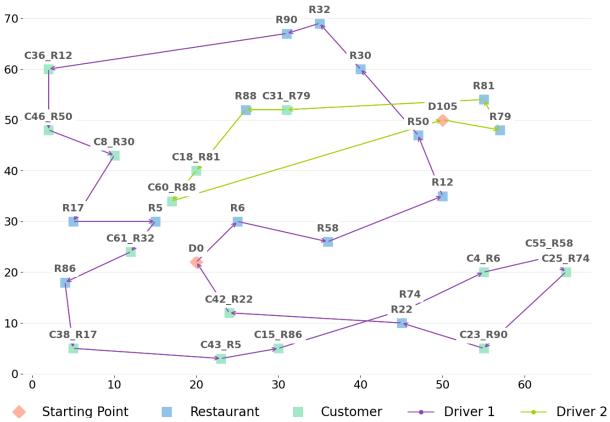


Fig. 7: Visualization of routes for 15 orders, illustrating the spatial distribution of drivers and associated pickup and delivery locations under different time window constraints such as earliest start time, latest end time, and service duration.

assignment of vehicles, orders, and routes. This combination of near-optimal performance with markedly lower computation times underscores the C3-Gurobi's potential as a scalable solution for real-world logistics challenges, where balancing speed and accuracy is essential.

VI. Conclusions and Future Directions

This paper addresses the Multi-Pickup and Delivery Problem with Time Windows by employing advanced capacity-constrained clustering techniques alongside traditional optimization methods. Our experimental analysis demonstrated the effectiveness of our heuristic compared to exact solutions, achieving near-optimal results with significantly reduced computational times.

The heuristic's capability to effectively manage spatial density, temporal scatteredness, and the integration of pickups and deliveries underscores its practical applicability to real-world logistics scenarios. It highlights the potential of integrating heuristic approaches with operational constraints to improve efficiency and responsiveness in transportation systems. Looking ahead, the integration of graph-aware deep reinforcement learning offers a promising avenue for future research. By applying this technique to optimize routing decisions within each cluster, we can potentially enhance the speed and accuracy of the solution process. This approach would leverage the structural and temporal data of routing problems to rapidly generate high-quality solutions, thereby accommodating larger and more dynamic datasets. Additionally, the adoption of adaptive algorithms that can dynamically adjust to changing environments and real-time data feeds will be crucial in enhancing the responsiveness of logistics systems to unforeseen changes in demand and vehicle availability. Additionally, exploring Multi-Agent Reinforcement Learning (MARL) to address the entire MPMDPTW problem represents another research direction. MARL can enable multiple autonomous agents to collaboratively learn optimal routing strategies, ensuring that orders are assigned correctly and efficiently, minimizing the total costs by dynamically adapting to real-time data, and reducing the impact of incorrect order assignments. This work contributes to the field of logistics and transportation by offering a scalable and efficient solution to complex routing problems, challenging traditional methods with a heuristic that balances computational speed with solution quality.

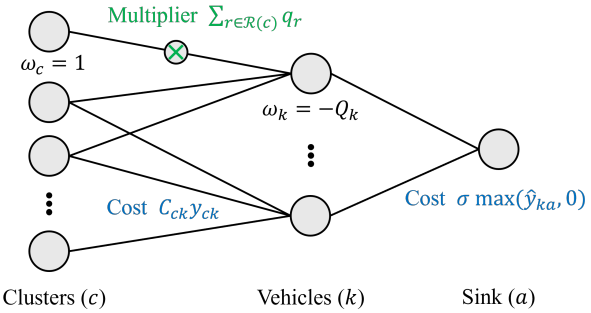


Fig. 8: Schematic of equivalent minimum cost flow formulation for problem (P1').

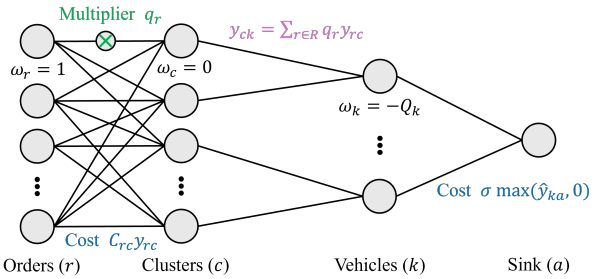


Fig. 9: Schematic of equivalent minimum cost flow formulation for problem (P2').

Appendix

In this appendix, we provide Figures 8 and 9 to help understand the equivalent minimum cost flow formulation presented in Section IV.

References

[1] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 54, no. 1, pp. 7–22, 1991.

[2] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol. 40, no. 10, pp. 1276–1290, 1994.

[3] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995.

[4] S. Mitrović-Minić, R. Krishnamurti, and G. Laporte, "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows," *Transportation Research Part B: Methodological*, vol. 38, no. 8, pp. 669–685, 2004.

[5] Q. Lu and M. M. Dessouky, "A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 175, no. 2, pp. 672–687, 2006.

[6] H. Li and A. Lim, "A metaheuristic for the pickup and delivery problem with time windows," in *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001*. IEEE, 2001, pp. 160–167.

[7] N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh, "Time slot management in attended home delivery," *Transportation Science*, vol. 45, no. 3, pp. 435–449, 2011.

[8] M. W. Ulmer, D. C. Mattfeld, and F. Köster, "Budgeting time for dynamic vehicle routing with stochastic customer requests," *Transportation Science*, vol. 52, no. 1, pp. 20–37, 2018.

[9] Y. Wang, K. Assogba, J. Fan, M. Xu, Y. Liu, and H. Wang, "Multi-depot green vehicle routing problem with shared transportation resource: Integration of time-dependent speed and piecewise penalty cost," *Journal of Cleaner Production*, vol. 232, pp. 12–29, 2019.

[10] D. S. Mankowska, F. Meisel, and C. Bierwirth, "The home health care routing and scheduling problem with interdependent services," *Health Care Management Science*, vol. 17, pp. 15–30, 2014.

[11] J. A. Nasir and Y.-H. Kuo, "A decision support framework for home health care transportation with simultaneous multi-vehicle routing and staff scheduling synchronization," *Decision Support Systems*, vol. 138, p. 113361, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923620301160

[12] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[13] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *The World Wide Web Conference*, 2019, pp. 983–994.

[14] Z. T. Qin, H. Zhu, and J. Ye, "Reinforcement learning for ridesharing: An extended survey," *Transportation Research Part C: Emerging Technologies*, vol. 144, p. 103852, 2022.

[15] J. K. Lenstra and A. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[16] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[17] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[18] J.-F. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 37, no. 6, pp. 579–594, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261502000450

[19] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004.

[20] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006.

[21] M. Gendreau, G. Laporte, and R. Séguin, "Stochastic vehicle routing," *European Journal of Operational Research*, vol. 88, no. 1, pp. 3–12, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722179500050X

[22] G. Laporte, F. Louveaux, and H. Mercure, "The vehicle routing problem with stochastic travel times," *Transportation Science*, vol. 26, no. 3, pp. 161–170, 1992.

[23] J.-F. Cordeau and G. Laporte, "A tabu search algorithm for the site dependent vehicle routing problem with time windows," *INFOR: Information Systems and Operational Research*, vol. 39, no. 3, pp. 292–298, 2001.

[24] J.-Y. Potvin and J.-M. Rousseau, "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows," *European Journal of Operational Research*, vol. 66, no. 3, pp. 331–340, 1993.

[25] I. H. Osman and N. A. Wassan, "A reactive tabu search metaheuristic for the vehicle routing problem with back-hauls," *Journal of Scheduling*, vol. 5, no. 4, pp. 263–285, 2002.

[26] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study," in *Metaheuristics for Multiobjective Optimization*. Springer, 2004, pp. 177–199.

[27] N. Jozefowiez, F. Semet, and E.-G. Talbi, "Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 271–280.

[28] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[29] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf

[30] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang, "Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2306–2315, 2021.

[31] X. Li, W. Luo, M. Yuan, J. Wang, J. Lu, J. Wang, J. Lü, and J. Zeng, "Learning to optimize industry-scale dynamic pickup and delivery problems," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2511–2522.

[32] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[33] W. Kool, H. van Hoof, J. Gromicho, and M. Welling, "Deep policy dynamic programming for vehicle routing problems," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2022, pp. 190–213.

[34] E. R. Hruschka, R. J. Campello, A. A. Freitas *et al.*, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (applications and reviews)*, vol. 39, no. 2, pp. 133–155, 2009.

[35] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 81–87, 1984.

[36] R. Dupas, T. Hsu, and E. Taniguchi, "A clustering-routing approach for assigning customers to depots in last mile delivery," *Transportation Research Procedia*, vol. 79, pp. 13–20, 2024.

[37] Y. Mo, K. Yang, S. Han, and S. Gupta, "Multi-period heterogeneous fleet vehicle routing problem with self-pickup point selection: a last-mile delivery scenario in urban and rural areas," *Annals of Operations Research*, pp. 1–35, 2024.

[38] J. Shi, "Optimization of frozen goods distribution logistics network based on k-means algorithm and priority classification," *Scientific Reports*, vol. 14, no. 1, p. 22477, 2024.

[39] S. Fatemi-Anaraki, M. Mokhtarzadeh, M. Rabbani, and D. Abdolhamidi, "A hybrid of k-means and genetic algorithm to solve a bi-objective green delivery and pick-up problem," *Journal of Industrial and Production Engineering*, vol. 39, no. 2, pp. 146–157, 2022.

[40] Y. Wang, L. Ran, X. Guan, and Y. Zou, "Multi-depot pickup and delivery problem with resource sharing," *Journal of Advanced Transportation*, vol. 2021, no. 1, p. 5182989, 2021.

[41] Y. Zhou, R. Xie, T. Zhang, and J. Holguin-Veras, "Joint distribution center location problem for restaurant industry based on improved k-means algorithm with penalty," *IEEE Access*, vol. 8, pp. 37 746–37 755, 2020.

[42] T. D. C. Le, D. D. Nguyen, J. Oláh, and M. Pakurár, "Clustering algorithm for a vehicle routing problem with time windows," *Transport*, vol. 37, no. 1, pp. 17–27, 2022.

[43] H. Wang, S. Chen, X. Yin, L. Meng, Z. Wang, and Z. Wang, "A hybrid fuzzy c-means heuristic approach for two-echelon vehicle routing with simultaneous pickup and delivery of multi-commodity," *IEEE Transactions on Fuzzy Systems*, 2024.

[44] H. Akeb, A. Bouchakhchoukha, and M. Hifi, "A beam search based algorithm for the capacitated vehicle routing problem with time windows," in *2013 Federated Conference on Computer Science and Information Systems*. IEEE, 2013, pp. 329–336.

[45] P. S. Bradley, K. P. Bennett, and A. Demiriz, "Constrained k-means clustering," *Microsoft Research, Redmond*, vol. 20, no. 0, p. 0, 2000.

[46] SINTEF, "Li & Lim benchmark," https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/, SINTEF TOP Project.