OPTIMIZING INVENTORY ROUTING: AN INTEGRATED MACHINE LEARNING SOLUTION APPROACH

A Thesis

by

Taha Huzeyfe Aktaş

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Data Science

Özyeğin University
September 2023

OPTIMIZING INVENTORY ROUTING: AN INTEGRATED MACHINE LEARNING SOLUTION APPROACH

Approved by:

Prof. O. Örsan Özener, Advisor
Dept. of Industrial Eng.
Özyeğin University

Prof. Ali Ekici
Dept. of Industrial Eng.
Özyeğin University

Assoc. Prof. Ertan Yakıcı
Dept. of Industrial Eng.
National Defense University

Date Approved: 10 August 2023

To my illuminating father and devoted mother,

Your unwavering love, wisdom, and knowledge

have been the guiding beacons on my path of learning.

Thank you for lighting the way and for your endless support.

ABSTRACT

Inventory Routing Problem (IRP) arises from vendor-managed inventory business settings where the supplier is responsible for replenishing the inventories of its customers over a planning horizon. In the IRP, the supplier makes the routing and inventory decisions together to improve the overall performance of the system. In our setting, the supplier's goal is to minimize total transportation costs over a planning horizon while avoiding stock-outs at the customer locations. We assume that the supplier has a fleet of homogeneous capacitated delivery vehicles and abundant availability of the product to be delivered to the customers. Each customer has a constant demand/consumption rate and limited storage capacity to keep inventory. To address this problem, we propose a novel integrated clustering and routing algorithm. In the clustering phase, we partition the customer set into clusters, ensuring that each cluster is served by a single vehicle. To accomplish this, we employ a novel deep learning model within the clustering framework. In the routing phase, we develop the delivery schedule for each cluster. What sets our approach apart is its consideration of the three key decisions—when to deliver, how much to deliver, and how to route—by integrating both a mathematical model and a machine learning model in the decision-making process. We evaluate the performance of the proposed clustering and routing algorithms against existing literature, and our results demonstrate significant improvements. Furthermore, the proposed neural network-based clustering approach serves as an effective representation of how machine learning algorithms can enhance decision-making structures.

Keywords: inventory routing problem; integer programming-based heuristic; neural network; machine learning based decision-making

ÖZETÇE

Envanter Rotalama Problemi (ERP), tedarikçinin bir planlama ufku boyunca müşterilerinin stoklarını yenilemekten sorumlu olduğu ve satıcının stokları yönettiği bir teslimat ve envanter planlama işidir. IRP'de tedarikçi, sistemin genel performansını iyileştirmek için yönlendirme ve envanter kararlarını birlikte verir. Bu çalışmamızda tedarikçinin amacı, bir planlama ufku boyunca toplam nakliye maliyetini en aza indirirken müşteri lokasyonlarında stok tükenmesini önlemektir. Tedarikçinin; aynı taşıma kapasitesine sahip teslimat araçlarından oluşan bir filosuna ve müşterilere teslim edilecek ürünün bol miktarda bulunabilirliğine sahip olduğunu varsayıyoruz. Her müşterinin sabit bir talep/tüketim oranı ve stok tutmak için sınırlı depolama kapasitesi vardır. Bu sorunu çözmek için yeni bir entegre kümeleme ve yönlendirme algoritması öneriyoruz. Kümeleme aşamasında, müşteri setini kümelere ayırarak, her kümeye tek bir araç tarafından hizmet verilmesini sağlıyoruz. Bunu başarmak için, kümeleme çerçevesinde yeni bir derin öğrenme modeli kullanıldı. Yönlendirme aşamasında, her bir küme için teslimat programını geliştirildi. Yaklaşımımızı diğerlerinden ayıran şey, karar verme sürecine hem matematiksel bir modeli hem de bir makine öğrenimi modelini entegre ederek üç temel kararı (ne zaman teslim edileceği, ne kadar teslim edileceği ve nasıl yönlendirileceği) dikkate almasıdır. Önerilen kümeleme ve yönlendirme algoritmalarının performansını mevcut literatüre göre değerlendirdik ve sonuçlarımız önemli gelişmeler gösterdi. Ayrıca, önerilen sinir ağı tabanlı kümeleme yaklaşımı, makine öğrenimi algoritmalarının karar verme yapılarını nasıl geliştirebileceğinin etkili bir temsili olmuştur.

Keywords: envanter rotalama problemi; tamsayı programlama tabanlı sezgisel; sinir ağı; makine öğrenimi tabanlı karar verme

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Prof. O. Örsan Özener and Prof. Ali Ekici, for their invaluable guidance, unwavering support, and continuous encouragement throughout the course of this research.

I would also like to extend my sincere appreciation to Taha Varol, whose collaboration, insightful discussions, and constructive feedback have greatly contributed to the development and refinement of this work.

I am sincerely grateful for the generous financial support provided by the Scientific and Technological Research Council of Turkey (TÜBİTAK-BİDEB) throughout my master's degree.

In addition, I am immensely grateful to my parents, Bedriye and Hasan, for their unwavering love, encouragement, and belief in my abilities. Their constant support and sacrifices have been the foundation of my academic pursuits.

I am also deeply thankful to my loving wife, Ayça, whose patience, understanding, and encouragement have been invaluable throughout this journey. Her unwavering support and belief in my abilities have been a constant source of inspiration and motivation.

Lastly, I would like to acknowledge the support and contributions of all the individuals who have been a part of this research in various ways. Their collaboration, feedback, and assistance have significantly enriched the outcomes of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

CCLP Capacitated Concentrator Location Problem. 38

CH Constructive Heuristic. 38

EOQ Economic Order Quantity. 2

IRP Inventory Routing Problem. 1

MAPE Mean Absolute Percentage Error. 32

MLB Machine Learning-based Clustering Algorithm. 41

MLP Multi-layer Perceptron. 30

MSE Mean Squared Error. 32

PSLP Pattern Selection Linear Programming. 29

TGBH Tour-Generation Based Heuristic. 13

TSP Traveling Salesman Problem. 3

VRP Vehicle Routing Problem. 1

CHAPTER I

INTRODUCTION

The *Inventory Routing Problem* (IRP) is a well-known operational research problem that involves controlling customer inventories while scheduling deliveries with minimum transportation cost. There are various variants of the IRP, but cyclic inventory routing specifically refers to the one studied by Ekici et al.[1] which assumes constant demands and continuous time. In our work, we have integrated a machine learning approach to improve the solution quality of this problem. This integration can also pave the way for applying similar modern tools to other routing problems with demand considerations.

1.1 Cyclic Inventory Routing Problem

In a vendor-managed inventory setting, the supplier is responsible for replenishing the inventories of the customers. Compared to traditional inventory systems, it has several advantages. With the help of vendor-managed inventory, the supplier can better plan its production/procurement and distribution activities and hence decrease its operational costs. To realize the potential benefits of a vendor-managed inventory setting, the supplier has to develop models and tools to make decisions including (i) when to replenish the customers and (ii) how much to deliver to each customer. The IRP integrates inventory and vehicle routing decisions that arise from such a need. In the IRP, a supplier replenishes the inventories of multiple customer locations using a fleet of vehicles, and the three main decisions to be made are: (i) when to deliver to each customer, (ii) how much to deliver, and (iii) how to route delivery vehicles to make these deliveries[2]. Compared to classical *Vehicle Routing Problem* (VRP), the main difference in IRP is the flexibility of determining the delivery amount to each

customer on each route/period. This brings the advantage of better synchronizing the deliveries to different customers considering the effect of each decision in the long run. However, such an advantage comes with an additional complexity. In addition to how to route the delivery vehicles, one has to decide on the amount of delivery to each customer on each tour, and this decision has to be made taking its long-term effect into account. More specifically, if one decides to deliver less to a customer to make a larger delivery to another customer due to limited vehicle capacity, such a decision may result in another delivery to the first customer in the following route/period.

In this paper, we study a variant of IRP where multiple customers demanding a single product are replenished from a central depot. Customers have limited storage capacity and consume the product at a constant rate during operating hours. Demands at the customer locations are realized on a continuous time basis (similar to classical *Economic Order Quantity* (EOQ) model). This demand structure has applications in the delivery of industrial gases and petroleum products [3][4][5]. Deliveries are made using a fleet of homogeneous capacitated vehicles. The supplier's goal is to develop a delivery schedule for a planning horizon with minimum total transportation cost while avoiding stock-outs at the customer locations. Inventory holding costs are handled differently in different variants of IRP. In the distribution of industrial gases, which motivates our study, customers are not part of the same company as the vendor, and hence, the inventory holding costs of the customers are not relevant [3]. We focus on *fixed partition policies* where each vehicle serves a cluster of customer locations and on each route, the vehicle makes delivery to a subset of the customers in its cluster. Although such a policy may result in sub-optimal solutions, it not only significantly simplifies the planning but also helps keep the problem tractable. Both distributors and customers prefer such policies since it allows the driver and customer to develop a personal relationship that can benefit both parties. Moreover, drivers gain increased familiarity with the region as they visit the same customers

[6][7][8]. However, even with this simplification, the problem remains quite complex and forming "good" clusters of customers for such a policy is not straightforward. The cost of serving a cluster has to be considered when partitioning the customer set into clusters, and this requires solving the single-vehicle version of the problem either exactly or approximately. Otherwise, the total cost of serving the clusters may be too high or, in the worst case, this may lead to infeasibility if there is no feasible delivery schedule for the constructed clusters. Moreover, solving the single-vehicle version of the problem itself is quite challenging.

The IRP variant studied in this paper has several practical features including the realization of demand on a continuous time basis and limiting the operating hours during the day. Under such a demand structure, the delivery time during the day becomes quite critical in terms of avoiding the stock-outs at the customer locations and determining the delivery amount to the customers. Unlike the discrete demand setting where the demand is realized at the end of the day, the routing aspect of the problem cannot be addressed by solving a *Traveling Salesman Problem* (TSP) for each delivery truck when the demand is realized on a continuous time basis. The continuous demand structure complicates the routing problem by imposing time windows for the deliveries to avoid stock-outs. In this study, we also remove any additional restrictions on the delivery strategy by allowing arbitrary delivery amounts and multiple visits to a customer in a day. These practical features and the removal of additional restrictions add additional complexity to an already difficult problem. However, we want to note that restrictions on the delivery times, number of visits per day, etc., can also be handled by modifying the integer programming model proposed for the single-vehicle case.

Since the clustering and routing decisions cyclically affect each other, we develop a solution approach that integrates the clustering and routing phases to obtain better results. We propose an enhancement of the clustering approach proposed by Ekici et

al. (2015)[1] and a novel integer programming-based routing algorithm. In Ekici et al. (2015) [1], the process of randomly generating several clusters of customers and selecting the best set of such clusters to form a partition of the customer set has the problem of having significantly overlapping clusters. This decreases the possibility of extracting a perfect partition of the customer set with minimum total cost because most of the clusters are overlapping with one another. To increase the compatibility among the clusters, we propose a modern approach where we generate much larger (possibly overlapping) subsets of the customer set and partition each subset into clusters using a well-developed machine learning framework. By doing this, we plan to evaluate massive candidate subsets and finally have a "better" partition of the customer set at the end. At this stage, any effective machine learning algorithm can be employed, as we present a promising framework that utilizes modern technology tools to enhance a complex routing problem. This is proven by the computational results where our clustering framework provides better results compared to partitioning the customer set directly using the clustering approach by Bramel and Simchi-Levi[9]. Furthermore, we propose a new versatile integer programming-based algorithm to solve the single-vehicle version of the problem more effectively. This algorithm uses a set of pre-generated tours to solve the single-vehicle version of the problem and works well regardless of the clustering approach used to partition the customer set. In this algorithm, for a cluster of customers we pre-generate "good" delivery routes and solve an integer programming formulation to determine which routes to use, when to execute them and how much to deliver to each customer. If all possible routes of a cluster are to be generated, this approach would find the optimal solution. However, generating all the routes is not a polynomial task. Hence, we limit the cluster size and generate only a subset of routes (specifically, the cost-effective ones). Considering the real-life applications of the IRP in certain industries including industrial gas distribution [10] and cash management in automated teller machines [11], limiting the

size of each cluster is not an unrealistic assumption. We compare the performance of the proposed approach against the approach by EKici et al.[1] using the same instances and trying different combinations of clustering and routing algorithms. Even though Ekici et al.[1] already find solutions with quite low optimality gaps, we not only obtain better results in terms of optimality gaps but also solve one of the instances for which Ekici et al.[1] could not find a feasible solution. The average optimality gap of the solutions found by the new approach is only 2.38%.

Our contributions in this study can be summarized as (i) a new machine learning-based clustering algorithm to partition the customer set, and (ii) a new versatile integer programming-based heuristic algorithm to solve the single vehicle problem of a complex variant of the IRP. Both the clustering and routing algorithms perform quite well independently of the other. Moreover, unlike the standard IRP instances used in the literature [12] [13] [6] the proposed algorithms are tested on larger instances and can find good solutions for the instances with up to 14 periods.

The rest of the paper is organized as follows. In the remainder of this chapter, we review the relevant literature. We provide a formal problem definition in Chapter 2. Chapter 3 is divided into two sections. In Section 3.1, we discuss the routing algorithm proposed for solving the single-vehicle version of the problem. We explain the integrated approach proposed for the multi-vehicle case in Section 3.2. We give details of our proposed machine learning algorithm in Chapter 4. We present the results of the computational study in Chapter 5. Finally, concluding remarks are provided in Section 6.

1.2 Literature Review

The IRP is first introduced by Bell et al.[14], and the first implementation is in the distribution of industrial gases. Following this study, several variants of IRP each of which has application-specific features have been studied in the literature. We refer

the reader to [15] and [16], [17] for reviews of IRP literature.

Variants of the IRP differ in many aspects including the objective function, replenishment policy, demand structure, product availability, fleet structure and storage capacity. A detailed discussion of these aspects is provided by [18], [16] and [15]. A typical objective function includes transportation and/or inventory-related costs. Transportation-related costs may include (i) variable routing cost [19][1][20] and (ii) fixed vehicle/routing cost [21][22][23]. Inventory related costs may include (i) holding cost at the depot and/or customer locations [24][25][26], (ii) shortage cost [11][27], and (iii) fixed order/setup cost [28][29][30]. Since the optimal solution can be quite complicated and unstructured, many studies focus on solutions with more structured replenishment policies. For example, in the fixed partition policy the set of customer locations is partitioned into some clusters and each cluster is served independently [31][26], in the order-up-to-level policy the inventory of each demand location is filled up to a maximum level at every visit [32][6][33], and in the zero-inventory-ordering policy a customer is replenished if and only if its inventory is zero [34][22]. Both stochastic and deterministic demand settings are studied in the literature [35][36][37]. While most studies assume that the demand is satisfied immediately after the delivery [24][12][35], some assume that the demand is realized at a certain rate throughout the day [3][38]. When the demand is realized at a constant rate throughout the day, the delivery time to a customer, and hence the visiting order of the customer in a route becomes important. In the setting where the demand at the customer locations is realized immediately, after determining the customers to be visited and the delivery amounts on a given day, then the problem turns into a VRP (or TSP if there is only one delivery vehicle). However, in the case of a demand realized on a continuous time basis, even if the delivery amounts are determined, determining efficient delivery routes is not trivial. One has to take not only the travel cost but also the demand level of the customers and the delivery time to the customers into account

when determining the visiting order of the customers. While some studies assume that the products to be distributed are already available at the depot [39][40], some others analyze the case where the items arrive at the depot in waves at the beginning of each period [41][42][30]. Most studies assume a homogeneous fleet of delivery vehicles, but a heterogeneous fleet setting is also investigated [43][42]. Finally, in some cases customer locations and/or the depot may have limited storage capacities [6][44].

Most of the heuristic algorithms proposed in the literature decompose the problem into two phases: (i) determining the delivery amount, and (ii) vehicle routing [3][29]. In the first phase, the delivery amount is determined for each customer for different periods, and then the routes are constructed to solve a VRP. However, such approaches ignore the interaction between the delivery amount determination and route construction. To address this interaction, some suggested iterating between these two phases to find better solutions [35][45][46].

In terms of the demand structure, several studies assume discrete demand where the demand is realized instantly at the customer locations. For example, Coelho et al.[6] analyze a variant where the objective function includes inventory holding cost at the customer locations and the depot. Customer locations are assumed to have limited storage capacity, and the products to be distributed arrive at the depot at the beginning of each period. They look for a solution under the order-up-to-level policy and develop an adaptive large neighborhood search-based matheuristic. Desaulniers et al. [25] study a similar setting and develop a new mathematical model and new valid cuts. They solve the problem using a branch-price-and-cut algorithm. For the same problem, Avella et al.[47] develop a branch-and-cut algorithm that uses valid inequalities for the single-period setting. Archetti et al.[12] develop a matheuristic that utilizes a tabu search algorithm to solve the same problem.

Archetti et al.[13] and Van Anholt et al.[48] introduce a new variant of IRP with pickups and deliveries where some locations supply the product and others demand

the product with different amounts in each period. Archetti et al.[13] analyze a single-vehicle version of the problem and solve the problem with a branch-and-cut method. Van Anholt et al.[48] consider the multi-vehicle version and propose a cluster-first route-second approach where each customer is assigned to one of the depots and then the single depot formulation is solved by a branch-and-cut method after strengthening it with valid inequalities.

Similar to the demand structure in our study, Chitsaz et al.[22], Raa and Dullaert[26] and Zenkar et al.[23] assume that the demand is realized continuously at the customer locations and focus on cyclic delivery policies. Moreover, they use a homogeneous fleet for delivery and do not allow stock-outs at customer locations. However, they visit each customer in one route only, and hence the stock-outs can be avoided easily. As long as the frequency of a route is determined to meet the demand between two consecutive visits at a customer location, stock-out does not occur. Moreover, the routing decisions can be made relatively easily by solving a TSP, and there is no daily operating hours limitation. In our study, we allow multiple routes to visit a customer location, and in this case, the necessity of synchronization between the routes and the delivery amounts on each route makes the problem quite complicated [49]. Finally, unlike our study, these studies assume that the customer locations have unlimited storage capacity.

The studies most related to the current study are the ones by [3] and [1]. Campbell and Savelsbergh[3] study a variant of IRP that is similar to the one in this study. They focus on a rolling horizon solution approach and propose a two-phase heuristic algorithm. They first determine how much to deliver to each customer in each period. They generate a large set of potential clusters of customers and determine an approximate value for the cost of serving the customers in each cluster by solving an approximate integer program. Then, they solve a set partitioning problem to determine the final clusters of customers. In the second phase, using the delivery amounts

determined in the first phase they solve a series of vehicle routing problems with time windows (VRPTW). They solve the VRPTW using an insertion heuristic developed in an earlier study[50].

Ekici et al.[1] study the same problem and propose an integrated clustering and routing approach. They also focus on a fixed partition policy where each vehicle serves a subset of customers only, and each subset of customers is served by one vehicle only. They first generate several (possibly overlapping) clusters of customers randomly. Then, they estimate the cost of serving each cluster by a lower bound using the lower bounding mechanism proposed by Song and Savelsbergh[10]. Then, they solve a series of set partitioning problems iteratively to determine the final partition of the customer set among the clusters generated. Each time they choose a set of clusters after solving a set partitioning problem, they update the cost of serving each cluster by an upper bound which is found by solving the single vehicle version of the problem. To solve the single-vehicle version of the problem, they develop a constructive heuristic algorithm.

Our solution framework is inspired by Ekici et al. [1], in the sense that we generate several (possibly overlapping) clusters of customers and select among them to form a partition of the customers. However, unlike their iterative framework, we do not employ an iterative approach. Our objective is to create a large number of candidate clusters and estimate their cost using a neural network model. Rapid estimations allow us to evaluate a large number of clusters. This contribution helps eliminate the bias that may arise from a 'good' partition of such clusters, as we evaluate nearly all reasonable clusters within our machine learning framework. Furthermore, our methodology solves the relatively expensive set partition model only once instead of iteratively solving it. Additionally, our approach is novel in the literature as it utilizes machine learning methods in the decision-making mechanism of an IRP. Moreover, we propose a new integer programming-based routing algorithm for the single-vehicle

case. In this routing algorithm, we solve a mixed-integer programming formulation to make the interdependent routing, delivery amount and delivery time decisions simultaneously. We simplify the mixed-integer model by pre-generating "good" routes and feeding them into the model. Compared to the study by [1], the new solution approach is quite versatile, and hence can be used for other variants of the IRP as well. Since the new routing algorithm is based on solving a mixed-integer model using a set of pre-generated tours, variants of the IRP with inventory holding cost, fixed vehicle/routing cost, discrete demand structure, specific replenishment policies and a heterogeneous fleet of vehicles can easily be addressed by modifying the objective function and/or constraints.

CHAPTER II

DETAILED PROBLEM DESCRIPTION

In this section, we present a formal problem definition. The IRP setting studied in this paper is defined on a Euclidean graph $G = (V_0, E)$ where $V_0 = \{0, 1, 2, \ldots, n\}$ is the set of customer locations ($V = \{1, 2, \ldots, n\}$) and the depot (0), E is the set of edges connecting the nodes in V_0. For each edge, there is an associated traversal cost and time. We use c_{ij} to denote the traversal cost of edge (i, j) and t_{ij} to denote the traversal time of edge (i, j). Each customer location and the depot operates for K hours per day. This daily operating hours restriction complicates the problem since the routes on any given day have to be completed before the end of operating hours. During the operating hours, each customer i demands/consumes a single product at a constant rate denoted by q_i per day. The daily consumption at a customer location is realized at a uniform rate throughout the day. Customer i has a storage capacity of C_i. At any point in time, this capacity cannot be exceeded and stock-out at any customer location is not allowed. There are M homogeneous capacitated vehicles located at the depot. We use Q to denote the capacity of a delivery vehicle. When a delivery vehicle is dispatched, it visits a subset of customers and delivers a certain amount of product to each and then returns to the depot. Each vehicle can perform multiple tours during a day as long as it returns to the depot by the end of the day. Each customer has an initial inventory of $\mathcal{I}_i^0$ at the beginning of the cyclic planning horizon.

Our goal is to develop a delivery schedule (which includes the delivery routes, their execution times and the delivery amount to each customer on each route) with

minimum total transportation cost. For practicality reasons, we focus on a fixed partition policy and a cyclic delivery schedule which is repeated every T days. That is, the set of customers is partitioned into clusters of customers such that each cluster is served by a single vehicle, and the delivery schedule of each cluster is periodic in the sense that it is repeated with a certain frequency. Hence, at the end of T days, each customer has an inventory equal to its initial inventory. By constructing a cyclic delivery schedule, we do not allow the model to take advantage of the starting inventory and finish the planning horizon with zero inventory at the customer locations. Since a delivery schedule that is cyclic for a submultiple of T days is cyclic for T days as well, we run the proposed approach for submultiples of T and report the best solution. We limit the size of each cluster by *five*. While this approach might lead to sub-optimal solutions, it's important to note that removing the cluster size limit generally did not result in improved solutions. In some scenarios, further enhancements can be made to the machine learning model in order to effectively handle clusters with larger customer counts.

CHAPTER III

SOLUTION APPROACH

We provide detailed explanations of our solution approach in this chapter, which is divided into two subcategories based on our cluster-first-route-second approach. In the first part, we analyze the scenario of serving all customers with a single vehicle and present a corresponding mathematical model. Subsequently, we address the question of customer partitioning into non-overlapping clusters, enabling us to treat each cluster as an individual single-vehicle case.

3.1 Single Vehicle Case

The problem's single vehicle case is introduced and a sub-routine called the *Tour-Generation Based Heuristic (TGBH)* is presented as a solution approach. The TGBH method aims to schedule deliveries to customers in a cluster denoted as $\hat{V}$ with the objective of minimizing transportation costs. This sub-routine is used in the proposed solution approach to find a solution for a given subset of customers.

The proposed solution approach, TGBH, considers three key decisions simultaneously. These decisions include (i) the time of dispatch, (ii) the route to be followed, and (iii) the delivery amount to each customer in the route. As the problem is multi-period in nature, each decision has a significant impact on the future schedule. For instance, delivering a low amount to a customer in the current route may result in a direct route to this customer in the upcoming route. Similarly, late deliveries may provide more available storage capacity for a customer, enabling larger amounts to be delivered. However, it is important to avoid stock-out situations, which may require frequent visits to certain customers.

TGBH considers interactions between these three decisions by a mathematical

model. A pre-processing step of generating delivery routes is required to solve TGBH. Thus, TSP is solved for every subset of customers in $\hat{V}$. Given that an increase in the number of customers within a cluster can lead to the infeasibility of meeting demands for all customers, we have constrained the cluster size to a maximum of five. This limitation not only ensures the feasibility of solutions but also reduces computational time to a reasonable level. For every possible subset of the customers in $\hat{V}$, we solve a TSP to find the route that visits all the customers in the subset with minimum cost. The number of tours that a vehicle can execute during a planning horizon is limited, which gives to model an upper bound to state the maximum number of tours that a vehicle can execute. We assume that the total number of routes executed is equal to this upper bound. As we may not want to execute a delivery route for each order, we include a dummy route with zero cost and zero delivery amounts, which corresponds to skipping an execution order.

The following notation is used to describe the mixed integer mathematical model for the single vehicle case of the problem.

$\mathcal{P}$ = Set of all feasible delivery routes including the dummy route

$\mathcal{P}^-$ = Set of all feasible delivery routes except the dummy route

θ_j = Traveling cost of delivery route $j \in \mathcal{P}$(Optimal cost of the corresponding TSP instance)

β_j =: Duration (in hours) of delivery route $j \in \mathcal{P}$

$$a_{ij} : \begin{cases} 1, & \text{customer } i \in V \text{ is visited in route } j \in \mathcal{P} \\ 0, & \textit{otherwise} \end{cases}$$

α_{ij} = Time (in hours) to reach customer i when delivery route j is started

(zero if customer i is not visited)

U = Maximum number of routes to executed in a cycle, $U = \left\lceil \frac{T \sum_{i \in \hat{V}} q_i}{Q} \right\rceil$

Following decision variables are used:

S_l = Start time of the route at order l $\quad \forall l \in \{1, 2, \ldots, U\}$

I_{il} = Inventory at customer i at the beginning of the route at order l $\quad \forall i \in \widetilde{V}, l \in \{1, 2, \ldots, U+1\}$

m_l = Day on which the route at order l is executed $\quad \forall l \in \{1, 2, \ldots, U\}$

d_{ijl} = Delivery amount to customer i by route j at order l $\quad \forall i \in \widetilde{V}, j \in \mathcal{P}, l \in \{1, 2, \ldots, U\}$

$$x_{jl} = \begin{cases} 1, & \text{if route } j \text{ is executed at order } l \\ 0, & \text{otherwise} \end{cases} \qquad \forall j \in \mathcal{P}, l \in \{1, 2, \ldots, U\}$$

The resulting mixed-integer programming formulation is given below:

$$\text{Min} \quad \frac{1}{T} \sum_{l=1}^{U} \sum_{j \in \mathcal{P}} \theta_j x_{jl} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{P}} x_{jl} = 1 \qquad \forall l \in \{1, 2, \ldots, U\} \tag{2}$$

$$S_l + \sum_{j \in \mathcal{P}} \beta_j x_{jl} - S_{l+1} \leq 0 \qquad \forall l \in \{1, 2, \ldots, U-1\} \tag{3}$$

$$K(m_l - 1) - S_l \leq 0 \qquad \forall l \in \{1, 2, \ldots, U\} \tag{4}$$

$$K m_l - \left(S_l + \sum_{j \in \mathcal{P}} \beta_j x_{jl} \right) \geq 0 \qquad \forall l \in \{1, 2, \ldots, U\} \tag{5}$$

$$I_{i1} - (\mathcal{I}_i^0 - S_1 \frac{q_i}{K}) = 0 \qquad \forall i \in \widetilde{V} \tag{6}$$

$$I_{i,l-1} + \sum_{j \in \mathcal{P}} d_{ij,l-1} - \left((S_l - S_{l-1}) \frac{q_i}{K} + I_{il} \right) = 0 \qquad \forall i \in \widetilde{V}, l \in \{2, 3, \ldots, U\} \tag{7}$$

$$I_{iU} + \sum_{j \in \mathcal{P}} d_{ijU} - \left((KT - S_U) \frac{q_i}{K} + I_{i,U+1} \right) = 0 \qquad \forall i \in \widetilde{V} \tag{8}$$

$$I_{i,U+1} - \mathcal{I}_i^0 \geq 0 \qquad \forall i \in \widetilde{V} \tag{9}$$

$$I_{il} \leq C_i \qquad \forall i \in \widetilde{V}, l \in \{1, \ldots, U+1\} \tag{10}$$

$$I_{i,l-1} - \left(\sum_{j \in \mathcal{P}} x_{j,l-1} \alpha_{ij} \frac{q_i}{K} \right) \geq 0 \qquad \forall i \in \widetilde{V}, l \in \{2, 3, \ldots, U+1\} \tag{11}$$

$$I_{i,l-1} + \sum_{j \in \mathcal{P}} d_{ij,l-1} - \left(\sum_{j \in \mathcal{P}} x_{j,l-1} \alpha_{ij} \frac{q_i}{K} \right) \leq C_i \qquad \forall i \in \widetilde{V}, l \in \{2, 3, \ldots, U+1\} \quad (12)$$

$$\sum_{i \in \widetilde{V}} d_{ijl} a_{ij} - Q x_{jl} \leq 0 \qquad \forall j \in \mathcal{P}, l \in \{1, 2, \ldots, U\} \quad (13)$$

$$\sum_{i \in \widetilde{V}} d_{ijl} (1 - a_{ij}) = 0 \qquad \forall j \in \mathcal{P}, l \in \{1, 2, \ldots, U\} \quad (14)$$

$$\sum_{i \in \widetilde{V}} \sum_{j \in \mathcal{P}} \sum_{l=1}^{U} d_{ijl} - T \sum_{i \in \widetilde{V}} q_i = 0 \quad (15)$$

$$m_l \in \{1, 2, \ldots, T\} \qquad \forall l \in \{1, 2, \ldots, U\} \quad (16)$$

$$x_{jl} \in \{0, 1\} \qquad \forall j \in \mathcal{P}, l \in \{1, 2, \ldots, U\} \quad (17)$$

$$S_l \geq 0 \qquad \forall l \in \{1, 2, \ldots, U\} \quad (18)$$

$$I_{il} \geq 0 \qquad \forall i \in \widetilde{V}, l \in \{1, 2, \ldots, U+1\} \quad (19)$$

$$d_{ijl} \geq 0 \qquad \forall j \in \mathcal{P}, i \in \widetilde{V}, l \in \{1, 2, \ldots, U\} \quad (20)$$

The objective function (1) expresses the minimization of average transportation cost per day. Constraints (2) guarantee that at each order exactly one delivery route is executed, which may be the dummy route as well. Constraints (3) ensure that the consecutive executions of the delivery routes do not overlap. Constraints (4-5) ensure that each delivery route starts and ends within the same day. Constraints (5) also guarantee that the execution of all the delivery routes is completed before the end of the planning cycle. Constraints (6-8) are the inventory flow-balance equations. Constraints (9) guarantee that the ending inventory at the end of the planning cycle is equal to the inventory at the beginning of the cycle. Constraints (10-12) make sure that customers do not have stock-outs and excess inventory. Constraints (13) guarantee that the vehicle capacity is not exceeded for each delivery route. The delivery amounts to the customers that are not visited on a given delivery route are set to zero by Constraints (14). Constraint (15) guarantees that the total delivery amount to the customers over the planning cycle is exactly equal to the total demand

of the customers over the planning cycle. Although this constraint is redundant, we retain it in the formulation for ease of understanding. Finally, the remaining constraints are nonnegativity, integrality and binary constraints.

In our setting, due to demand structure and storage capacity at the customer locations, waiting along a route has the potential to improve the solution by reducing the number of deliveries to certain customers and thus, the total routing cost. Although we do not allow waiting along a route in the proposed model, it can be handled by defining an additional decision variable for the amount of waiting time for each customer along the route at each order.

The main disadvantage of TGBH is possibly longer computational times to obtain a satisfactory solution (depending on the instance, even a feasible solution). On the other hand, by using such a method the selection of the delivery routes and their execution times and the determination of the delivery volumes are simultaneously optimized.

3.2 Multi-Vehicle Case

In this section, we explain the integrated clustering and routing approach proposed for the multi-vehicle version of the problem. The clustering phase constitutes a critical component of our approach, and it is here where our machine learning methods play a key role. By identifying high-quality clusters, we can effectively reduce transportation costs. However, the combinatorial explosion of possible cluster combinations poses a significant computational challenge. For instance, in an instance with 50 customers and 15 vehicles, the number of possible clusters can be on the order of $\frac{(50+15-1)!}{15!}$, which makes exhaustive evaluation of all combinations infeasible. To address this challenge, intelligent methods are required to identify high-quality clusters in a reasonable time frame. In this regard, our approach leverages machine learning techniques to enable rapid and efficient evaluation of numerous clusters, thus facilitating the identification

of comparable good cluster configurations with the help of a well-trained machine learning model.

The flow of our algorithms is illustrated in algorithm 1. Our approach consists of three main stages: (i) generating a large number of candidate clusters, (ii) making rapid estimations of these clusters and (iii) selecting clusters for the best partition. To achieve rapid estimations, we developed a machine learning model that estimates the IRP cost. Each step of the algorithm is explained in detail below.

Algorithm 1 *Clustering stage*

1: **Input:** IRP Instance
2: **Output:** high quality customer partition
3: Generate massive number of clusters[51] by $2 : K$
4: Estimate the cost of serving each cluster with proposed algorithm 3, $\forall k \in K$
5: Solve proposed set partitioning model with estimations

3.2.1 Generation of Candidate Clusters

The first step in our approach is to generate candidate clusters. Ideally, we would like to generate all possible clusters and know the exact cost of serving each of these clusters, which would allow us to choose the best clusters based on our solution approach. However, this is not feasible when the total number of clusters becomes large due to the huge number of possible subsets of customers. Even if we could generate all possible clusters, it would be computationally expensive and impractical to store all of them. Moreover, explained the single-vehicle problem is computationally expensive, so calculating the exact cost for even a few clusters requires considerably large times. Therefore, an intelligent way should be followed to generate a large but manageable number of candidate clusters.

While it is impractical to exhaustively explore all feasible candidate solutions, generating as many high-quality clusters as possible can significantly expand the search space, particularly since their cost can be rapidly estimated using our proposed solution approach. To achieve this, we used utilized version of approximate stability

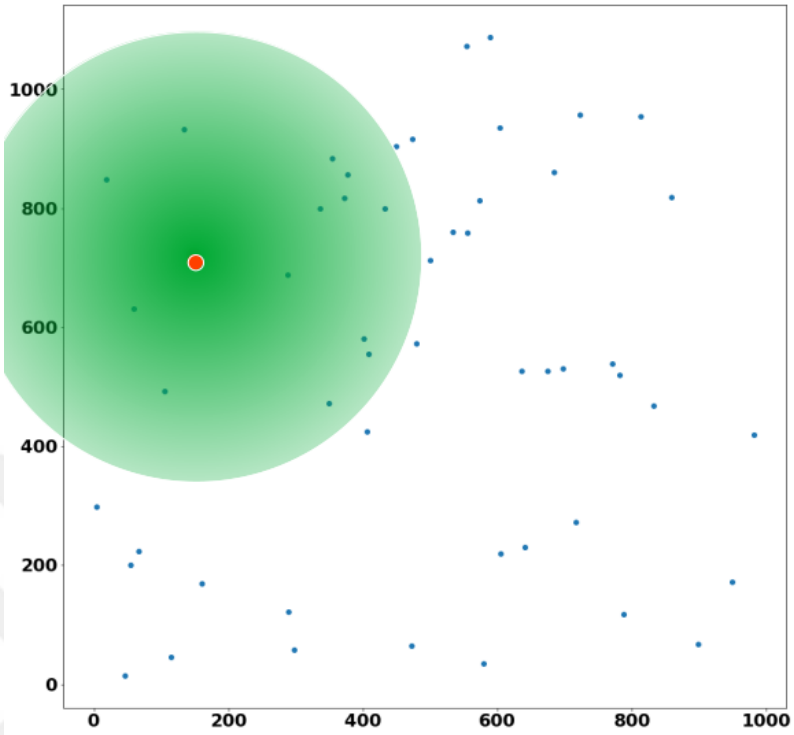

Figure 1: Cluster generation

assessment method proposed by Ozener et al. [51]. Figure 1 shows this process. This involves randomly selecting a point, the red point in figure 1, from the map and assigning a probability to each customer for building a cluster near this point. Darker green colors spot the areas where the selection of a customer is more likely in figure 1. The probability assigned to each customer is determined using the following calculation:

$$D = (\textit{length in } x \textit{ coordinate} + \textit{length in } y \textit{ coordinate})/4$$

$$d_{ip} = \textit{ distance of customer } i \in V \textit{ to random selected point}$$

$$p_i = \begin{cases} 0, & \textit{if } d_{ip} \geq D \\ \sqrt{1 - (d_{ip}/D)}, & \textit{otherwise} \end{cases}$$

To generate clusters, we calculate the probability for each customer and shuffle

them for every selected point. To ensure diversity in cluster sizes, we limit the number of appended customers and allow clusters of varying sizes. However, clusters with more than a few customers are generally not qualitative due to time window constraints in the problem. Therefore, our machine learning algorithm is designed to work up to 5 customers. Consequently, the largest cluster size is limited to 5. We also adopt the probability calculation method proposed by Ekici et al.[1] and further enrich our clusters by including all subsets of generated clusters. Algorithm 2 shows pseudo-code to create clusters.

Algorithm 2 *Cluster Generation*

Input: IRP Instance
Output: Candidate Clusters (K)
n: number of customers
V: customers set
N: number of clusters to be created for each size
$x \leftarrow 0$
$K \leftarrow [\,]$
for $clusterSize \in [1, 2, 3, 4, 5]$ **do**
  $counter \leftarrow 0$
  while $counter < N$ **do**
    $p \leftarrow$ *spot a random point on map*
    $probabilities \leftarrow (probability\ calculation(V) = [p_1 \dots p_n]$
    $V \leftarrow shuffle(V)$
    $K_x \leftarrow [\,]$
    for $i \in V$ **do**
      if p_i **then**
        $K_x \leftarrow K_x.append(i)$
        if $length(K_x) = clusterSize$ **then**
          break
        end if
      end if
    end for
    $x \leftarrow x + 1$
    $counter \leftarrow counter + 1$
  end while
end for
return K

3.2.2 Machine Learning Estimations of IRP

Our approach aims to estimate the routing costs of clusters. In the routing phase section, we present a mat-heuristic method to solve the routing phase of clusters. However, the mathematical model utilized in this method is computationally intensive and can require a significant amount of time to solve. This novel approach is motivated by the question of what if we could quickly solve a large number of clusters to select the best cluster partition. We explore the use of machine learning techniques to estimate the routing phase costs of numerous clusters in a short amount of time. By leveraging machine learning techniques, we can efficiently estimate the routing costs of many clusters, thus enabling the rapid identification of optimal routing configurations.

The proposed machine learning approach is the contributed part of this study. Therefore, it is explained in detail in the upcoming chapter 4.

3.2.3 Cluster Selection

The machine learning algorithms employed in this study enable the estimation of the cost of generated subsets. Subsequently, the optimal partition of candidate clusters is selected to solve the routing phase of the problem. To achieve this, a set partition problem was developed to identify the best-estimated clusters from among the candidate clusters. Decision variables and problem parameters in the set partition model are defined as follows:

Sets:

$K = \{1, \ldots, k, \ldots, \mathrm{K}\}$*: Set of clusters*

$V = \{1, \ldots, k, \ldots, \mathrm{V}\}$ *: Set of customers*

Parameters:

e_k : *Cost estimation of cluster* $k \in K$

$L : the maximum distance a truck can travel within its operating hours$

$M : the number of vehicles available$

$$v_{ik} : \begin{cases} 1, & \textit{customer } i \in V \textit{ is included in cluster } k \in K \\ 0, & \textit{otherwise} \end{cases}$$

Decision variable:

$$x_k : \begin{cases} 1, & \textit{cluster } k \in K \textit{ is selected in the partition} \\ 0, & \textit{otherwise} \end{cases}$$

Set partition model is formulated as follows:

$$\min \quad \sum_{k \in K} e_k * x_k \tag{21}$$

$$st. \quad \sum_{k \in K} x_k v_{ik} = 1 \qquad \forall i \in V \tag{22}$$

$$\sum_{k \in K} x_k <= M \tag{23}$$

$$e_k * x_k <= L \qquad \forall k \in K \tag{24}$$

$$x_k \in \{0, 1\} \qquad \forall k \in K \tag{25}$$

The objective 21 is designed to minimize the total estimated cost of selected subsets in the set partition problem. To ensure that each customer belongs to exactly one of the selected subsets, constraint 22 is introduced. Constraint 23 restricts the

number of selected clusters by the number of available vehicles. Constraint 24 ensures that any cluster with an estimated cost that is larger than a vehicle can travel in operation hours is not included in the partition of clusters. In addition, constraint 25 is a binary restriction. By solving the integer programming formulation, we obtain a partition of the generated subsets with a maximum of M clusters. We then evaluate the top solutions found by the mathematical model by solving their routing phase in two periods and selecting the best partition among them. This post-process approach helps to reduce the impact of errors in regression models.

CHAPTER IV

IRP ESTIMATIONS VIA MACHINE LEARNING

This section will provide information on how machine learning algorithms are designed to estimate the cost of candidate IRP clusters. The machine learning model development process consists of three branches: generation of massive data to train a model, feature engineering, and modeling. We also explained the framework in the last part of this chapter to explain how we utilize developed models for the IRP.

4.1 Massive Data Creation

Ekici et al.[1] studied the same problem using a batch of instances. The paper provides a detailed description of the real-world instance generation method, which will be discussed in depth in Section 5.1. According to the proposed approach, the distribution of customers can be uniform across the map or concentrated around a virtual point. Additionally, each customer is randomly assigned specific features related to their storage capacity and demand. Therefore, this method is highly suitable for increasing the generalization power of the data. We adopted a similar method to create a large number of instances.

We aim to estimate the objectives of clusters that contain 3, 4, or 5 customers. Since the cost of clusters with fewer than 3 customers can be rapidly estimated, they are not included in the training of the model. Furthermore, serving more than 5 customers is generally inefficient and can lead to infeasibility, so they are also not included in the training process. Therefore, we create three clusters of different sizes from each instance generated. This approach ensures that the distribution of cluster sizes in the training data is uniformly distributed. The clusters are generated using the same method described in Section 3.2.1. The final part involves determining the labels

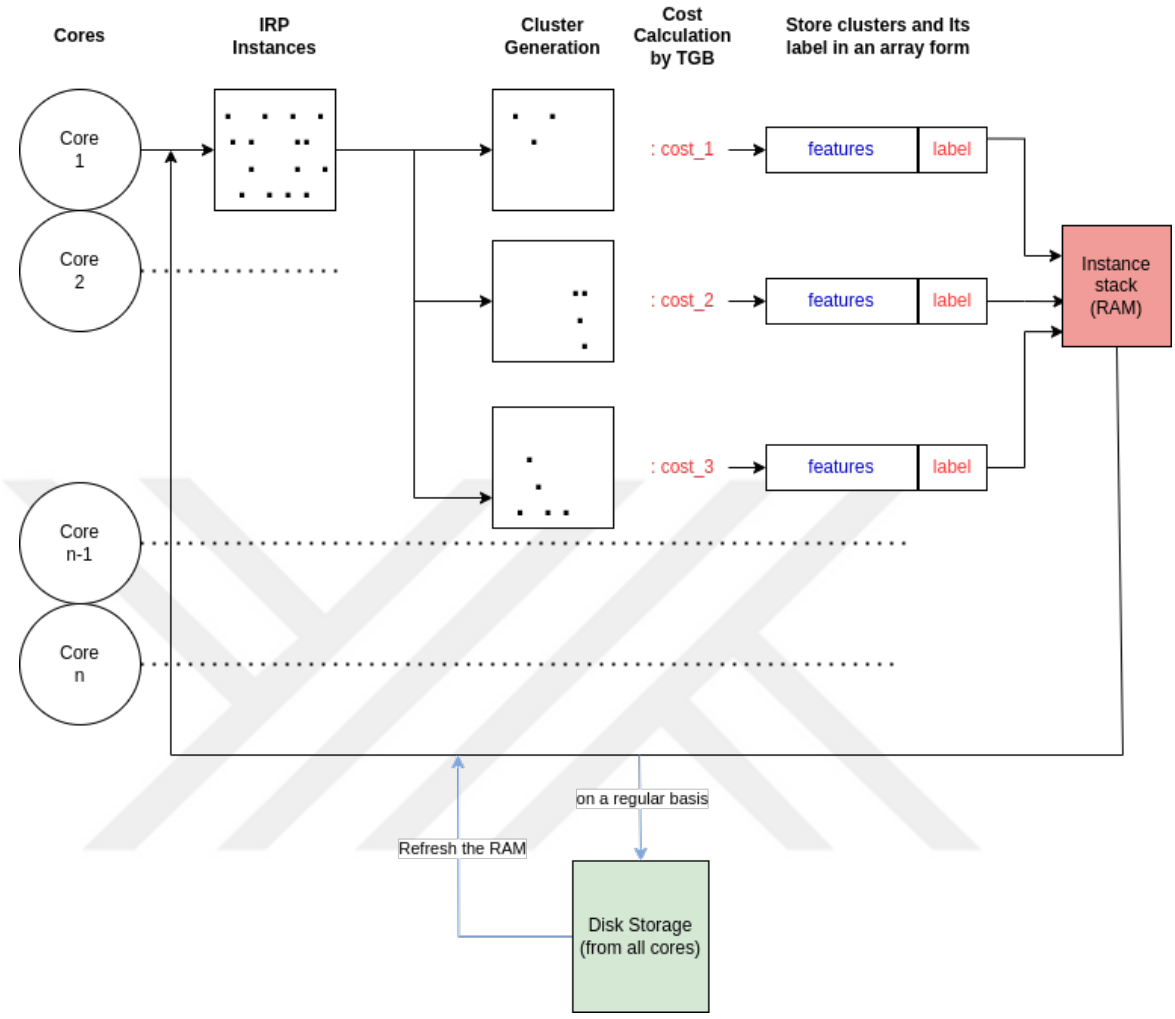

Figure 2: Multi-core massive data generation process

that require the most computational effort. The costs of the clusters are calculated using the proposed mathematical model. However, due to the high computational effort involved in searching for costs over full periods, we assume that solving the 2-period case of the problem will be representative of the full-period case. This assumption helps reduce the required computation time to a reasonable duration. We follow these approaches and generate approximately 700,000 data points. The mathematical model is designed using the CPLEX Optimizer, utilizing only a single thread. Subsequently, multi-core processes are developed to create separate instances for each core.

4.2 Feature Engineering

For each generated instance, we store the coordinates, storage capacities, and demands of customers, as well as the truck capacity and cluster size. Subsequently, we manipulate the instance data to prepare it for use in the neural network model. Given that our objective is to solve a combinatorial problem, the selection and extraction of relevant features are crucial for our framework. These features serve as representations of our instances to estimate the objective. As a result, we extract two sub-categories of features. The first category encompasses distance-related features, which primarily relate to the locations of nodes like the TSP. In the second category, we introduce new features to the literature that are specifically related to the IRP.

4.2.1 Distance related features

Distance-related features are derived from the coordinate information of the instances. These features are specifically designed to estimate the problems with distance-related objectives, such as the TSP. In our framework, we include similar features since our main objective is also to minimize transportation costs. By incorporating these distance-related features, we can capture the spatial relationships and distances between nodes, which are crucial in optimizing the routing and minimizing transportation expenses.

Cluster Window Lengths: The lengths of the x-axis and y-axis are calculated and included in the input vector. The rectangular structure formed by the nodes is of significance as it provides insights into the potential types of routes that the truck may follow. For example, if the customers are situated along a thin and elongated plane, likely, the truck can efficiently visit multiple customers by traversing a single line and returning, resulting in traveling distances that resemble the Manhattan distance. This information about the spatial arrangement of the nodes helps us gain intuition about the expected routing patterns and influences the optimization of the vehicle

routes.

Depot-Oriented Coordinate Information: We utilize the coordinate information of customers in our input vector, taking into account their relative positions according to the depot node. The depot node serves as the starting point for all vehicles and also serves as a refilling node. To ensure consistency and prevent unwanted effects on the deep learning model, we use absolute values for these distances rather than incorporating negative values. By using absolute distances, we maintain a consistent representation that aligns with the desired behavior of the model.

Nearest Neighbors: In addition, we determine the nearest neighbors of the customers by calculating the Euclidean distances between the nodes. Subsequently, we incorporate the distances to the closest three nodes for each node in our model. This additional information proves valuable in enhancing our model as it provides insights into the suitability of serving two or more customers together on a single route. By considering the distances to the closest neighboring nodes, the model gains a better understanding of the proximity and potential cohesiveness of customer clusters, thereby enabling more effective objective estimation.

4.2.2 IRP-related Features

The introduction of IRP-related machine learning features represents a significant contribution to our study. These features play a vital role in enhancing the performance of our model. Moreover, their potential extends beyond our specific problem domain, as they can be applied to other scheduling problems with similar characteristics. By incorporating these IRP-related features, we not only improve the accuracy and effectiveness of our model but also pave the way for their utilization in various related scheduling and optimization contexts.

Cluster size: This feature is a general attribute that applies not only to general IRP scenarios but specifically to our problem setting. The size of the cluster plays

a crucial role in determining the associated costs. Additionally, to accommodate the fixed-size input requirement of the neural network model, we introduce a novel approach. We clone the depot node to augment the number of nodes, ensuring that it reaches an upper bound that is suitable for the developed neural network model. By adopting this approach, we ensure that our feature vector is appropriately sized and compatible with the neural network architecture.

Demand and truck capacity rate: We compute the ratios of the daily consumption of each customer to the truck's capacity. This calculation provides a lower bound estimation for the number of visits required to serve each customer adequately. By considering the demand and truck capacity rate, we gain insights into the minimum number of visits needed to meet the customer's requirements.

Demand and storage capacity rate: Another feature we calculate is the ratio of customer demand to their storage capacity. This rate is particularly relevant in our continuous-time setting, as it indicates the customer's ability to sustain without receiving deliveries. By considering the demand and storage capacity rate, we gain insights into the resilience and capacity of each customer to withstand periods without deliveries, providing valuable information for the neural network model.

Storage capacity and truck capacity rate: We also calculate the ratio of storage capacity to truck capacity for each customer. This feature provides valuable insights into how much of the storage capacity can be utilized by full truck delivery. When the storage capacity is lower than the truck capacity, this limitation becomes another crucial factor in our problem setting, further enhancing the relevance of this feature within our proposed model. By considering the storage capacity and truck capacity rate, we gain valuable insights into the capacity utilization and potential limitations of the system.

Lower bound on total number of routes: Another important feature we calculate is the ratio of the total demand within a cluster to the truck's capacity. This ratio

provides an estimate of the minimum number of routes that a truck needs to execute to meet the cluster's demand. Although this is a basic assumption, it offers a flexible lower bound that guides the planning and optimization process. Additionally, this feature is directly related to the label in our proposed mathematical model, further emphasizing its relevance and importance in accurately estimating the objective function.

$$\sum_{i \in \hat{V}} \frac{q_i}{Q}$$

Required time to satisfy demand: We also determine the required time for a customer to fulfill its demand. This feature serves as a flexible lower bound, considering the possibility of a non-integer number of visits. By utilizing the time distance to the related node, we calculate the estimated time required for a customer to have its demand satisfied. This feature provides valuable insights into how well a customer fits within the time window constraints of the problem. For example, if a customer requires more time than the allotted operating hours, it indicates that the corresponding cluster is likely to become infeasible. This feature enables our model to learn and adapt to the time constraints and aids in identifying clusters that can be efficiently served within the given time frame.

$$\frac{q_i}{Q} \frac{2d_{i0}}{speed}$$

Lower bound for the IRP: In our study, we incorporate a lower bound for the IRP based on the work of Song and Savelsbergh [10]. They proposed a *Pattern Selection Linear Programming* (PSLP) model to estimate this lower bound. We utilize this lower bound in our model as well. By leveraging their linearity assumptions, we can rapidly solve the linear programming model and obtain an effective lower bound for the IRP.

4.3 Model Development

As our objective is to rapidly estimate the cost of the IRP for clusters, we have devised a *multi-layer perceptron* MLP neural network model employing the generated features. The adoption of neural networks for this purpose is grounded in multiple reasons. Primarily, neural networks serve as swift estimators once an effective model has been established. As elucidated in the preceding chapter, our fundamental goal in creating a machine learning model is the expedited analysis of a substantial number of clusters. Neural networks utilize straightforward linear processes following model training, aligning seamlessly with our specific case.

"Furthermore, neural networks exhibit competence in handling non-linear relationships. Given that our problem pertains to the IRP, the intricate interplay between various features can exert a pronounced influence on the overall objective. To illustrate, if the storage capacity is disproportionately high compared to the truck's capacity, the outcome might involve full truck deliveries. Conversely, when this relationship is reversed, achieving full truck deliveries may not be feasible, potentially escalating the costs. The construction of a mathematical model encompassing numerous constraints would also underscore the challenge of quantifying the impact of these relationships on the objective value.

Neural networks operate with a multitude of parameters, granting them the capability to discern intricate relationships with the objective. It is worth noting that this trait is often viewed as a drawback due to the "black-box" nature of neural networks. Nevertheless, in our context, we exclusively employ this model during the clustering phase, and the inner workings of the black box are of no concern to us at this stage. Subsequently, the selected clusters proceed to the routing phase, where a comprehensive search for solutions, inclusive of all delivery and routing decisions, is conducted.

We have also harnessed several advantageous features of neural networks to align

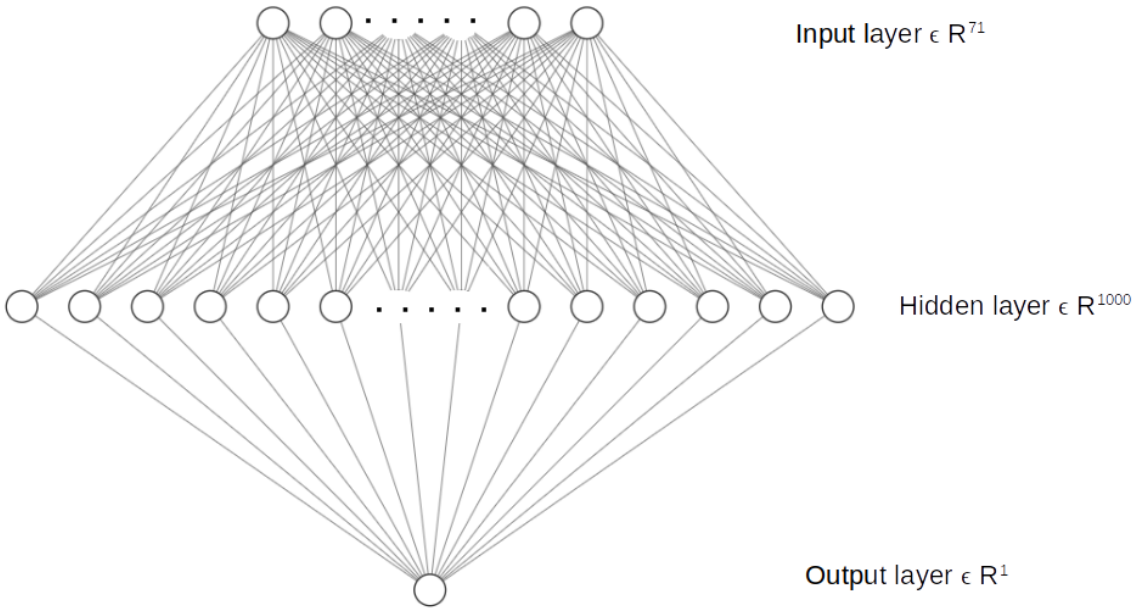

Figure 3: Multi-layer perceptron model visualization

with our case. One notable trait is that neural network performance is greatly enhanced when trained on a large dataset. To this end, we have introduced a novel data generation step, as explained earlier, allowing us to generate training data at a substantial scale. Additionally, neural networks seamlessly utilize GPUs, a necessity when dealing with a problem of this magnitude. Furthermore, these networks are versatile in handling both classification and regression tasks. In our case, we have adapted the same architecture for both a regression model and a classification model. The classification model serves to estimate the feasibility of a cluster, a role we will delve into in the subsequent sections.

The previous section outlined the features incorporated into our problem, culminating in a total of 71 features. Consequently, the MLP model proposed in this study operates with these 71 features. After numerous iterations, our model demonstrated optimal performance with a single hidden layer comprising 1000 neurons as shown in 3. In developing the MLP model, we harnessed the Glorot normal initialization

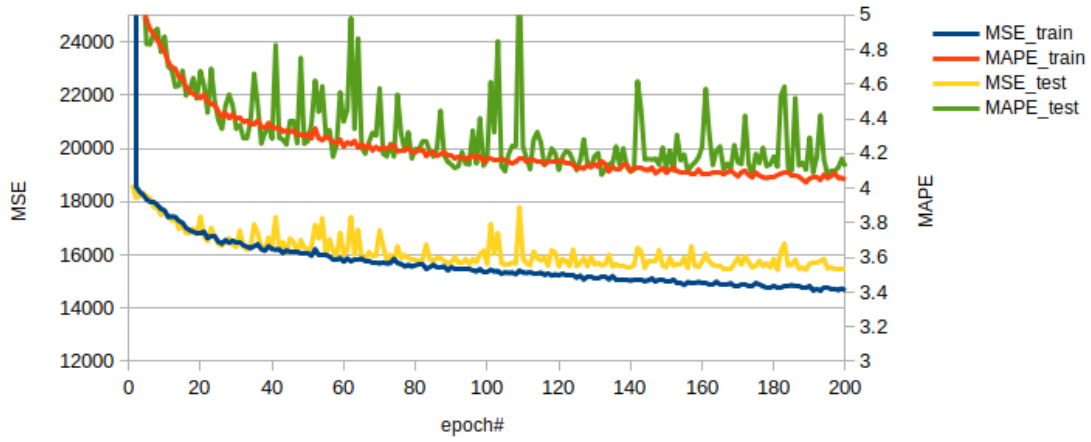

Figure 4: Loss during MLP training

technique to initialize the neural network's weights. This initialization method, recognized as Xavier initialization, adapts the initial weights according to the network's architecture to address challenges like vanishing and exploding gradients during the training process. Furthermore, it's worth noting that the rectifier activation function is employed in each layer of the model.

A crucial problem with the machine learning model is the distribution of objectives. A cluster with high consumption, small truck capacity, and more customers may result in a cost that is 20 times higher than the cost of serving a cluster with lower consumption, larger truck capacity, and fewer customers. This situation makes the learning process challenging, as costly clusters contribute to a much larger increase in the loss function compared to smaller ones. To address this issue, we trained the regression model using the *mean squared error* (MSE) but took checkpoints using the *mean absolute percentage error* (MAPE). This approach allowed us to learn a model with minimum error while also ensuring similar trends between labels from varying ranges.

Various parameters were tested to improve the MLP model. We figure out the training steps in figure 4 for 200 epochs. As a result, the regression model performed well with a single hidden layer, achieving a MAPE score of 4.12. Our classification

model shares a similar architecture and yields an accuracy score of 98.56%.

4.4 Framework for IRP

The previous steps provide us with the characteristics of a cluster in an array. These features are utilized to estimate the label, which is the IRP cost of the cluster found by the proposed tour generation-based heuristic mathematical model. A multi-layer perceptron framework has been developed for this purpose. We used these developed models to find the best partition of customers. We harnessed the developed models to determine the optimal customer partition. The utilization of these models is illustrated in Algorithm 3, which outlines the pseudo-code of the framework.

Algorithm 3 *Machine Learning Framework*

Input: Candidate clusters (K)
Output: Cost estimations for candidate clusters ($E = e_k \quad \forall k \in K$)
n: number of customers
N: number of clusters to be created for each size
threshold: to evaluate infeasibility scores
$K_{small} \leftarrow$ *clusters with less than 3 customers*
$K_{big} \leftarrow$ *clusters that has 3 or more customers*
$K_{filtered} \leftarrow$ *candidate clusters that is predicted as feasible*
for $k \in K_{small}$ **do**
  $e_k \leftarrow TGBH(k)$
end for
$featureVectors \leftarrow featureExtraction(K)$
$infeasibilityRate \leftarrow classificationModel(featureVectors)$
for $k \in K_{big}$ **do**
  if $infeasibilityRate_k > threshold$ **then**
    $delete(k)$
  end if
end for
$e_k = regressionModel(featureVectors) \quad \forall k \in K_{filtered}$
return $E = e_k \quad \forall k \in K_{small} + K_{filtered}$

One problem in estimating the cost of clusters by a machine learning model is that clusters can be infeasible. The infeasibility of a cluster means that the proposed mathematical model could not find any solution within the given time limit. Consequently, there is most probably no way to serve this subset of customers together.

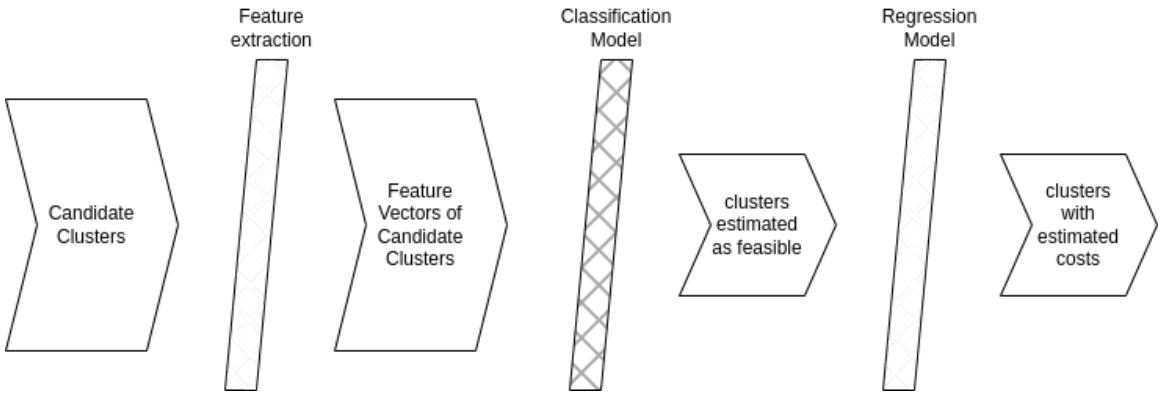

Figure 5: Machine learning framework used in process of clustering

To address this issue, we developed two different machine learning models. The first model is a classification model that predicts whether a cluster is feasible or not. In Figure 5, we illustrate the machine learning framework. In the figure, We also highlight the role of the classification model as a filter. The clusters that successfully pass through this filter, indicating their feasibility as estimated by the classification model, proceed to be evaluated in the subsequent steps of the process. This filtering mechanism allows us to focus our attention and resources on the clusters that are deemed feasible, enhancing the efficiency and effectiveness of the overall solution approach.

Subsequently, the cost of a cluster is estimated by the regression model only if it is classified as feasible by the classification model. As a result, the regression model is trained using data that contains only feasible clusters. This approach offers two advantages. Firstly, the classification model eliminates infeasible clusters, preventing the main problem from becoming infeasible. Secondly, training the regression model exclusively with feasible clusters provides that labeling for infeasible clusters can be ignored. Otherwise, the model would need to guess a non-existing objective, leading to the learning of unexpected patterns. We additionally employ the regression model, restricting its application to clusters with sizes ranging from 3 to 5. In cases where the cluster size is less than 3, the TGBH algorithm can swiftly compute the objective in under a second, particularly when dealing with 2 periods. Consequently,

the contribution of the regression model is somewhat constrained for these smaller clusters. This distinction in cluster treatment is driven by computational efficiency. The cost estimation for clusters comprising more than 2 customers is undertaken through the regression model, while others are directly solved by TGBH.

CHAPTER V

COMPUTATIONAL STUDY

In this chapter, we first present the instance structure used for the benchmark. Following that, we provide a brief description of the benchmark algorithm and explain why it was chosen as the benchmark instance. Finally, we share the computational results and demonstrate how our TGBH model and machine learning clustering method outperform the benchmark algorithm.

5.1 Instance Structure and Computational Resources

We test the new clustering and routing algorithms on the instances from the literature and on real-life instances[1]. The same instances are used since the proposed algorithm already works with quite low optimality gap values. here are 50 customers in the randomly generated instances used by [1]. The customer locations and the depot are randomly located on a map of size 1000×1000. Travel cost between two locations is equal to the Euclidean distance between them. Travel time is calculated assuming a vehicle speed of 10 distance units per minute. The daily operating hours are assumed to be 10 hours.

Table 1 provides an overview of the customer and truck features. The b value represents the base value utilized in this study. During the initial customer creation step, the storage capacity is randomly designated as either "low" or "high". The "min" and "max" values in the table define the respective range of values. For instance, if a customer is assigned a "low" storage capacity, its storage range extends from $0.75 \times b$ to $1.25 \times b$, equating to 750 and 1250 in our specific context. Moreover, a customer's demand category is determined based on their storage capacity, classified as either "*low*" or "*high*". For instance, if a customer's storage capacity is randomly

set to 1100 and categorized as having high demand, its demand will be randomly selected within the range of 1100×0.45 to 1100×0.65, resulting in values of 495 to 715 respectively. Additionally, a customer's initial inventory is assumed to be equal to its daily consumption rate. Each instance is generated with both "low" and "high" truck capacity options.

Five such instances are generated. For each of these instances, (i) relaxed, (ii) normal, (iii) restricted feasibility and (iv) lower consumption rate copies are created. For each instance, the average number of single customer delivery trips that can be made in a day by a single vehicle is estimated as $nK/\sum_{i \in V} 2t_{0i}$. Then, the average delivery amount, *base delivery capacity value*, when each vehicle performs a single direct delivery trip is estimated by dividing $\sum_{i \in V} q_i$ by the average number of single customer delivery trips. For relaxed, normal and restricted feasibility instances, the total capacity of the vehicles is set to 3, 2.5 and 2 times this base delivery capacity value, and the number of vehicles is determined accordingly.

The algorithms proposed in this study are implemented using Python programming language and CPLEX Concert Technology. On the other hand, the same codes are used to run CCLP and CH algorithms[1] that are implemented on C++ and Cplex Concert Technology. The computational experiments are carried out on a 64-bit Ubuntu 20.04 operating system with 32 Core AMD Threadripper 3970X CPU and 64 GB RAM. We report both the average optimality gap and the average solution time for each algorithm.

Table 1: Instance structure

b=1000	**low**		**high**	
	min	**max**	**min**	**max**
storage (s)	*0.75b*	*1.25b*	*1.75b*	*2.25b*
daily demand	*0.15s*	*0.35s*	*0.45s*	*0.65s*
truck capacity	*0.5b*	*0.5b*	*1b*	*1b*

5.2 Benchmark Algorithm

We used the same benchmark algorithm that Ekici et al.[1] used. They utilized *capacitated concentrator location problem (CCLP)* proposed by Bramel and Simchi-Levi[9]. Then they constructed the delivery routes for each cluster via proposed heuristics called *constructive heuristic (CH)*. We also use the clustering procedure proposed by Bramel and Simchi-Levi[9] to partition each of the randomly generated subsets. When solving CCLP for each subset, we limit the number of concentrators that can be located by M since we have M delivery vehicles available. Moreover, we restrict the size of each partition by five because our proposed clustering algorithm works up to five nodes. When solving the CCLP for each generated subset, we assign a fixed setup cost to each customer location and a cost for connecting/assigning each customer location to any other customer location. Then, we choose M customer locations as concentrator locations with the objective of minimizing total fixed setup and connection costs. In addition to the base model[9], we limit the number of customer locations assigned/connected to each concentrator location.

We use $\widehat{V}$ to denote the set of customers in a subset randomly generated as explained above. The fixed setup cost of locating a concentrator at customer location j is represented by v_j, and the cost of connecting customer location i to a concentrator at customer location j is represented by f_{ij}. Finally, L denotes the upper bound (seven in our case) on the number of customer locations assigned to a concentrator location. We estimate v_j and f_{ij} values using the *seed tours heuristic* proposed by [9]. Fixed setup cost v_j is estimated assuming a full truckload direct delivery to customer location j:

$$v_j = \frac{2c_{0j}}{Q} q_j \tag{26}$$

Connection cost f_{ij} is estimated based on the length of the tour starting from the

depot and visiting customer locations i and j:

$$f_{ij} = \frac{c_{0i} + c_{ij} + c_{j0}}{Q}(q_i + q_j) - v_j \tag{27}$$

In CCLP, two types of decisions variables are used:

$$u_{ij} = \begin{cases} 1, & \text{if customer location } i \text{ is assigned to customer location } j \\ 0, & \text{otherwise.} \end{cases} \qquad \forall i, j \in \widehat{V}$$

$$y_j = \begin{cases} 1, & \text{if a concentrator is located at customer location } j \\ 0, & \text{otherwise.} \end{cases} \qquad \forall j \in \widehat{V}$$

CCLP is formulated as follows:

$$\text{Min} \quad \sum_{i \in \widehat{V}} \sum_{j \in \widehat{V}} f_{ij} u_{ij} + \sum_{j \in \widehat{V}} v_j y_j \tag{28}$$

$$\text{s.t.} \quad \sum_{j \in \widehat{V}} u_{ij} = 1 \qquad \forall i \in \widehat{V} \tag{29}$$

$$\sum_{i \in \widehat{V}} u_{ij} - L y_j \leq 0 \qquad \forall j \in \widehat{V} \tag{30}$$

$$\sum_{j \in \widehat{V}} y_j \leq M \tag{31}$$

$$\sum_{i \in \widehat{V}} x_{ij} \leq 5 \qquad \forall j \in \widehat{V} \tag{32}$$

$$v_j + \sum_{i \in \widehat{V}} x_{ij} f_{ij} \leq L \qquad \forall j \in \widehat{V} \tag{33}$$

$$u_{ij} \in \{0, 1\} \qquad \forall i, j \in \widehat{V} \tag{34}$$

$$y_j \in \{0, 1\} \qquad \forall j \in \widehat{V} \tag{35}$$

The objective function (28) minimizes the total fixed setup and connection costs. Constraints (29) guarantee that each customer location in the subset is assigned to a concentrator location. The number of customer locations that can be connected to a concentrator is limited by Constraints (30). These constraints also force opening a concentrator at a customer location if a customer location is assigned to it. Constraint (31) limits the number of concentrators that can be located by the number

of vehicles. Constraints (34-35) are the binary restrictions. After solving this integer programming formulation, the concentrator location and the customer locations assigned to it form a cluster of the generated subset. In total, we have a partition of the generated subset with at most M clusters.

5.3 Results and Comparison

Ekici et al.[1] compared the results with a lower bound calculated by the following formula proposed by Song and Savelsbergh[10].

$$\sum_{i \in V} \frac{T q_i}{Q} 2 c_{i0}$$

This is a weak lower bound since it assumes direct full truckload deliveries to the customers and ignores the storage capacity of the customers. In other words, we relax the fleet size limitation by allowing a single vehicle dedicated to each customer and allowing any delivery amount ignoring the storage capacity restriction at the customer locations.

Consider an example where a customer's daily demand is 7 and the number of periods is 2, resulting in a total demand of 14 units over the specified period. To satisfy this demand, a total of 14 units must be delivered to the customer. With the knowledge of the truck capacity, we can then determine the minimum number of trips required for this customer. If the truck capacity is assumed to be 5, approximately 2.8 trips would be needed to fulfill the customer's demand.

Furthermore, if the round-trip cost to this customer from the depot is given as 100, we can readily calculate that a minimum expenditure of at least 280 units is necessary to fulfill the demands of the customer. By aggregating these values for each customer, we can derive the lower bound as proposed by Song and Savelsbergh[10]. After that, we use this value to find the gap between the objective we found. For example, if the lower bound 4500 and we found a solution with objective of 5000, then its gap is

calculated as $\frac{5000-4500}{5000}$ which is 10%

We present the average optimality gap of the solutions obtained by each implementation in Table 2. The first two columns of Table 2 indicate the clustering and routing procedures tested. In this study, we propose a *machine learning-based clustering algorithm* (MLB) and TGBH algorithm for routing. The gaps reported in the table are calculated only for feasible results. It is worth noting that Ekici et al.[1] also reported the CCLP algorithm, which can create clusters with up to 7 customers. In our comparison, we limited the cluster size to 5 to ensure a fair assessment. The results demonstrate that our new tour generation-based model outperforms the CH algorithm. Additionally, while the objectives achieved by the two clustering algorithms are at similar levels, the machine learning-based clustering method significantly contributes to preventing infeasibility, making it a favorable choice over the CCLP method. Thus, the MLB-TGBH duo gives the best results among the presented methods.

Table 2: Summary of results

Routing	**Clustering**	**#Fea**	**Min gap**	**Max gap**	**Average gap**
CH	**CCLP**	13	1.17%	7.22%	3.32%
	MLB	18	1.82%	5.97%	3.82%
TGBH	**CCLP**	36	1.15%	5.25%	2.28%
	MLB	40	1.15%	5.90%	2.38%

We want to emphasize the contribution of both the clustering and routing algorithms. Firstly, when comparing the two clustering algorithms, TGBH outperforms CH in all instances where the clustering algorithms are the same. When using the CCLP clustering method, TGBH provides better solutions in 39 out of 40 instances, whereas both algorithms failed to find a feasible solution for the remaining CCLP instance. Similarly, for our proposed MLB clustering method, TGBH outperforms

CH in 39 out of the 40 instances. Although CH runs almost three times faster than TGBH, the results demonstrate that the additional time invested in TGBH is certainly worthwhile.

When comparing the two clustering algorithms, we observe that they yield similar objective values for the 40 instances. However, when the CH algorithm is used as the routing algorithm, the MLB method outperforms it in 11 instances, while the CCLP method outperforms it in 10 instances. Both methods result in infeasible solutions for the remaining instances. Conversely, when the routing algorithm is switched to TGBH, the MLB clustering method surpasses it in 19 instances. Although these results do not consistently favor our new MLB clustering approach in a one-to-one comparison, it does prevent certain solutions from becoming infeasible. Moreover, these results are highly promising for a relatively untried machine learning-based method in solving a combinatorial problem.

In Table 3, we classify the instances in terms of the truck capacity and compare the performances of the algorithms on different truck capacities. When the truck capacity is large, the size of the generated clusters decreases, and hence, the problem becomes relatively easy. Accordingly, we observe lower optimality gaps for the instances with a large truck capacity.

Table 3: Summary of results according with a large and small fleet

Routing	**Clustering**	**Large Fleet**		**Small Fleet**	
		#Fea	**GAP**	**#Fea**	**GAP**
CH	**CCLP**	4	1.33%	9	4.21%
	MLB	3	2.07%	15	4.17%
TGBH	**CCLP**	19	1.44%	17	3.21%
	MLB	20	1.50%	20	3.26%

In addition to solution quality, we evaluate the run time performance of each

algorithm as well. In Table 4, we present the average run time of each algorithm. Here, we only report the run time of the final routing stage. We observe that the run time of TGBH is around 150% higher than that of CH. This is mainly since TGBH solves a mixed-integer program for each cluster.

Table 4: CPU time (in seconds) for the routing phase

Routing	Clustering	CPU Time
CH	**CCLP**	759.97
	MLB	672.99
TGBH	**CCLP**	2036.09
	MLB	2188.43

CHAPTER VI

CONCLUSIONS

In this paper, we study a variant of the well-known *Inventory Routing Problem* (IRP). In the setting studied, a single supplier replenishes the inventories of multiple customers over a planning horizon using a fleet of capacitated homogeneous delivery vehicles. Customers consume a single product at a constant rate and have limited storage capacity. The supplier is interested in constructing a cyclic delivery schedule with minimum transportation costs such that customers do not have stock-outs. We assume that the customers consume the product at a given rate uniformly throughout the day. Since the demand is not realized at one point in time, the time of delivery to each customer becomes more critical compared to other variants of IRP where the demand is realized instantly.

We focus on fixed partition policies where the customer set is partitioned into clusters, and each cluster is served independently by a single vehicle. Although this makes the delivery schedule more practical and helps keep the problem tractable, it is not straightforward to come up with a good partition. In the proposed approach, we take the interdependence between clustering and routing decisions into account by using the machine learning model to estimate the cost of serving each cluster when forming clusters. We generate a large number of clusters as we can estimate their cost in seconds with our novel deep learning model. Then, we choose the best partition of the customer set among these generated clusters by solving a set partitioning problem. In this approach, we find the cost of serving a cluster using a novel integer programming-based approach called *Tour Generation-Based Heuristic*(TGBH). In TGBH, we pre-generate a set of "good" delivery routes and form the delivery schedule

for a cluster of customers by solving a mixed-integer model. In this model, we decide which routes to execute, when to execute and how much to deliver to each customer at the same time not to isolate the three main decisions that depend on each other.

Through an extensive computational study, we have observed that the new machine learning cluster generation procedure performs well compared to previously implemented clustering algorithms. This outcome is highly promising for future studies, as it suggests that further improvements in the machine learning component can bring the results even closer to the lower bound. Additionally, unlike the previous study by Ekici et al. [1], we have limited the number of customers that a truck can serve to five. This restriction presents a potential area for future research, where a well-designed machine learning model can be developed to estimate clusters with a higher number of customers. Consequently, the overall objective value is also improved.

The new *Tour Generation-Based Heuristic* performs well regardless of the clustering algorithm used to partition the customer set. Moreover, the proposed heuristic is quite versatile in the sense that additional restrictions such as (i) limiting the number of visits to a customer location in a given period and (ii) imposing time windows for the deliveries can be handled by modifying the mathematical model used for solving the single-vehicle case. Compared to related studies in the literature, we have significantly better results. Although the average run time of the proposed approach is around 100% higher than the one in [1], the proposed approach not only finds solutions with lower optimality gaps but also finds a feasible solution for an instance in which the algorithm by Ekici et al. [1] cannot solve.

APPENDIX A

COMPUTATIONAL RESULTS

Table 5: Instance-wise Objective Values

instance	Lower Bound	CCLP-CH	MLB-CH	CCLP-TGBH	MLB-TGBH
inputn50s61	27305.46	infeasible	28194.37	28323.26	28134.47
inputn50s62	54610.92	infeasible	infeasible	55354.20	55361.31
inputn50s63	27305.46	infeasible	28191.04	28323.26	28137.76
inputn50s64	54610.92	infeasible	infeasible	55354.20	55361.31
inputn50s65	27305.46	28197.66	28207.44	28125.84	28137.66
inputn50s66	54610.92	infeasible	infeasible	55354.20	55362.29
inputn50s67	41636.13	infeasible	43647.61	42448.52	42690.44
inputn50s68	83272.27	infeasible	infeasible	84502.47	84643.44
inputn50s69	41636.13	infeasible	43647.61	42459.66	42691.00
inputn50s70	83272.27	infeasible	infeasible	84502.47	84643.44
inputn50s71	41636.13	infeasible	43647.61	42460.19	42691.00
inputn50s72	83272.27	infeasible	infeasible	84502.47	84643.44
inputn50s73	32272.40	33225.13	33592.11	33172.15	33418.32
inputn50s74	64544.81	infeasible	infeasible	65416.85	65377.34
inputn50s75	32272.40	33225.13	33592.11	33172.12	33421.84
inputn50s76	64544.81	infeasible	infeasible	65416.85	65377.34
inputn50s77	32272.40	33574.13	33492.95	33419.60	33304.17
inputn50s78	64544.81	infeasible	infeasible	65416.85	65377.36
inputn50s79	28397.27	infeasible	infeasible	29191.04	29128.24
inputn50s80	56794.53	infeasible	infeasible	57612.54	57452.89
inputn50s81	28397.27	infeasible	infeasible	29212.45	29130.71
inputn50s82	56794.53	infeasible	infeasible	57612.54	57452.89
inputn50s83	28397.27	infeasible	infeasible	29164.54	29174.70
inputn50s84	56794.53	infeasible	infeasible	57612.54	57452.89
inputn50s85	32297.67	infeasible	infeasible	infeasible	33199.02
inputn50s86	64595.34	65358.65	infeasible	65348.69	65591.89
inputn50s87	32297.67	infeasible	infeasible	infeasible	33118.13
inputn50s88	64595.34	65358.65	infeasible	65348.69	65591.89
inputn50s89	32297.67	infeasible	33319.07	infeasible	33053.29
inputn50s90	64595.34	65358.65	infeasible	65348.69	65591.89
inputn50s121	13643.40	14406.64	14449.29	14399.09	14348.62
inputn50s122	27286.81	infeasible	27791.86	27910.66	27771.80
inputn50s123	20799.48	21647.06	21476.56	21626.43	21480.07
inputn50s124	41598.96	infeasible	infeasible	42265.59	42427.68
inputn50s125	16129.39	16948.76	17154.07	16937.52	17139.91
inputn50s126	32258.77	32859.30	32957.35	32840.73	32870.58
inputn50s127	14184.24	15287.94	14933.19	14799.42	14921.65
inputn50s128	28368.47	infeasible	infeasible	28857.54	28775.10
inputn50s129	16141.40	16782.09	16991.25	16749.89	16982.60
inputn50s130	32282.80	infeasible	33030.69	infeasible	33009.59
average		***34017.68***	***29350.90***	***44460.10***	***43361.00***

Table 6: Instance-wise Deviation From Lower Bound

instance	CCLP-CH	MLB-CH	CCLP-TGBH	MLB-TGBH
inputn50s61	-	3.15%	3.59%	2.95%
inputn50s62	-	-	1.34%	1.36%
inputn50s63	-	3.14%	3.59%	2.96%
inputn50s64	-	-	1.34%	1.36%
inputn50s65	3.16%	3.20%	2.92%	2.96%
inputn50s66	-	-	1.34%	1.36%
inputn50s67	-	4.61%	1.91%	2.47%
inputn50s68	-	-	1.46%	1.62%
inputn50s69	-	4.61%	1.94%	2.47%
inputn50s70	-	-	1.46%	1.62%
inputn50s71	-	4.61%	1.94%	2.47%
inputn50s72	-	-	1.46%	1.62%
inputn50s73	2.87%	3.93%	2.71%	3.43%
inputn50s74	-	-	1.33%	1.27%
inputn50s75	2.87%	3.93%	2.71%	3.44%
inputn50s76	-	-	1.33%	1.27%
inputn50s77	3.88%	3.64%	3.43%	3.10%
inputn50s78	-	-	1.33%	1.27%
inputn50s79	-	-	2.72%	2.51%
inputn50s80	-	-	1.42%	1.15%
inputn50s81	-	-	2.79%	2.52%
inputn50s82	-	-	1.42%	1.15%
inputn50s83	-	-	2.63%	2.66%
inputn50s84	-	-	1.42%	1.15%
inputn50s85	-	-	-	2.71%
inputn50s86	1.17%	-	1.15%	1.52%
inputn50s87	-	-	-	2.48%
inputn50s88	1.17%	-	1.15%	1.52%
inputn50s89	-	3.07%	-	2.29%
inputn50s90	1.17%	-	1.15%	1.52%
inputn50s121	5.30%	5.58%	5.25%	4.91%
inputn50s122	-	1.82%	2.24%	1.75%
inputn50s123	3.92%	3.15%	3.82%	3.17%
inputn50s124	-	-	1.58%	1.95%
inputn50s125	4.83%	5.97%	4.77%	5.90%
inputn50s126	1.83%	2.12%	1.77%	1.86%
inputn50s127	7.22%	5.02%	4.16%	4.94%
inputn50s128	-	-	1.69%	1.41%
inputn50s129	3.82%	5.00%	3.63%	4.95%
inputn50s130	-	2.26%	-	2.20%
average	***3.32%***	***3.82%***	***2.28%***	***2.38%***

Table 7: Instance-wise Routing Algorithm Times (in seconds)

instance	CCLP-CH	MLB-CH	CCLP-TGBH	MLB-TGBH
inputn50s61	993.70	920.52	3091.36	3365.56
inputn50s62	935.53	744.31	1170.80	1542.98
inputn50s63	993.96	921.04	3093.74	3366.06
inputn50s64	939.23	747.80	1171.53	1541.64
inputn50s65	1096.14	1024.93	3845.52	3843.43
inputn50s66	938.01	746.90	1168.00	1537.43
inputn50s67	674.60	756.93	3248.14	2839.05
inputn50s68	951.07	436.06	516.21	309.56
inputn50s69	675.72	758.42	3247.67	2836.99
inputn50s70	951.21	436.07	517.45	314.88
inputn50s71	675.29	758.20	3247.60	2837.19
inputn50s72	952.71	436.66	517.15	311.90
inputn50s73	915.68	759.95	3047.58	3351.61
inputn50s74	579.09	260.85	657.24	1477.95
inputn50s75	914.88	760.06	3047.20	3352.93
inputn50s76	580.17	260.81	657.90	1478.61
inputn50s77	883.82	806.94	3181.68	3726.97
inputn50s78	578.46	259.18	658.98	1475.06
inputn50s79	1019.96	991.16	2894.50	3228.59
inputn50s80	865.38	793.45	1108.38	2211.29
inputn50s81	1020.04	990.95	2893.23	3229.95
inputn50s82	866.83	793.30	1108.38	2211.86
inputn50s83	957.95	977.34	3921.02	3847.22
inputn50s84	866.47	793.12	1110.10	2212.75
inputn50s85	779.99	696.44	3196.90	2971.74
inputn50s86	614.05	851.68	1014.87	1491.33
inputn50s87	780.55	698.18	3195.08	2973.00
inputn50s88	613.18	851.60	1018.66	1494.95
inputn50s89	723.29	777.56	3593.13	3239.96
inputn50s90	613.38	851.55	1016.49	1489.15
inputn50s121	463.84	544.49	1947.56	1964.96
inputn50s122	588.41	636.79	1689.19	1617.58
inputn50s123	648.81	470.82	2141.59	1558.09
inputn50s124	533.87	398.52	2300.95	2088.86
inputn50s125	567.41	503.90	2051.60	1349.77
inputn50s126	539.56	513.97	1680.51	1460.78
inputn50s127	491.01	474.44	2234.52	2211.81
inputn50s128	586.88	633.57	1360.30	1658.49
inputn50s129	534.28	294.39	1446.55	1887.33
inputn50s130	494.55	587.01	2434.69	1628.14
average	***759.97***	***673.00***	***2036.10***	***2188.43***

REFERENCES

[1] A. Ekici, O. Ö. Özener, and G. Kuyzu, "Cyclic delivery schedules for an inventory routing problem," *Transportation Science*, vol. 49, no. 4, pp. 817–829, 2015.

[2] A. Campbell, L. Clarke, A. Kleywegt, and M. Savelsbergh, *Fleet Management and Logistics*, ch. The inventory routing problem, pp. 95–112. Kluwer Academic Publishers, 1998.

[3] A. Campbell and M. Savelsbergh, "A decomposition approach for the inventory-routing problem," *Transportation Science*, vol. 38, no. 4, pp. 488–502, 2004.

[4] A. Campbell and M. Savelsbergh, "Delivery volume optimization," *Transportation Science*, vol. 38, no. 2, pp. 210–223, 2004.

[5] F. Cornillier, F. Boctor, G. Laporte, and J. Renaud, "A heuristic for the multi-period petrol station replenishment problem," *European Journal of Operational Research*, vol. 191, no. 2, pp. 295–305, 2008.

[6] L. Coelho, J.-F. Cordeau, and G. Laporte, "Consistency in multi-vehicle inventory-routing," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 270–287, 2012.

[7] C. Groer, B. Golden, and E. Wasil, "The consistent vehicle routing problem," *Manufacturing & Service Operations Management*, vol. 11, no. 4, pp. 630–643, 2009.

[8] K. Smilowitz, M. Nowak, and T. Jiang, "Workforce management in periodic delivery operations," *Transportation Science*, vol. 47, no. 2, pp. 214–230, 2013.

[9] J. Bramel and D. Simchi-Levi, "A location based heuristic for general routing problems," *Operations Research*, vol. 43, no. 4, pp. 649–660, 1995.

[10] J.-H. Song and M. Savelsbergh, "Performance measurement for inventory routing," *Transportation Science*, vol. 41, no. 1, pp. 44–54, 2007.

[11] H. Larrain, L. Coelho, and A. Cataldo, "A variable MIP neighborhood descent algorithm for managing inventory and distribution of cash in automated teller machines," *Computers & Operations Research*, vol. 85, pp. 22–31, 2017.

[12] C. Archetti, N. Boland, and M. Speranza., "A matheuristic for the multivehicle inventory routing problem," *INFORMS Journal on Computing*, vol. 29, no. 3, pp. 377–387, 2017.

[13] C. Archetti, M. Christiansen, and M. Speranza, "Inventory routing with pick-ups and deliveries," *European Journal of Operational Research*, vol. 268, no. 1, pp. 314–324, 2018.

[14] W. Bell, L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, and P. Prutzman, "Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer," *Interfaces*, vol. 13, no. 6, pp. 4–23, 1983.

[15] L. Coelho, J.-F. Cordeau, and G. Laporte, "Thirty years of inventory routing," *Transportation Science*, vol. 48, no. 1, pp. 1–19, 2014.

[16] L. Bertazzi and M. Speranza, "Inventory routing problems: an introduction," *EURO Journal on Transportation and Logistics*, vol. 1, no. 4, pp. 307–326, 2012.

[17] L. Bertazzi and M. Speranza, "Inventory routing problems with multiple customers," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 255–275, 2013.

[18] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Lokketangen, "Industrial aspects and literature survey: combined inventory management and routing," *Computers & Operations Research*, vol. 37, no. 9, pp. 1515–1536, 2010.

[19] A. Diabat, T. Abdallah, and T. Le, "A hybrid tabu search based heuristic for the periodic distribution inventory problem with perishable goods," *Annals of Operations Research*, vol. 242, no. 2, pp. 373–398, 2016.

[20] W. Lefever, E.-H. Aghezzaf, and K. Hadj-Hamou, "A convex optimization approach for solving the single-vehicle cyclic inventory routing problem," *Computers & Operations Research*, vol. 72, pp. 97–106, 2016.

[21] C. Cheng, M. Qi, X. Wang, and Y. Zhang, "Multi-period inventory routing problem under carbon emission regulations," *International Journal of Production Economics*, vol. 182, pp. 263–275, 2016.

[22] M. Chitsaz, A. Divsalar, and P. Vansteenwegen, "A two-phase algorithm for the cyclic inventory routing problem," *European Journal of Operational Research*, vol. 254, no. 2, pp. 410–426, 2016.

[23] M. Zenker, S. Emde, and N. Boysen, "Cyclic inventory routing in a line-shaped network," *European Journal of Operational Research*, vol. 250, no. 1, pp. 164–178, 2016.

[24] C. Archetti, L. Bertazzi, A. Hertz, and M. Speranza, "A hybrid heuristic for an inventory routing problem," *INFORMS Journal on Computing*, vol. 24, no. 1, pp. 101–116, 2012.

[25] G. Desaulniers, J. Rakke, and L. Coelho, "A branch-price-and-cut algorithm for the inventory-routing problem," *Transportation Science*, vol. 50, no. 3, pp. 1060–1076, 2016.

[26] B. Raa and W. Dullaert, "Route and fleet design for cyclic inventory routing," *European Journal of Operational Research*, vol. 256, no. 2, pp. 404–411, 2017.

[27] S. Mirzaei and A. Seifi, "Considering lost sale in inventory routing problems for perishable goods," *Computers & Industrial Engineering*, vol. 87, pp. 213–227, 2015.

[28] Y. Adulyasak, J.-F. Cordeau, and R. Jans, "Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems," *INFORMS Journal on Computing*, vol. 26, no. 1, pp. 103–120, 2014.

[29] R. Nambirajan, A. Mendoza, S. Pazhani, T. Narendran, and K. Ganesh, "CARE: Heuristics for two-stage multi-product inventory routing problems with replenishments," *Computers & Industrial Engineering*, vol. 97, pp. 41–57, 2016.

[30] Q.-H. Zhao, S. Chen, and C.-X. Zang, "Model and algorithm for inventory/routing decision in a three-echelon logistics system," *European Journal of Operational Research*, vol. 191, no. 3, pp. 623–635, 2008.

[31] J. Li, F. Chu, and H. Chen, "A solution approach to the inventory routing problem in a three-level distribution system," *European Journal of Operational Research*, vol. 210, no. 3, pp. 736–744, 2011.

[32] D. Aksen, O. Kaya, F. Salman, and O. Tuncel, "An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem," *European Journal of Operational Research*, vol. 239, no. 2, pp. 413–426, 2014.

[33] O. Solyali and H. Sural, "A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem," *Transportation Science*, vol. 45, no. 3, pp. 335–345, 2011.

[34] L. Bertazzi, L. Chan, and M. Speranza, "Analysis of practical policies for a single link distribution system," *Naval Research Logistics*, vol. 54, no. 5, pp. 497–509, 2007.

[35] J.-F. Cordeau, D. Lagana, R. Musmanno, and F. Vocaturo, "A decomposition-based heuristic for the multiple-product inventory-routing problem," *Computers & Operations Research*, vol. 55, pp. 153–166, 2015.

[36] J. Jung and K. Mathur, "An efficient heuristic algorithm for a two-echelon joint inventory and routing problem," *Transportation Science*, vol. 41, no. 1, pp. 55–73, 2007.

[37] Y. Yu, C. Chu, H. Chen, and F. Chu, "Large scale stochastic inventory routing problems with split delivery and service level constraints," *Annals of Operations Research*, vol. 197, no. 1, pp. 135–158, 2012.

[38] B. Raa, "Fleet optimization for cyclic inventory routing problems," *International Journal of Production Economics*, vol. 160, pp. 172–181, 2015.

[39] T. Abdelmaguid, M. Dessouky, and F. Ordonez, "Heuristic approaches for the inventory-routing problem with backlogging," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1519–1534, 2009.

[40] K. Li, B. Chen, A. Sivakumar, and Y. Wu, "An inventory-routing problem with the objective of travel time minimization," *European Journal of Operational Research*, vol. 236, no. 3, pp. 936–945, 2014.

[41] C. Archetti, G. Desaulniers, and M. Speranza., "Minimizing the logistic ratio in the inventory routing problem," *EURO Journal on Transportation and Logistics*, vol. 6, no. 4, pp. 289–306, 2017.

[42] L. Coelho and G. Laporte, "The exact solution of several classes of inventory-routing problems," *Computers & Operations Research*, vol. 40, no. 2, pp. 558–565, 2013.

[43] C. Cheng, P. Yang, M. Qi, and L.-M. Rousseau, "Modeling a green inventory routing problem with a heterogeneous fleet," *Transportation Research Part E: Logistics and Transportation Review*, vol. 97, pp. 97–112, 2017.

[44] A. Javid and N. Azad, "Incorporating location, routing and inventory decisions in supply chain network design," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 5, pp. 582–597, 2010.

[45] A. Federgruen and P. Zipkin, "A combined vehicle routing and inventory allocation problem," *Operations Research*, vol. 32, no. 5, pp. 1019–1037, 1984.

[46] W. Qu, J. Bookbinder, and P. Iyogun, "An integrated inventory-transportation system with modified periodic policy for multiple products," *European Journal of Operational Research*, vol. 115, no. 2, pp. 254–269, 1999.

[47] P. Avella, M. Boccia, and L. Wolsey., "Single-period cutting planes for inventory routing problems," *Transportation Science*, vol. 52, no. 3, pp. 497–508, 2018.

[48] R. Van Anholt, L. Coelho, G. Laporte, and I. Vis, "An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines," *Transportation Science*, vol. 59, no. 3, pp. 1077–1091, 2016.

[49] L. Bertazzi, M. Savelsbergh, and M. Speranza, *The Vehicle Routing Problem: Latest Advances and New Challenges*, ch. Inventory routing, pp. 49–72. Springer, 2008.

[50] A. Campbell and M. Savelsbergh, "Efficient insertion heuristics for vehicle routing and scheduling problems," *Transportation Science*, vol. 38, no. 3, pp. 369–378, 2004.

[51] O. Ö. Özener, Ö. Ergun, and M. Savelsbergh, "Allocating cost of service to customers in inventory routing," *Operations Research*, vol. 61, no. 1, pp. 112–125, 2013.

VITA

Taha Huzeyfe Aktaş graduated from İstanbul Atatürk High Science School in 2012. Following his outstanding performance in the university placement exam that same year, he earned the opportunity to study Industrial Engineering at Boğaziçi University. He completed his bachelor's degree in 2018 and subsequently gained diverse business experience in the telecommunication and banking sectors. Later, under the supervision of Prof. O. Örsan Özener, he embarked on a Data Science master's degree, focusing on routing problems within the realm of machine learning disciplines. Additionally, he made valuable contributions to a project funded by the Scientific and Technological Research Council of Türkiye (TÜBİTAK), advised by Prof. Ali Ekici. Aside from his passion for data science, Taha is also interested in Turkish literature and holds the distinction of being a published poet, having written a poetry book at the age of ten, making him one of the youngest published poets.