CLUSTERING-BASED HEURISTICS FOR EFFICIENT METER READING PLANNING IN ELECTRICITY DISTRIBUTION NETWORKS

BURAK KARA

ÖZYEĞİN UNIVERSITY

AUGUST, 2025

CLUSTERING-BASED HEURISTICS FOR EFFICIENT METER READING PLANNING IN ELECTRICITY DISTRIBUTION NETWORKS

by
Burak Kara

Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in
Industrial Engineering

Advisor: Assoc. Prof. İhsan Yanıkoğlu

Graduate School of Science and Engineering
Özyeğin University
İstanbul

August, 2025

CLUSTERING-BASED HEURISTICS FOR EFFICIENT METER READING PLANNING IN ELECTRICITY DISTRIBUTION NETWORKS

Approved by:

Assoc. Prof. İhsan Yanıkoğlu, Advisor
Department of Industrial Engineering
Özyeğin University

Asst. Prof. Mehmet Önal
Department of Industrial Engineering
Özyeğin University

Asst. Prof. Tonguç Yavuz
Department of Industrial Engineering
Istanbul Bilgi University

Approval Date:

DECLARATION OF ORIGINALITY

I hereby declare that I am the sole author of this thesis and that this is the true copy of my thesis, including the final revisions, approved by my thesis committee. All data and information have been obtained, produced, and presented in accordance with the rules of research ethics and principles of academic honesty. As required by these rules, to the best of my knowledge I have acknowledged ideas, thoughts, and any copyrighted material in accordance with the standard referencing rules. I certify that any part of this thesis has not been submitted for a degree or diploma in another educational institution.

Burak Kara

ABSTRACT

In energy distribution systems, the measurement of customer consumption is an important task for utility companies. In Türkiye, electricity meters are periodically read as part of an operation that requires field workers to record the data manually. This meter reading planning operation constitutes a complex problem. This paper addresses the meter reading planning problem faced by an electricity distribution company responsible for several cities, including metropolitan areas. The problem is examined as a case study using real data. Due to national regulations, to ensure consistent billing, the company must read all customer meters on time. A large workforce is employed to ensure that all meters are read on a monthly basis. The objective is to minimize both the number of workers and the total distance traveled while completing the task within a limited time frame. Due to the NP-hard nature of the problem, a clustering based heuristic approach is developed to provide a sustainable and efficient plan. This solution uses clustering algorithms, local search algorithms, and mathematical optimization methods to generate efficient routes for workers. The performance and applicability of the algorithm are tested on real data from residential zones. The final output of the algorithm is intended to be integrated into future planning processes of the utility company, potentially leading to significant improvements and a reduction in the planning process time.

Keywords: Meter reading optimization, vehicle routing problem, clustering, workforce scheduling, heuristic algorithms

ÖZET

Enerji dağıtım sistemlerinde, müşteri tüketiminin ölçülmesi hizmet sağlayıcı şirketler için önemli bir görevdir. Türkiye'de, elektrik sayaçları belirli aralıklarla sahadaki çalışanlar tarafından manuel olarak okunmaktadır. Bu sayaç okuma planlama işlemi karmaşık bir problemi oluşturmaktadır. Bu makale, büyükşehirlerin de dahil olduğu birçok şehirden sorumlu bir elektrik dağıtım şirketinin karşılaştığı sayaç okuma planlama problemini ele almaktadır. Problem, gerçek veriler kullanılarak bir vaka çalışması olarak incelenmiştir. Ulusal düzenlemeler gereği, faturalamada tutarlılığı sağlamak için şirketin tüm müşteri sayaçlarını zamanında okuması gerekmektedir. Tüm sayaçların aylık olarak okunmasını sağlamak amacıyla geniş bir iş gücü istihdam edilmektedir. Amaç, sınırlı bir zaman dilimi içinde görevi tamamlayarak hem çalışan sayısını hem de toplam kat edilen mesafeyi en aza indirmektir. Problemin NP yapısı nedeniyle, sürdürülebilir ve verimli bir plan sağlamak amacıyla kümeleme tabanlı sezgisel bir yaklaşım geliştirilmiştir. Bu çözüm, çalışanlar için verimli güzergâhlar oluşturmak üzere kümeleme algoritmaları, yerel arama algoritmaları ve matematiksel optimizasyon yöntemlerini kullanmaktadır. Algoritmanın performansı ve uygulanabilirliği, yerleşim bölgelerine ait gerçek veriler üzerinde test edilmiştir. Algoritmanın nihai çıktısının, hizmet sağlayıcı şirketin gelecekteki planlama süreçlerine entegre edilmesi ve bu sayede önemli iyileştirmeler sağlanarak planlama süresinin azaltılması hedeflenmektedir.

Anahtar Kelimeler: Sayaç okuma optimizasyonu, araç rotalama problemi, kümeleme, iş gücü çizelgeleme, sezgisel algoritmalar

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

In energy distribution systems, accurate and timely meter readings are essential for fair customer billing. Traditionally, this task is performed manually by field workers who travel between buildings to read the electricity meters. Although technological advancements, such as radio frequency identification (RFID), wireless communication, and smart city systems [1, 2], many utility companies, particularly in Türkiye, continue to rely on manual meter reading. This is due to significant initial infrastructure costs and regulatory requirements that require consistent and periodic billing cycles. As a result, manual operations remain the dominant approach across the country. However, manual meter reading creates significant challenges, especially in densely populated metropolitan areas. Consider a metropolitan city, where millions of residents are spread across vast and diverse neighborhoods. In such a setting, although it looks straightforward, the task of reading meters becomes a complex logistical operation. Workers must be systematically assigned to buildings and workdays, often navigating congested streets, locating individual buildings, and managing a large and variable workload. Without a well-structured plan, workers might backtrack, revisit areas unnecessarily, or waste time on inefficient routes. These inefficiencies lead to increased operational costs, reduced productivity, and fatigue of workers. To address these challenges, this paper presents a planning methodology aiming to improve the efficiency of meter reading operations. The objective is to optimize worker routes and minimize the total number of person-days required, while ensuring the resulting plans are practical and applicable in the field. In practice, field workers favor routes composed of consecutive buildings within the same street or neighborhood, as such routes are easier to interpret and execute. Therefore, the proposed solution prioritizes both optimization and operational usability. The methodology is inspired by a real-world problem faced by an electricity distribution company operating in several cities in Türkiye. Field workers in the company are assigned daily building lists to visit

and read all relevant meters. The assignment of buildings, workers, and workdays must be determined in a way that increases daily productivity and reduces total labor requirements. The Gurobi Solver fails to find feasible routes for the challenging mathematical optimization model, while the number of buildings containing meters to read exceeds **300**. Therefore, it is not practical or viable to solve this problem with a commercial solver.

To address the complexity of the real-world problem that arises in a metropolitan city, a heuristic approach is proposed. The methodology begins by compiling comprehensive data for each district, covering residential and commercial buildings. Since the company already divides the city into administrative districts to manage planning on a scale, each district is treated as an independent subproblem. In the optimal solution, it is assumed that a field worker will not serve neighborhoods across multiple districts in a single day. Therefore, a data set is generated for each district. However, data sets for districts are still too large to obtain and handle realistic distance or travel time information using matrices. Therefore, the data for each district must also be broken down into smaller parts. After creating district based data sets, a method is applied using k-means clustering to organize the large amount of meters data into smaller manageable groups. After splitting data from each district into manageable parts, the travel time matrix for each data is obtained using some services, as bird flight distance can result in unrealistic or infeasible routes in real life, and route times are calculated using some routing solution techniques to have optimal or good routes. The heuristic algorithm then addresses each smaller data set individually, applying clustering techniques to group buildings into candidate worker routes. In this solution approach, each cluster represents a route to be completed in a single workday. The algorithm initially generates a wide variety of clusters using different clustering strategies. From these, a set covering model is used to select an initial subset of clusters that covers all meter locations, while minimizing the number of routes and the total route time, i.e. the total person-day clusters. Finally, a

local search algorithm iteratively refines the initial solution by exploring adjacent configurations. This iterative refinement process aims to improve operational efficiency by fine-tuning clustering and routing decisions. The local search approach continues until the solution converges to a low level of resource usage. In addition to reducing resource usage, the proposed algorithm also ensures that the generated plans are practical and applicable for field operations. The applicability of the resulting routes is evaluated through real-case experiments and detailed analysis.

The heuristic approach presented in this paper provides a framework that can be adapted and applied to similar operational challenges in other contexts. The effectiveness of this approach is evaluated through its application to real data from the districts for which the utility is responsible. Significant improvements in the meter reading process are achieved by implementing the proposed methodology. Increased efficiency and reduced operational costs are the most important results of the improvements. This work contributes to the broader field of operational optimization in utility management. The outcome of the work is offering insights and methodologies that can be applied in similar contexts.

The remainder of this paper is structured as follows. Section 2 discusses the outstanding and recent related works in the literature. Section 3 describes the details of the problem with a mathematical optimization model. Section 4 details the solution approach by discussing in two main parts constructive heuristics and improvement heuristics. In Section 5, numerical experiments and a case study are provided to understand the performance and applicability of the proposed algorithm.

2. LITERATURE REVIEW

Meter reading planning has long been studied as a specialized variant of the vehicle routing problem (VRP). One of the first formalizations of the problem was by [3], who approached meter reading using a version of the Chinese postman problem with tour length constraints. Later, [4] proposed one of the first practical optimization-based scheduling models for utility companies, focusing on employee allocation and assigning daily route assignments. As the operational realities of utility services became more complex, the formulation of the problem evolved accordingly. The introduction of the multi depot periodic VRP (MDPVRP) marked a significant advancement, enabling the planning of periodic preventive maintenance services with mobile teams across multiple depots [5]. These models captured the logistical challenges facing utility providers that must ensure recurring service visits while operating under various constraints.

To better support real time operations, integrated systems became essential. [6] developed an evolutionary decision support system that combines geographic information system (GIS) and enterprise resource planning (ERP) technologies to address distance constrained routing problems in public utilities. In parallel with this technological integration, the notion of workload balance gained prominence, as uneven route lengths and assignments over a billing cycle can hinder operational efficiency. Addressing this issue, [7] proposed the balanced billing cycle VRP to distribute workloads more equitably over time, improving both efficiency and fairness in route planning. In addition to balancing workload, consistency in service also emerged as a critical concern. The consistent VRP (ConVRP), introduced by [8], aims to assign the same field worker to the same customers over multiple periods. This strategy promotes familiarity and improves service quality, aligning well with the goals of utility companies that want reliable and repeatable operations.

Expanding beyond meter reading, broader operational planning models have been

developed to jointly manage various utility services. For example, [9] presented an integrated framework for planning water and gas network maintenance programs, reflecting the growing need for coordination across service domains. In parallel, [1] addressed the inherent uncertainty in RFID based meter reading by formulating the problem as a stochastic close enough VRP (CEVRP) on real urban street networks. Their approach combines data analytics, integer programming, and Bayesian statistical modeling to minimize missed meter reads while generating robust and feasible routes. The study is notable for its use of large scale real world data and its potential integration into practical decision support systems for utility management.

The related literature on arc routing problems has also offered valuable structural insights, as these problems share key characteristics with meter reading operations. Applications such as winter road maintenance and salt spreading have been studied under arc routing frameworks [10], while the scheduling challenges faced by postal services like the United States Postal Service have inspired practical route planning models that focus on local adaptations [11]. Moreover, the arc partitioning problem presented by [12], which aims to divide service arcs into balanced workloads, conceptually parallels the division of utility regions into manageable zones, an approach crucial for the fair distribution of meter reading responsibilities.

Given the computational difficulty in optimally solving real-world VRPs, metaheuristics have become the predominant approach in practical settings. In particular, evolutionary algorithms have proven to be effective in a wide range of VRP variants. For example, [13] introduced a simple yet powerful evolutionary algorithm for classical VRPs, while [14] extended this line of work to cover more complex settings such as multi-depot and periodic VRPs using a hybrid genetic algorithm. More recently, [15] introduced the Swap* neighborhood structure and published an open-source implementation tailored for capacitated VRPs. These contributions are not only incorporated into

the solution framework of the present study but also serve as benchmarking references to evaluate the performance of the proposed heuristics.

Despite this extensive body of work, few studies explicitly address large scale, real world meter reading problems under regulatory and operational constraints, particularly in the context of utility services and dense urban environments. Furthermore, there has been minimal focus on how the proposed methods can be practically applied, specifically regarding their suitability for daily execution by field workers. This study contributes to closing this gap by presenting a scalable, case based heuristic solution that integrates clustering, set covering, and local search to support real time planning for meter reading operations in a major city of Türkiye.

3. PROBLEM STATEMENT

The utility company is responsible for reading all customer electricity meters each month, a task that requires physical visits to every meter. This operation is constrained by a limited number of workers and a fixed number of workdays, with each workday capped at eight hours. Therefore, assigning meters to workers and throughout the workdays must be carefully planned to satisfy these resource limitations. In the real-world case that motivates this study, the utility company operates in several districts, managing millions of electricity meters spread over more than a million buildings, called *points*. The scale of this operation creates a substantial workload, as all meters must be read within a defined period that repeats in monthly cycles. To handle this, meters and buildings are systematically assigned to specific pairs of workday-worker, which together define the available workforce resources of the company.

Multiple customer meters may be located within the same building. The total number of meters in a building determines the service time required for that building. Each building is referred to as a *point* and the total number of meters in a building is called the *weight* of that point, which represents the cumulative workload for that building. In the current plan, rather than assigning individual meters to workers, points consisting of meters are assigned as visit locations for workers. When a worker is assigned to a point, the worker is responsible for reading all meters within that point during the assigned workday. At the start of each workday, workers are transported to locations near their assigned set of points. Each worker must visit all points assigned to the current workday. Each worker follows an open route, meaning that the route does not need to end at the starting point, i.e., tours are not complete. Instead, workers start their routes at a point in the set of points assigned to them and visit other points sequentially to read all the meters assigned to them.

The service time required to read meters at a point is called the *reading time* of that

point. The reading time of a point increases depending on the weight of that point, due to the fact that each customer meter requires time to be read. Besides, the reading time of a point includes a fixed time to enter and leave the building. The time a worker takes to travel between points is called the *travel time* of the worker. The challenge is to efficiently assign points and plan routes for workers to minimize the workforce, time spent, and the required effort while ensuring that all meters are read within the eight hours time limit. In addition, routes must be optimized to use the workforce efficiently. The total time consumption of each worker, caused by travel time and the total reading time of points assigned to the worker, must be within the workday time limit, which is eight hours.

Optimizing the assignment and routing strategy is critical to effectively addressing this problem. However, determining the optimal deployment combination for even a single district is highly complex and time consuming. The complexity of the problem increases exponentially when scaled up to cover all districts and the millions of meters within them. Therefore, this paper proposes a clustering-based heuristic approach to solve the problem, which aims to find a practical and efficient solution within a reasonable time frame, balancing efficiency and effectiveness. There are several key steps in the proposed heuristic method, which are constructive heuristics where the algorithm generates an initial promising solution and improvement heuristics where the algorithm proposes several methods to improve the initial solution to achieve a better solution.

The purpose of the proposed algorithm is to determine the minimum workforce required to complete the reading plan of the meter. Consequently, the objective is to minimize both the number of routes and their corresponding route times, as detailed in Section 3.1. In addition, the algorithm seeks to generate solutions that are applicable and explainable in real world settings. To achieve this, the routes must be aggregated, which means that the points within each route should be located along the same road segments or within the same zones as possible. To assess the quality of the proposed solutions in

terms of applicability, a set of evaluation metrics is defined, as described in Section 5.2.

3.1 Mathematical Model

Table 3.1: *Nomenclature.*

Sets	
I	Set of all points, $I = \{0, 1, 2, \ldots, d\}$.
W	Set of workdays, $W = \{0, 1, 2, \ldots\}$.
C	Set of generated clusters, $C = \{0, 1, 2, \ldots\}$.
Parameters	
d	Dummy depot point.
K	Number of workers.
T_{max}	Maximum allowed time for a route.
t_{ij}	Travel time between points i and j.
t_j^r	Meter reading time (including fixed entry time) at point j.
τ_{ij}	Total of reading time at point j and travel time between points i and j, i.e., $\tau_{ij} = t_{ij} + t_j^r$.
α	Unit travel time cost.
β	Fixed cost for a worker.
Q	A large number for sub-tour elimination constraints, $Q = T_{\text{max}} + \max_{i,j}(\tau_{ij})$.
Decision Variables	
$x_{ij} \in \{0, 1\}$	Binary variable, equals 1 if any vehicle travels from point i to point j, and 0 otherwise.
u_i	Continuous variable representing the visit completion time of point i, $t_i^r \leq u_i \leq T_{\text{max}}$.

$$\min \quad \alpha \sum_{i \in I \setminus \{d\}} \sum_{j \in I \setminus \{d\}} t_{ij} x_{ij} + \beta \sum_{j \in I \setminus \{d\}} x_{dj}$$

$$\text{s.t.} \quad \sum_{i \in I, i \neq j} x_{ij} = 1 \qquad \forall j \in I \setminus \{d\} \qquad (3.1)$$

$$\sum_{i \in I} x_{ij} = \sum_{i \in I} x_{ji} \qquad \forall j \in I \qquad (3.2)$$

$$\sum_{j \in I} x_{dj} \leq K|W| \qquad (3.3)$$

$$u_i - u_j + Q x_{ij} \leq Q - \tau_{ij} \qquad \forall i \in I \setminus \{d\}, \forall j \in I \setminus \{d\} \qquad (3.4)$$

$$u_i \leq T_{max} \qquad \forall i \in I \setminus \{d\} \qquad (3.5)$$

$$u_i \geq t_i^r \qquad \forall i \in I \setminus \{d\} \qquad (3.6)$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in I, \forall j \in I \qquad (3.7)$$

The aim of the problem is to minimize the total cost arising from worker operations. This total cost accounts for both the workforce utilized and the operational effort of the workers. In the model, the cost is represented by two components: the travel cost, which depends on the travel time (and is proportional to the distance covered by each worker), and a fixed cost, which is incurred each time a worker is assigned to a route for a workday. The relative importance of these two components is controlled by parameters α and β. Since the fixed cost has a higher weight, the primary objective is to minimize the number of workers. Thus, the overall effort and workforce usage are represented as a cost function that must be minimized.

Constraints (3.1) ensure that all points are visited once. Forces the total visit to a point j to be exactly one. In other words, if the total number of binary variables x represents the incoming arcs to that point increases, while ensuring that the total is exactly one.

Constraints (3.2) is the flow conservation constraint [16]. The total number of incoming arcs to a point must be equal to the total number of outgoing arcs from that point. The constraint (3.3) indicates the total workforce, which is the multiplication of the number of workers and the number of days of work for the plan. The left-hand side gives the total number of workers departed from the depot, whereas the right-hand side is the total available workforce for the plan. The nature of the model will try to minimize the total departures from the depot due to the fixed cost in the objective function. Constraints (3.4) is the modified sub-tour elimination constraint inspired by Miller, Tucker, and Zemlin [17]. The value Q has been chosen as the tightest value that ensures sub-tour elimination. The tightest value Q is the sum of the daily time limit and the maximum value of τ_{ij}. Besides, the daily time limit of workers is restricted by sub-tour elimination constraints, constraints (3.5), and (3.6). Due to the fact that all visit completion times for points are less than or equal to the daily time limit, the model ensures that each worker is working within the workday time limit which is an upper bound of u_i values. The lower bound of u_i values is needed to consider the reading time of the corresponding point. Constraints (3.7) are integrality constraints.

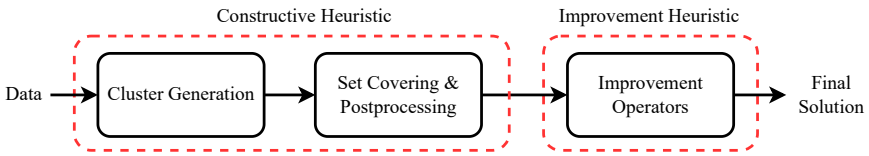
4. SOLUTION APPROACH

In the planning process, the customer meters located within the same building are aggregated. Each of these buildings is referred to as a point in the solution approach. A valid solution must ensure that all points are visited, which is essential to address the operational requirements of the problem. Therefore, the solution consists of a set of routes, each comprising a sequence of points. Together, these routes must cover all points on the network. Each route involves determining an efficient sequence to visit the assigned points. Assuming that each route is completed by a single worker within a single workday, it must satisfy the time constraint: the sum of travel time and meter reading time must not exceed the eight-hour limit. To generate an initial feasible solution, a clustering-based heuristic was developed, complemented by an improvement heuristic designed to enhance solution quality. These approaches are detailed in Sections 4.1 and 4.2.

The heuristic solution approach is applied to smaller subsets of the problem to obtain feasible solutions by using real travel-time matrices. These subsets are created by dividing all points that belong to the city into smaller parts. Initially, the data are naturally divided into segments based on district information. To further subdivide the problem, the bisection k-means approach is applied to each district segment. This approach starts by dividing the points into two clusters. Then, each large cluster is further split into two if its size exceeds a certain threshold. In this way, the data from the districts are divided into smaller parts, each containing fewer points than the threshold. As a result, the construction of a matrix for the data segments becomes more manageable. The constructive heuristic and improvement heuristic described in Sections 4.1 and 4.2 are then applied to each smaller manageable parts. Finally, a comprehensive solution is obtained by merging the output of all parts. The general flow of the algorithm is illustrated in Figure 4.1.

For each cluster, there exists a routing problem to determine the visiting order

Figure 4.1: *Flowchart of the general algorithm.*



of the points within that cluster. This routing problem is defined as an *open traveling salesperson problem (OTSP)* with a time constraint, due to the time limit of each vehicle and the open nature of the tours without a complete tour. To solve this problem, a classical TSP is first solved by introducing a dummy depot node. Then the dummy depot with zero distance to other points is removed from the solution. Finally, an open route is obtained and route time is calculated from the solution. The TSP is solved by the Lin-Kernighan heuristic method. At the end of the general algorithm, the routes are optimized to find the shortest travel time for the reading routes of the meters. Using the Lin-Kernighan heuristic for solving the TSP, the overall heuristic algorithm becomes faster. On the other hand, this speed gain option creates a trade-off between the route optimality and solution runtime. To determine distances and travel times between points, a distance matrix and a time matrix are generated for each part. These matrices provide real travel distances and travel times between points, avoiding the problems of using linear distances in geographically challenging city areas.

4.1 Clustering-Based Constructive Heuristic

To solve the problem, a practical approach is developed to generate clusters, each representing the route of a worker on a workday. These clusters consist of points, each point assigned to a single cluster. Each cluster corresponds to a specific worker–workday pair, forming a daily route plan. The route time of a cluster is defined as the total duration required to read the meters and travel between the points within it. A cluster is considered

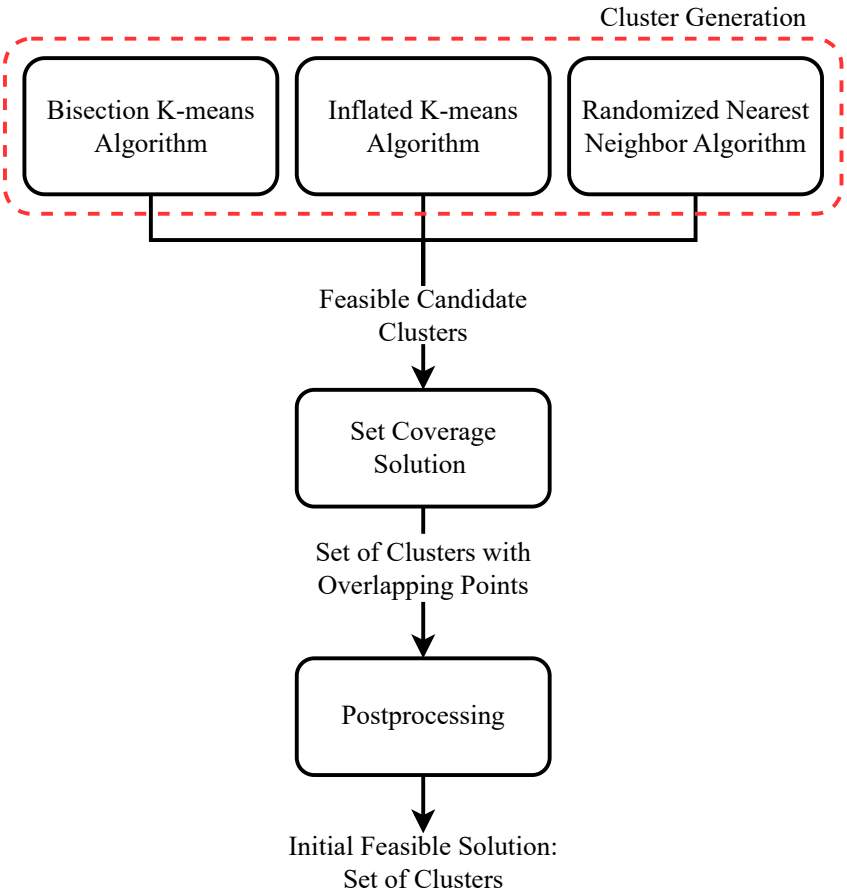
feasible if its route time does not exceed the eight-hour limit of a standard workday. Each worker–workday pair serves as a resource that can be assigned to a feasible cluster. As a result, minimizing the total number of clusters is essential to perform the meter reading plan efficiently, given the constraints of a limited workforce and available workdays.

A heuristic approach has been developed to generate this operational plan. This method involves creating a large set of feasible clusters through various clustering techniques outlined in Section 4.1.1. From this pool, a selection of clusters is chosen to cover all points. This selection process involves solving a set covering problem with a post-processing technique, as detailed in Section 4.1.2. The general flow of the constructive heuristic algorithm is illustrated in Figure 4.2.

4.1.1 Cluster Generation

The clustering approach aims to partition all points into clusters, each assigned to a worker for a specific workday. Once the points within a cluster are sequenced, the cluster forms a route that a worker follows during their shift. Each point may correspond to a building that contains multiple electricity meters, each meter requiring a certain amount of time to read. Routes are constructed by determining an efficient sequence of points within each cluster. The total travel time of a route depends on the length of the route and the characteristics of the road segments. These travel times are calculated using a predefined time matrix. The routing strategy assigns each route to a worker–workday pair, ensuring that the total duration, including both travel and meter reading times, respects the eight-hour workday constraint. To evaluate the feasibility of a route, the Lin-Kernighan heuristic is used to estimate the total route time. A route is considered feasible if the heuristic solution satisfies the time limit, as this heuristic typically provides an upper bound close to the optimal. After generating a diverse set of feasible clusters using various

Figure 4.2: *Flowchart of the constructive heuristic.*



techniques, a set covering method with a postprocessing step, detailed in Section 4.1.2, is applied to select a subset of clusters that covers all points while minimizing the total number of clusters. These feasible clusters that are generated by clustering methods, termed candidate clusters, are kept for use in the set covering stage. The methods used to generate feasible clusters are as follows:

Bisection k-means method. This clustering technique employs the standard k-means algorithm. By selecting a specific k-value, divide the points into k distinct clusters.

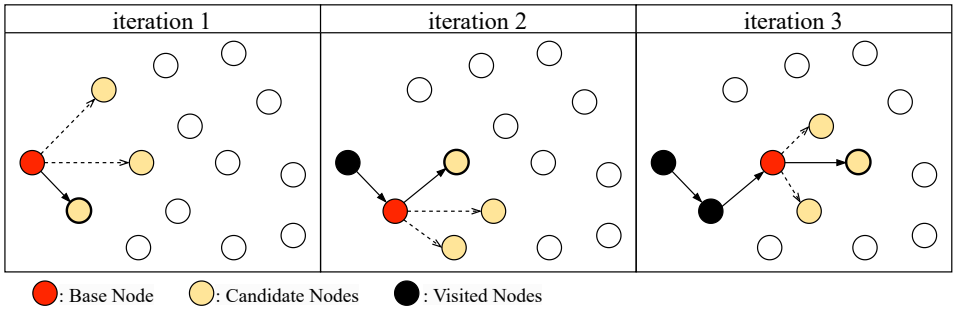
This k-value is derived using the problem-specific data. Since the overall reading durations, including initial entries of all points, are known, a method is developed to use this information to create a set of possible k values. The k-factor is determined by dividing the total reading times by the number of points. The set of possible k values is then composed of values obtained by multiplying the k-factor by certain coefficients. For instance, if the k-factor for data is 4, then the set of possible k values would be $\{2, 4, 6, 8\}$, representing 0.5, 1, 1.5, and 2 times the k-factor. This approach allows bisection clustering to begin with both relatively small and large clusters for dividing the points. The k-means algorithm operates with some k-value from the set. The result of the algorithm is a collection of clusters varying in size, some clusters feasible and others too large to be feasible. To determine the feasibility of the cluster, the time limit must be observed, which requires routing time calculations for each cluster. If the total meter reading time of a cluster falls within the time constraint, the route time is evaluated with the Lin-Kernighan heuristic. If a cluster exceeds the eight hour time limit, the cluster is bisected using k-means with k-value of two, and this partitioning is iterated until all clusters become feasible. After applying this method across each k-value yields a series of feasible clusters, which are then incorporated into the pool of candidate clusters created by the bisection k-means algorithm.

Inflated k-means method. After forming feasible clusters using the bisection k-means approach. Then, a copy of each candidate cluster undergoes processing using the inflated k-means method, which generates additional clusters from each. The centroid for each cluster derived from the bisection k-means is calculated. The points contained within the cluster are known as member points. The member point closest to the centroid is assigned as the center point. Non-member points, those outside the cluster, are ranked according to their distance to the center point, and the closest among them are sequentially added into the cluster. Each time a point joins the cluster, the closest member point to the

new point is noted as its predecessor. The new point is inserted after the predecessor, and the route time is increased accordingly. This method continues until no more points can be added. This systematic expansion allows the cluster to grow with new points while remaining feasible, and the expanded clusters are then included in the pool of candidate clusters.

Randomized nearest neighbor method. The algorithm starts with a base node serving as the starting point. Subsequently, a group of points nearest to this initial node is identified as potential subsequent points. From this group, one point is selected to become the next cluster member. This process is replicated using the newly incorporated point as the base node. Essentially, a randomly selected point near the last added point in the cluster is integrated into the cluster through successive iterations. This procedure persists until the cumulative route time of the cluster reaches the predefined limit. With this method, invoking an external routing algorithm is unnecessary, due to the inherent nature of the clustering approach. As points are progressively included, their order within the route is maintained. Examples of iterations for the randomized nearest neighbor algorithm are shown in Figure 4.3.

Figure 4.3: *Iterations of the randomized nearest neighbor (NN) algorithm.*



4.1.2 Set Covering with Postprocessing

Set Covering. The clustering methods described in Section 4.1.1 produce a candidate pool of feasible clusters for the initial solution construction. An appropriate initial solution aims to select a subset of these clusters to collectively cover all points. The set covering problem involves selecting the least number of clusters from the available candidates, ensuring that each point is included in at least one cluster. In this context, each cluster corresponds to a route assigned to a worker. Thus, reducing the number of clusters is directly related to minimizing the number of worker-workday pairs used.

A mathematical optimization model is used to solve the set covering problem. The objective is to minimize the number of selected clusters while ensuring that all points are covered, as enforced by constraints(4.1). Inequality constraints ensure that each point appears in at least one selected cluster. At this phase, we do not aim for a strict equality in the covering constraints (each point must appear in exactly one cluster), since this might lead to infeasibility due to points appearing in more than one cluster; these must be selected at the same time.

$$\min \sum_{c \in \mathcal{C}} \lambda_c$$

$$\text{s.t.} \sum_{c \in \mathcal{C}} a_{ci} \lambda_c \geq 1 \qquad \forall i \in \mathcal{I} \tag{4.1}$$

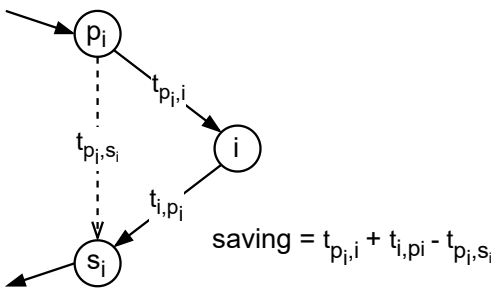
$$\lambda_c \in \{0, 1\} \qquad \forall c \in \mathcal{C} \tag{4.2}$$

In the model, the binary decision variable λ_c denotes whether the cluster c is selected from the set C. The a_{ci} parameter denotes whether the point $i \in \mathcal{I}$ is included $(= 1)$ or not $(\neq 1)$ in the cluster c. Solving the model with the covering constraint with inequality results in points included in multiple clusters. Ideally, each point should be assigned to exactly one cluster to form disjoint routes. If there are points included in multiple clusters, a postprocessing procedure is later applied to convert the solution into

a valid set partition. The resulting set of clusters guarantees complete coverage while eliminating multiple occurrences and minimizing the number of required clusters.

Postprocessing approach. The algorithm begins by identifying the points that appear in multiple clusters. For each such point, a removal saving is computed for every cluster to which it belongs. This savings represents the reduction in the route time achieved by removing the point from the cluster, as illustrated in Figure 4.4. In figure p_i and s_i denote the predecessor and successor of point i, respectively. The point is retained in the cluster where its removal would have the lowest saving and is removed from all the others. After resolving all the points included in multiple clusters, the route times are recalculated to reflect the new cluster compositions. This iterative postprocessing guarantees that the resulting solution assigns each point to a single cluster.

Figure 4.4: *Saving obtained by removing node i.*



4.2 Improvement Heuristics

After applying the clustering-based heuristic algorithm explained in Section 4.1, a feasible solution is obtained. To decrease the number of clusters obtained in the constructive heuristic, improvement heuristic operators are designed and applied. The operators of the improvement heuristic are as follows.

The first operator is detecting crossing clusters and eliminate them. Crossing clusters are identified by checking for intersections between their convex hulls. Once detected, these clusters are reclustered using a sweep clustering algorithm to eliminate crossings. Further details of this procedure are provided in Section 4.2.1. This improvement operator is initially applied after the constructive heuristic and is re-applied following the cluster merge and other improvement operators.

The second operator of the improvement heuristic is the merge algorithm. To reduce the number of resources required, a merge operator is designed to merge clusters. This operator aims to decrease the total number of clusters by merging them. Details of the operator are provided in Section 4.2.2.

The third operator of the improvement heuristic is the divide-and-merge algorithm. Following the merge operator, an additional cluster merge method called divide and merge is designed. This operator focuses on dividing clusters and merging the resulting fragments with other clusters to further reduce the total number of clusters. Details of the operator are provided in Section 4.2.3.

The fourth operator of the improvement heuristic is the outlier detection and reassignment method. To improve the assignment of nodes to clusters, a node transfer approach is designed. After applying the merge algorithms, outlier points are identified within each cluster. These outliers are then reassigned to nearby clusters using the KNN algorithm. More details are provided in Section 4.2.4.

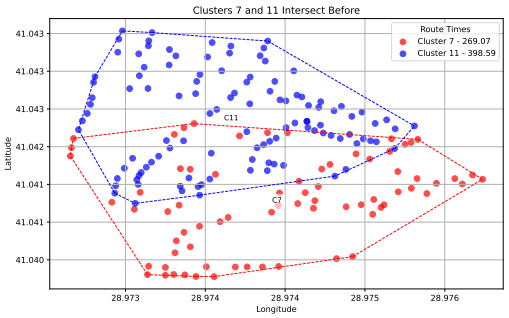
The fifth operator of the improvement heuristic is re-clustering via the sweep algorithm. The objective of this step is to achieve a better assignment, evaluated using a scoring mechanism. The score of a move is calculated based on the reduction in the number of clusters and the total route time, with a priority given to minimizing the number of clusters. Candidate moves are identified by applying the sweep algorithm to selected groups of clusters. Then a subset of these candidate moves is applied to improve the

solution. Details of this step are provided in Section 4.2.5.

The sixth and seventh operators of the improvement heuristics are the swap* and split algorithms. The general structure of these algorithms are provided by Vidal [15] and Prins [13]. The swap* aims to find a better assignment of points to clusters. The split tries to decrease the number of clusters in the solution. Details of this step are provided in Section 4.2.6.

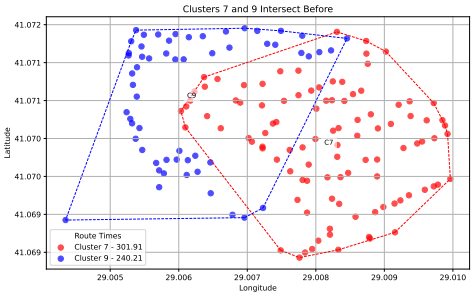
4.2.1 Crossing Clusters Detection and Elimination

Figure 4.5: *Illustrations of crossing clusters that occurred after the constructive heuristic and their resolution through re-clustering.*

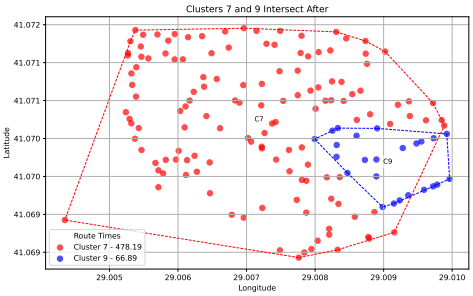


(a1) Before re-clustering

(a2) After re-clustering



(b1) Before re-clustering

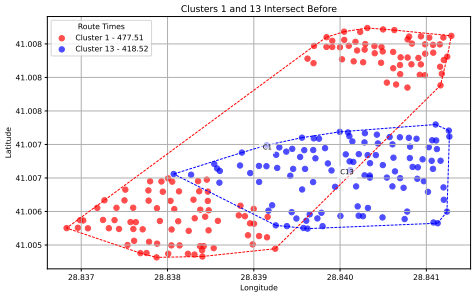


(b2) After re-clustering

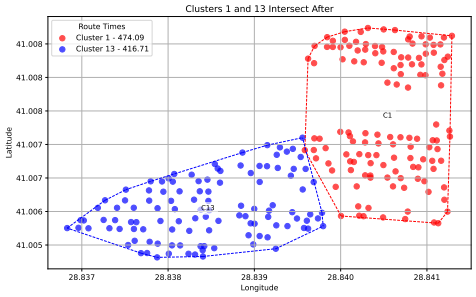
Crossing clusters are cluster pairs in which some points from one cluster may fall within the convex hull of another. Examples of such pairs of crossing clusters are shown in Figures 4.5 and 4.6 . The occurrence of crossing clusters can be attributed to two main

reasons: the constructive heuristic, which combines various clustering techniques and removes overlapping nodes; and the improvement heuristic, which includes the merging of clusters. The result of constructive heuristic, which includes the cluster generation and set covering processes, may lead to a set of clusters that intersect each other, as illustrated in Figure 4.5. Furthermore, additional improvement heuristics, such as merge algorithms, can also result in cluster crossing, as illustrated in Figure 4.6. Such structures are detected by a convex hull check and addressed by a reclustering method. This step is motivated by human intuition: In practical applications, crossing clusters are perceived as illogical and may lead practitioners to deviate from the proposed solution. Although this procedure does not necessarily improve the quality of the solution, it helps produce results that are more interpretable and acceptable in practice. Moreover, while it does not guarantee the complete elimination of all crossing cases, the reclustering process often generates at least one fully utilized cluster, as illustrated in Figure 4.5 (b), making the result still valuable for further improvement operators, such as cluster merge operators. In some cases, this step can also lead to an improved solution or a reduction in the number of clusters, as illustrated in Figures 4.7.

Figure 4.6: *Illustration of a crossing cluster that occurred after the merge operator and its resolution through re-clustering.*

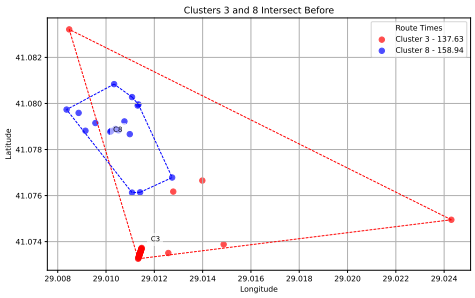


(a1) Before re-clustering

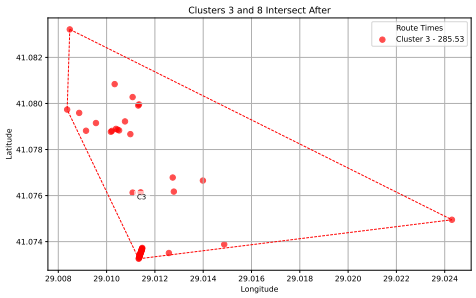


(a2) After re-clustering

Figure 4.7: *Illustration of a crossing cluster and its resolution through re-clustering, resulting in a reduction in the number of clusters.*



(a1) Before re-clustering



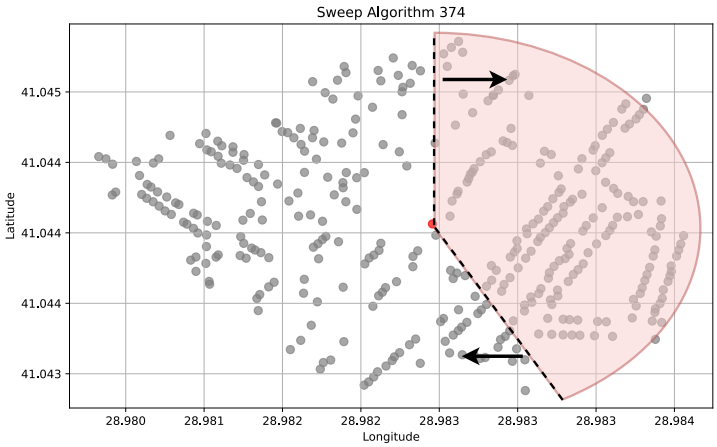
(a2) After re-clustering

To identify crossing cluster pairs, the intersection areas of the convex hulls of all cluster pairs are examined. If the intersection area between two clusters exceeds a certain ratio of the area of one of the clusters, the pair is classified as a crossing cluster pair. Graham's algorithm [18] provides a method to compute the convex hull of each cluster in polynomial time. The method sorts points by angle and constructs the convex hull using a stack-based approach. The convex hulls of the clusters form convex polygons in two-dimensional space. The intersection of two convex polygons results in another convex polygon. This intersection polygon can be computed using a linear-time algorithm that identifies the intersection points [19]. The intersection ratio between two convex polygons can then be determined by calculating the area of the intersection polygon using the Shoelace formula [20], also in linear time.

In practice, convex hulls are constructed using the SciPy package [21], which implements the Quickhull algorithm. For computing polygon intersections and their areas, the Shapely package [22], which contains the GEOS library, is used. These libraries implement efficient algorithms suitable for general polygon operations, and they offer reliable performance in real-world data. While the theoretical methods cited above offer good performance for specific polygon types ,e.g. convex, the chosen packages provide practical and scalable solutions with broader usage.

After detecting crossing cluster pairs, the points belonging to these pairs are re-clustered using the sweep clustering algorithm. The sweep algorithm begins at an initial angle and processes nodes sequentially by sweeping clockwise or counterclockwise, as illustrated in Figure 4.8. Each time a node is added to a cluster, the total reading time is updated. As the total read time approaches a predefined limit, the route time is also calculated by optimizing the node sequence. Once the cluster reaches the time limit, a new cluster is initiated starting from the last processed angle. This process continues until all nodes are assigned to clusters. The algorithm is applied with two different starting angles: one from the north, sweeping clockwise, and one from the east, sweeping counterclockwise. This dual-angle approach typically separates a crossing cluster pair into two clusters along vertical or horizontal axes. Between the two resulting clustering solutions, the one with the shorter total route time is selected as a candidate move.

Figure 4.8: *An iteration of the sweep algorithm.*



After determining all candidate moves for the current set of clusters, a score is assigned to each move. This score is calculated by considering both the number of resulting clusters and the total route time, with greater weight given to minimizing the number of clusters. The candidate moves are then sorted in ascending order on the basis of this score.

The move with the lowest score is selected and applied first. The clusters involved in the selected move are added to a tabu list to prevent their inclusion in subsequent moves. As the algorithm continues, the next best move is selected only if none of its clusters is present in the tabu list. This process continues until all candidate moves have been evaluated. Once all applicable moves are performed, the algorithm re-evaluates the candidate moves for the updated cluster set. The entire procedure is repeated until no further changes are observed. Then, the algorithm terminates.

4.2.2 Merge Operator

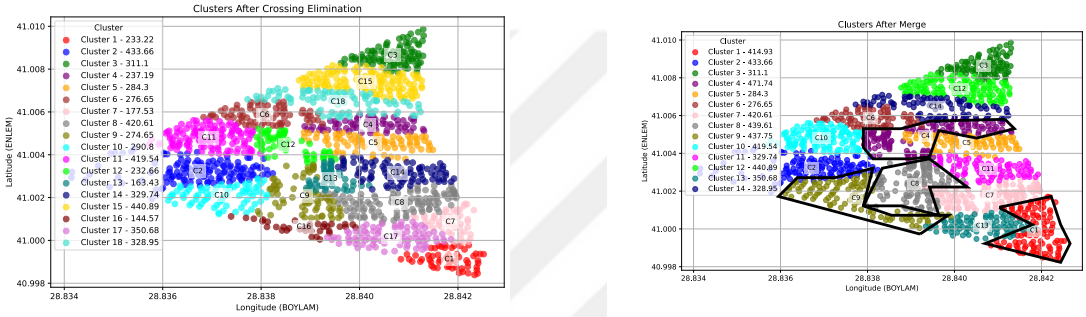
The cluster generation and set covering methods produce feasible clusters that cover all nodes. However, these methods may produce underutilized clusters in terms of time capacity. Some selected clusters could potentially be merged to achieve a more efficient solution, especially since resources such as workers and workdays are limited. In simpler terms, improvement operators can be applied to reduce the total number of clusters, thus reducing the total number of routes and the total required work resources.

A merge operation is developed as one of the improvement operators. The process starts by calculating the distances between the centroids of all cluster pairs in the current solution. These cluster pairs and their corresponding centroid distances are stored in a list called the merge pair list. Once the centroid distances have been calculated, the cluster pair with the shortest distance is selected and removed from the merge pair list to attempt a merge. The nodes of the selected clusters are combined and a new route is computed for the merged cluster using the Lin-Kernighan heuristic. If the route time of the merged cluster does not exceed the time limit, the merge is considered feasible. In that case, the original clusters are removed from the cluster set and the merged cluster is added. Additionally, all pairs that involve either of the original clusters are removed from the merge pair list. Then, new candidate pairs are generated by calculating the centroid

distances between the newly merged cluster and the remaining clusters in the cluster set, and these pairs are added to the merge pair list. The steps of the merge operator are provided in Algorithm 1

This process is repeated iteratively until no feasible merges remain in the merge pair list. A successful example of a merge iteration is illustrated in Figure 4.9.

Figure 4.9: *Successful execution of the merge operator. Cluster pairs (C1, C7), (C4, C12), (C9, C13), and (C10, C16) are merged to form new clusters.*



(a) Before merge operator

(b) After merge operator

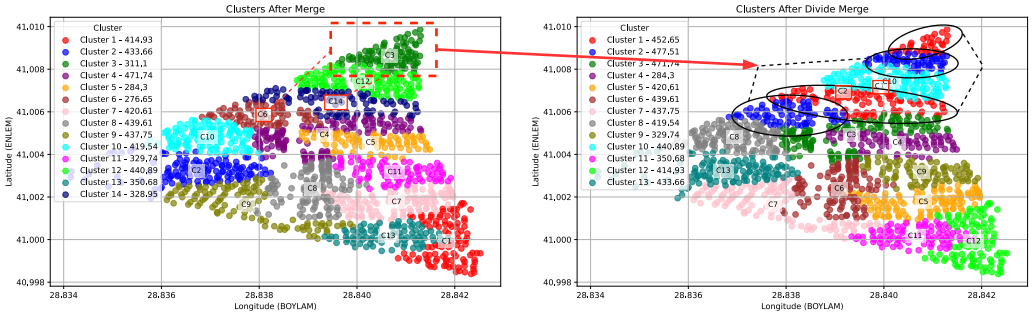
Algorithm 1 Merge Operator

```
Compute centroid distances for all cluster pairs and store in merge pair list
while merge pair list is not empty do
    Select the cluster pair with the shortest centroid distance
    Remove selected pair from the merge pair list
    Merge the nodes of the selected clusters into a merged cluster
    Solve TSP on the merged cluster using Lin-Kernighan heuristic
    if merged route time ≤ time limit then
        Remove original clusters from the cluster set
        Add merged cluster to the cluster set
        Remove all pairs involving either of the original clusters from the merge pair list
        Compute and add new centroid distance pairs between the merged and remaining clusters
    end if
end while
```

4.2.3 Divide and Merge Operator

After applying the merge operator, another cluster merge operator called divide and merge is applied to reduce the number of clusters. Divide-and-merge selects a cluster and divides it into two fragments using the k-means algorithm. Then it attempts to merge these divided fragments with other clusters in the solution. The algorithm iterates through each cluster, trying to split it into two. After each division, the merge operation is attempted on a fixed number of the nearest neighbors of these divided cluster fragments. For the merge operation, the clusters closer to the divided cluster among the neighbors are given priority first. The algorithm attempts to merge one of the divided cluster fragments first, and then the other. It is executed if a feasible merge operation is successful for one of the combinations; otherwise, the divided cluster fragments are attempted to merge in the reverse order. Suppose that none of the merge operations is feasible for any combination. In that case, the algorithm moves on to another cluster to attempt the divide-and-merge operator. An illustration of a successful divide and merge operator process is given in Figure 4.10

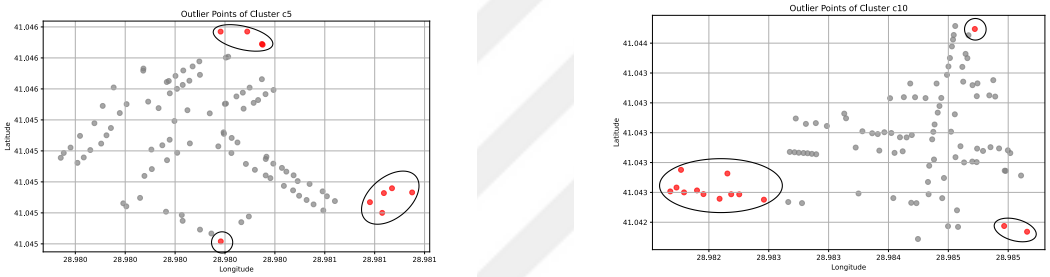
Figure 4.10: *A successful divide-and-merge operator iteration. Cluster C3 is divided into fragments, which are then merged with clusters C6 and C14. The resulting new clusters are C1 and C2.*



The divide-and-merge operator can be configured with parameters that allow clusters to be divided into three or more fragments to enable merging. However, in practical applications, the operator is used to divide the clusters into two fragments, as it has been observed that using the k-value of two in the k-means algorithm performs better for the case.

4.2.4 Outlier Detection and Reassignment Using K-Nearest Neighbors Algorithm

Figure 4.11: *Outlier points of two clusters.*



(a) Outlier points in cluster C5

(b) Outlier points in cluster C10

After applying constructive heuristic and improvement operators, some outlier points may occur for some of the clusters in the output. Outlier points are defined by their distance to the centroid of the cluster compared to the inertia value of that cluster, inspired by [23]. The inertia of each cluster is calculated by the sum of the squared distance between the centroid of the cluster and the points. The distance between each point and the centroid of the cluster is divided by the inertia value of that cluster for each point. This calculation gives a score which is a ratio value for each node. The highest values among these ratio scores indicate the possible outliers in the solution output. In other words, if a point is located in a separate place from the rest of the cluster, then that point is assumed to be an outlier. A certain portion of the points with the highest score

among all clusters are assumed as outlier candidates. Some outlier candidates belonging to two clusters are illustrated in Figure 4.11.

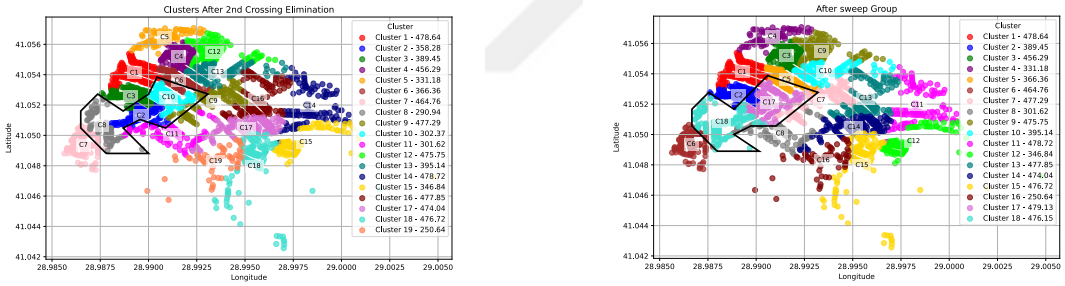
After detecting candidate outlier points, a k-nearest-neighbor algorithm is applied to these points. This algorithm identifies the k closest nodes to an outlier node as its neighbor nodes. The outlier points are then attempted to be added to the most frequent route among the routes assigned to the neighboring nodes. The operation is executed if adding the outlier point to the chosen route results in a feasible solution. All outlier nodes are sorted by their outlier ratio value and processed by the KNN algorithm in this order. After this operator, a better assignment and a shorter total time may be achieved.

4.2.5 Re-Clustering via Sweep Algorithm

The sweep algorithm applied to eliminate crossing clusters, explained in Section 4.2.1, is used to reduce the total number of clusters by applying a reconstructive method. The sweep clustering algorithm is applied to a group of clusters during this process. The goal is to decrease the number of clusters in that group by rearranging them using the sweep algorithm. Each of these groups consists of three clusters selected from the full set of clusters. To form groups, all possible triple combinations of clusters are generated, and their convex hull intersections are checked. If a convex hull intersection occurs between a pair of clusters, those clusters are considered neighbors. If there are two neighbor relations among the three clusters in a group, the group is identified as a neighbor group. The sweep clustering algorithm is applied to all neighboring groups. Sweep algorithm starts from different angles, as described in Section 4.2.1, and results in a reconstructed clustering output. If the output contains two or fewer clusters, the result is stored as a candidate move. In other words, each candidate move applies sweep clustering to a selected group of three clusters from the current set. Each move yields a different result that may affect the feasibility of other moves. After generating all possible candidate moves, a selection

process is performed to maximize the reduction in the number of clusters. This selection rule begins by calculating the frequency of each cluster, that is the number of times it appears across all moves. The total frequency of a move is defined as the sum of the frequencies of the clusters that it includes. The move with the lowest total frequency is selected first and the remaining moves are considered in the same manner. Once a move is selected, all other moves that involve the same clusters are discarded. After all valid moves are applied, the reconstructive algorithm is executed again and repeated until no further changes occur in the set of clusters. An illustration of a successful sweep algorithm applied after other improvement operators, resulting in a reduction in the number of clusters, is shown in Figure 4.12.

Figure 4.12: *A successful sweep group iteration resulting in a reduction of clusters from 19 to 18.*



(a) Clusters before sweep groups

(b) Clusters after sweep groups

4.2.6 Swap and Split Algorithms*

Before finalizing the solution, two improvement heuristics proposed by Vidal [15] and Prins [13], Swap* and Split, are applied to enhance the quality of the route assignment. The Swap* algorithm aims to exchange points between neighboring routes to reduce total travel time, while the Split algorithm attempts to consolidate existing routes into fewer routes without violating time constraints.

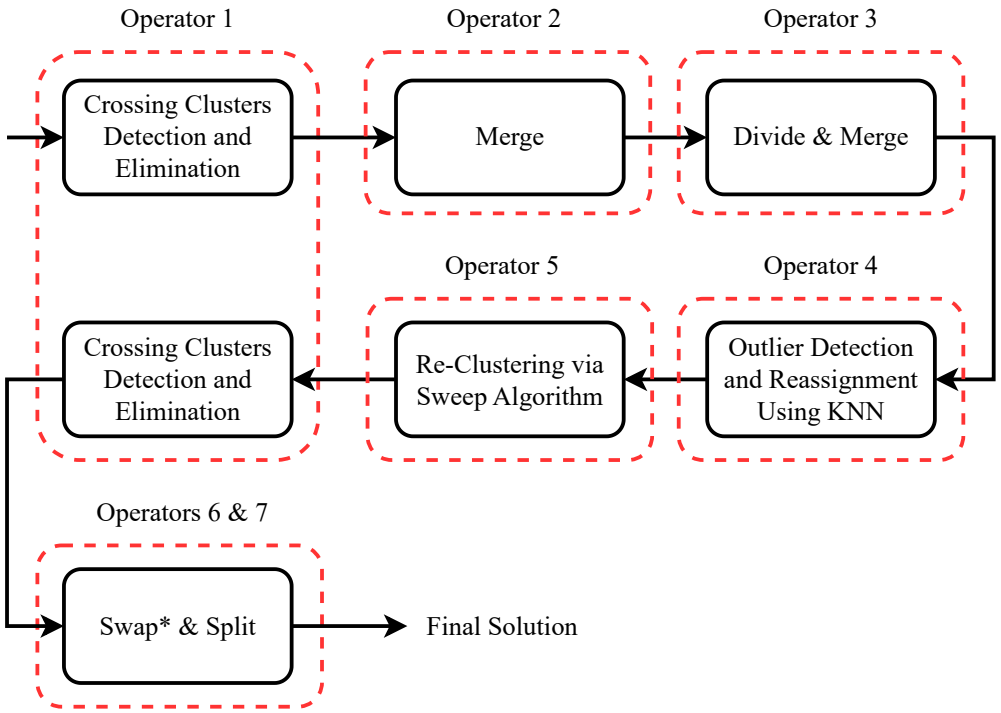
Swap* Algorithm. Swap* is an advanced local search operator that exchanges two points from different routes. Unlike standard point swaps that directly exchange the positions of two nodes, Swap* reinserts each point into the best possible position in the target route, based on an efficient insertion criterion. This allows for a more effective reconfiguration of routes. The swap* is applied to all pairs of neighboring routes. For each pair of routes, the best pair of points to swap is identified, and the swap is executed only if it results in a reduction in the combined route times.

Split Algorithm. The Split algorithm is used to divide a giant tour into feasible routes while minimizing the total number of routes. First, it requires the construction of a giant tour that visits all nodes. This is achieved by heuristically merging existing routes. Starting from an arbitrary route, the algorithm iteratively connects it to the closest available route by optimally matching their endpoints. The process continues, adding the nearest remaining route to the evolving tour until all routes are merged into a single giant tour. Once the giant tour is formed, the split algorithm, as proposed by Prins [13], is applied to divide it into the smallest possible number of feasible routes that respect time constraints. If the resulting solution contains fewer routes than the original, the improvement is accepted; otherwise, the operation is discarded.

4.2.7 Summary

The complete flowchart of the improvement heuristics is illustrated in Figure 4.13.

Figure 4.13: *Flowchart of the improvement heuristic.*



5. NUMERICAL RESULTS

To evaluate the performance and applicability of the algorithm, both numerical experiments and a case study have been carried out. Benchmark instances provided by Uchoa et al. [24], with modifications to the data set to reflect the weights of the points relevant to this problem. These data modifications are explained in detail in Section 5.1. Furthermore, new metrics to evaluate the applicability of the solution are described and analyzed in Section 5.2. For the case study, real-world data from various districts were used. Details of the numerical experiments and the case study are presented in Section 5.3. The performance of the algorithm is compared to the hybrid genetic search (HGS) approach proposed by Vidal [15]. While solving real world problems time matrices representing realistic travel durations between points are needed. To obtain these matrices, Openrouteservice [25] is used. To solve TSP problems during the heuristic methods, the Lin-Kernighan heuristic of the Elkai Python library [26] is used. At the end of each solution, the routes are optimized by the Gurobi solver. For the k-means clustering method, the Python Scikit-learn [27] library is used.

5.1 Data Generation

The benchmark instances provided by Uchoa et al. [24] do not contain crucial weight information for meter reading operations. Alternatively, these instances contain demand data for specific points. By integrating demand data from the instances with point weight data sourced from real-world information, the benchmark instances have been modified.

To achieve this, a weight distribution based on real meter reading data was used as a reference. This distribution outlines the number of points that each specific weight is expected to have, indicating the workload involved in reading each meter. On the other

hand, the original benchmark instances consist of artificial demand values. Although these are not directly relevant to the meter reading process, they can serve as an indicator to distribute relative workload levels.

The transformation procedure involves three main steps. First, all customer points in the instance are sorted in ascending order according to their original demand values. Second, the weight distribution obtained is proportionally scaled to match the number of customer points, maintaining consistency in the frequency of each weight category as observed in real-life data. Finally, scaled weights are assigned to the ordered points, ensuring that points with higher demand values are assigned higher weights. This approach maintains the increasing relationship between demand and workload, while matching the weight distribution with real-world data.

The result is a set of modified benchmark instances where every point is associated with a realistic weight value, allowing the evaluation of algorithms in a scenario that more accurately represents real meter reading operations.

5.2 Applicability of the Proposed Solutions

To solve this problem in the field, it is crucial that the outcome of the solution is implemented by the field workers. Workers should be assigned tasks they find practical and well-structured; otherwise, they might not adhere to the guidance provided by the solution. Solutions that outline tasks that are easily executable by workers are termed applicable solutions. Besides evaluating the effectiveness of the solution based on the objective function value, its applicability must also be evaluated. An optimal solution must perform well in terms of both objective function value and applicability metrics. Experts have defined several criteria to gauge the applicability of such operational solutions. Typically, these metrics are generally associated with clustering quality metrics because there is a tendency in humans to embrace and implement aggregated outcomes in large-scale

logistics problems. After obtaining various types of solution, the results are evaluated using selected clustering quality metrics. Subsequently, experts who supervise field workers validate the correlation between these metrics and the applicability of the solutions.

Among various clustering quality metrics, two widely used and interpretable measures are *compactness* and the *silhouette score*. These metrics not only help evaluate the structural quality of the clusters, but also contribute to understanding the operational feasibility of the generated solution from the perspective of the field workers.

Compactness. This metric measures how tightly the points within a cluster are grouped around a centroid. It is calculated as the mean squared distance from each point to the center of the cluster. A compact cluster implies that the tasks assigned to a worker are geographically close to each other, making the route easier and more intuitive to follow. This enhances the apparent practicality of the plan in practice. The evaluation of a solution involves calculating the average compactness of all clusters within that solution. The compactness value of a cluster is shown in (5.1).

$$C_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \tag{5.1}$$

where:

- C_k: k-th cluster,
- $|C_k|$: Number of points in cluster C_k,
- μ_k: Centroid of cluster C_k,
- $\|x_i - \mu_k\|^2$: Squared distance from point x_i to the cluster centroid.

Silhouette Score. This metric evaluates the quality of data point assignments to clusters by examining both the cohesion of points within the same cluster and the separation from points in other clusters. A higher silhouette score indicates more clearly defined

clusters, which often correlate with worker-friendly clusterings in field applications. The score ranges between -1 and 1, with higher values suggesting that the points are correctly clustered and well separated from the other clusters. The evaluation of a solution involves calculating the average silhouette score of all points within that solution. The silhouette score of a point is shown in (5.2).

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{5.2}$$

where:

- $a(i)$: Average distance from point i to all other points in the same cluster,
- $b(i)$: Minimum average distance from point i to all points in any other cluster,
- $s(i)$: Silhouette score of point i.

Interpretation:

- $s(i) \approx 1$: Point is well-clustered and far from other clusters,
- $s(i) \approx 0$: Point lies near the boundary between two clusters,
- $s(i) < 0$: Point may be incorrectly assigned to the wrong cluster.

In the context of this study, the lower compactness value ensures that the routes are spatially manageable and practical, while a high silhouette score validates the clarity and natural grouping of the routes. Both metrics serve as indicators of whether a solution performs well on the objective function value but is also acceptable for practical deployment in large scale field operations. During the case study, the travel durations between the points obtained from the time matrix are used to calculate the compactness and silhouette score metrics.

5.3 Numerical Experiments and Case Study

Uchoa benchmarks. For a comprehensive evaluation, the benchmark data generated using the Uchoa benchmark set, as detailed in Section 5.1, are utilized. The data set contains 100 instances, upon which the HGS method of Vidal and the CBH approach proposed in this paper have been applied. The results are presented in Appendix C. Table 5.1 presents the summary of the overall performance of the two compared approaches in terms of the average number of routes, average route times, average compactness scores, and average silhouette scores. On average, HGS outperforms CBH in the number of routes measure, whereas CBH demonstrates superior performance in applicability related measures. However, this advantage may partly be caused due to the fact that CBH produces a higher number of routes compared to HGS. To address this, a subset of instances where CBH performs equally well in terms of the number of routes is analyzed separately.

Table 5.1: *Comparison summary of HGS and CBH methods on Uchoa benchmarks.*

	HGS	**CBH**
Number of Instances	100	100
Average Number of Routes	**9.8**	10.8
Average Route Time	433.2	**404.7**
Average Compactness	2.13	**1.55**
Average Silhouette	0.23	**0.31**

The summary of the instances where CBH performs equally well or better than HGS in terms of the number of routes is presented in Table 5.2. This filtered analysis allows for a fairer comparison by controlling for the number of routes, thereby isolating the impact of the clustering approach on other applicability measures. The results indicate that even when the route count advantage is neutralized, CBH continues to outperform HGS in terms of compactness and silhouette scores. These findings strengthen the effectiveness of clustering based heuristics in producing practically relevant solutions. In

particular, the silhouette scores suggest that CBH consistently generates more homogeneous and well separated clusters, which can be more applicable to field workers.

Table 5.2: *Comparison summary of HGS and CBH metrics on Uchoa benchmarks where CBH performs the same as HGS in the number of routes.*

	HGS	**CBH**
Number of Instances	22	22
Average Number of Routes	6.6	6.6
Average Route Time	410.5	424.6
Average Compactness	2.14	**2.00**
Average Silhouette	0.28	**0.35**

Case Study To evaluate the performance and applicability of the proposed algorithm to real world problems, data from two districts from a city of a metropolitan in Türkiye are used. These data are named *district A* and *district B*. Two approaches, namely the HGS method by Vidal and the CBH method introduced in this paper, are implemented on this data to be compared.

District A This data contains more than eight thousand points and almost ninety thousand meters. The district was divided into seven different instances to be solved. The performance comparison of the *clustering based heuristics (CBH)* proposed in this paper, and the HGS approach proposed by Vidal is shown in Table 5.3. The size of the instances, the number of routes, the average time of the routes and the difference between the number of routes proposed by the solutions are given in this table. The applicability metrics, the average compactness and the average silhouette scores for these instances are given in Table 5.4. The plots of solutions for district A are given in Appendix A.

Table 5.3: ***District A*** *Comparison of HGS and CBH on number of clusters, total route time, and difference.*

Instance	**Size**	**HGS**		**CBH**		**Diff.**
		#routes	time	#routes	time	
A1	953	**7**	462.3	8	410,6	1
A2	1560	13	439.6	13	454,7	0
A2	1491	12	437.1	12	449,8	0
A4	942	8	442.9	8	456,0	0
A5	1431	**11**	478.4	12	444,0	1
A6	1204	**10**	476.8	11	441,4	1
A7	925	8	422.7	8	434,3	0

Table 5.4: ***District A*** *Comparison of HGS and CBH on compactness and silhouette scores.*

Instance	**HGS**		**CBH**	
	comp.	sil.	comp.	sil.
A1	4.36	0.10	**1,40**	**0,28**
A2	2.83	0.07	**1,68**	**0,13**
A3	2.01	0.11	**0,89**	**0,24**
A4	1.64	0.18	**1,20**	**0,23**
A5	3.73	-0.03	**0,82**	**0,21**
A6	2.76	0.01	**1,88**	**0,21**
A7	3.56	0.13	**1,15**	**0,28**

District B This data contains more than nineteen thousand points and almost two hundred thousand meters. The district was divided into twelve different instances to be solved. The comparison of these instances for performance and applicability metrics is given in Tables 5.5 and 5.6. The plots of solutions for district B are given in Appendix B.

Table 5.5: ***District B*** *Comparison of HGS and CBH on number of clusters, total route time, and difference.*

Instance	**Size**	**HGS**		**CBH**		**Diff.**
		#routes	time	#routes	time	
B1	2329	**16**	458.7	17	443,7	1
B2	1989	**18**	453.0	19	439,7	1
B3	1563	**12**	453.6	13	431,1	1
B4	1057	9	434.9	9	448,0	0
B5	1963	**16**	458.5	17	445,6	1
B6	1699	12	438.6	12	452,3	0
B7	1458	10	445.3	10	466,2	0
B8	1147	9	437.7	9	454,6	0
B9	2315	**18**	461.6	19	447,9	1
B10	996	**9**	478.8	10	429,7	1
B11	1462	14	442.9	14	452,9	0
B12	1219	10	449.2	10	463,8	0

Based on the numerical experiments performed for districts A and B, it is evident that HGS outperforms CBH in reducing the number of routes. However, CBH provides superior results in applicability metrics, including compactness and silhouette scores, due to the nature of clustering based heuristic approach.

Table 5.6: ***District B*** *Comparison of HGS and CBH on compactness and silhouette scores.*

Instance	**HGS**		**CBH**	
	comp.	sil.	comp.	sil.
B1	3.17	0.03	**1,13**	**0,20**
B2	2.46	0.09	**1,45**	**0,22**
B3	1.92	0.12	**0,73**	**0,23**
B4	2.45	0.14	**1,18**	**0,23**
B5	4.49	-0.02	**2,02**	**0,14**
B6	2.51	0.09	**1,74**	**0,21**
B7	2.78	0.04	**1,85**	**0,18**
B8	1.76	0.14	**0,95**	**0,24**
B9	2.27	0.05	**0,97**	**0,21**
B10	9.15	0.03	**2,84**	**0,25**
B11	4.25	0.12	**3,52**	**0,13**
B12	3.94	0.05	**2,57**	**0,20**

REFERENCES

[1] D. S. Roy, C. Defryn, B. Golden, and E. Wasil, "Data-driven optimization and statistical modeling to improve meter reading for utility companies," *Computers Operations Research*, vol. 145, 2022.

[2] N. Sushma, H. N. Suresh, J. M. Lakshmi, P. N. Srinivasu, A. K. Bhoi, and P. Barsocchi, "A unified metering system deployed for water and energy monitoring in smart city," *IEEE Access*, vol. 11, pp. 80429–80447, 2023.

[3] H. I. Stern and M. Dror, "Routing electric meter readers," *Computers Operations Research*, vol. 6, pp. 209–223, 1979.

[4] J. Wunderlich, M. Collette, L. Levy, and L. Bodin, "Scheduling meter readers for southern california gas company," *Interfaces*, vol. 22, no. 3, pp. 22–30, 1992.

[5] E. Hadjiconstantinou and R. Baldacci, "A multi-depot period vehicle routing problem arising in the utilities sector," *Journal of the Operational Research Society*, vol. 49, pp. 1239–1248, 1998.

[6] J. Mendoza, A. Medaglia, and N. Velasco, "An evolutionary-based decision support system for vehicle routing: The case of a public utility," *Decis. Support Syst.*, vol. 46, pp. 730–742, 2009.

[7] C. Groër, B. Golden, and E. Wasil, "The balanced billing cycle vehicle routing problem," *Networks*, vol. 54, no. 4, pp. 243–254, 2009.

[8] C. Groër, B. Golden, and E. Wasil, "The consistent vehicle routing problem," *Manufacturing Service Operations Management*, vol. 11, pp. 630–643, 10 2009.

[9] S. Kerwin and B. T. Adey, "Integrated planning of operational maintenance programs for water and gas distribution networks," *Journal of Infrastructure Systems*, vol. 27, no. 4, p. 04021039, 2021.

[10] R. Eglese, B. Golden, and E. Wasil, *Chapter 13: Route Optimization for Meter Reading and Salt Spreading*, pp. 303–320. 10 2013.

[11] L. Bodin and L. Levy, *Scheduling of Local Delivery Carrier Routes for the United States Postal Service*, pp. 419–442. Boston, MA: Springer US, 2000.

[12] L. Bodin and L. Levy, "The arc partitioning problem," *European Journal of Operational Research*, vol. 53, no. 3, pp. 393–401, 1991.

[13] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.

[14] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, pp. 611–624, 06 2012.

[15] T. Vidal, "Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood," *Computers Operations Research*, vol. 140, p. 105643, 2022.

[16] M. H. Hà, N. Bostel, A. Langevin, and L.-M. Rousseau, "Solving the close-enough arc routing problem," *Networks*, vol. 63, pp. 107–118, 2014.

[17] C. Miller, A. Tucker, and R. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of Association for Computing Machinery*, vol. 7, pp. 326–329, 1960.

[18] R. Graham, "An efficient algorith for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, no. 4, pp. 132–133, 1972.

[19] G. T. Toussaint, "A simple linear algorithm for intersecting convex polygons," *The Visual Computer*, vol. 1, pp. 118–123, Aug. 1985.

[20] B. B. and, "The surveyor's area formula," *The College Mathematics Journal*, vol. 17, no. 4, pp. 326–337, 1986.

[21] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . Contributors, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[22] S. Gillies and contributors, *Shapely 2.1.1 Documentation*, 2025. Accessed: 2025-06-02.

[23] M. Ahmed and A. N. Mahmood, "A novel approach for outlier detection and clustering improvement," in *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 577–582, 2013.

[24] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845–858, 2017.

[25] H. I. for Geoinformation Technology (HeiGIT), "openrouteservice," 2024. Accessed: 11 June 2025.

[26] F. Dimitrovski, “elkai: A python library for solving travelling salesman problems (tsp) based on lkh 3,” 2023. Version 2.0.1.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

APPENDICES

APPENDIX A. PLOTS OF DISTRICT A

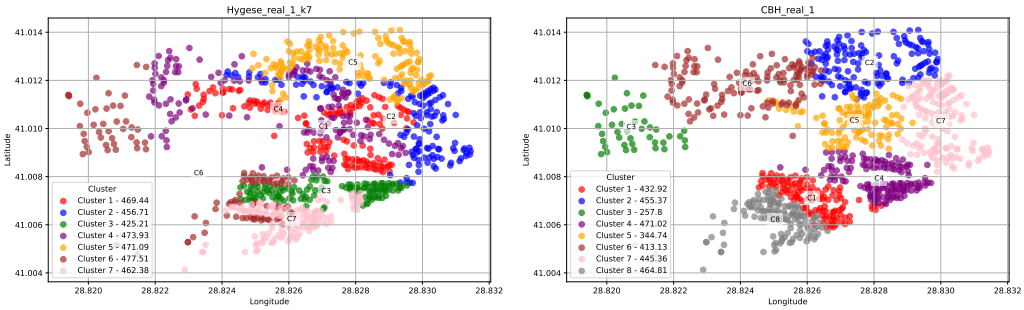


Figure A.1: *District A – Instance A1: HGS (left) vs CBH (right).*

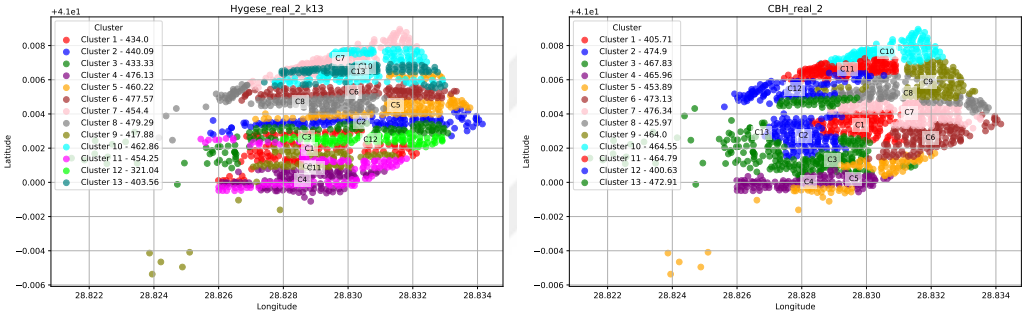


Figure A.2: *District A – Instance A2: HGS (left) vs CBH (right).*

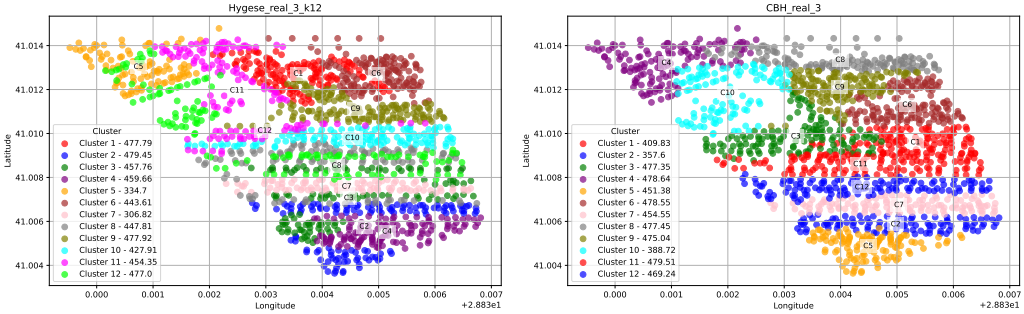


Figure A.3: *District A – Instance A3: HGS (left) vs CBH (right).*

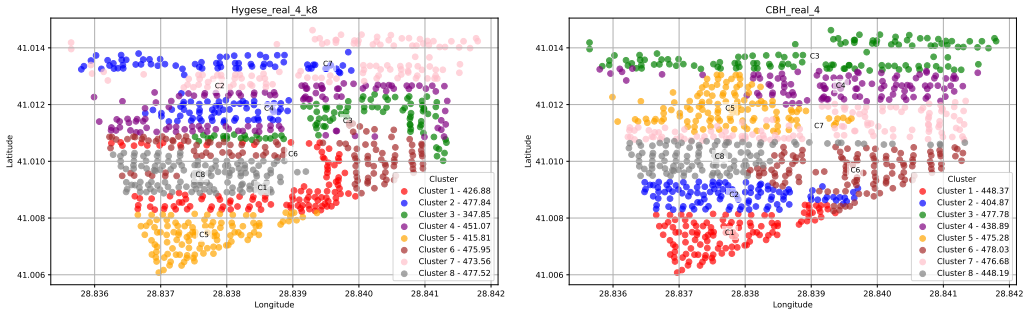


Figure A.4: *District A – Instance A4: HGS (left) vs CBH (right).*

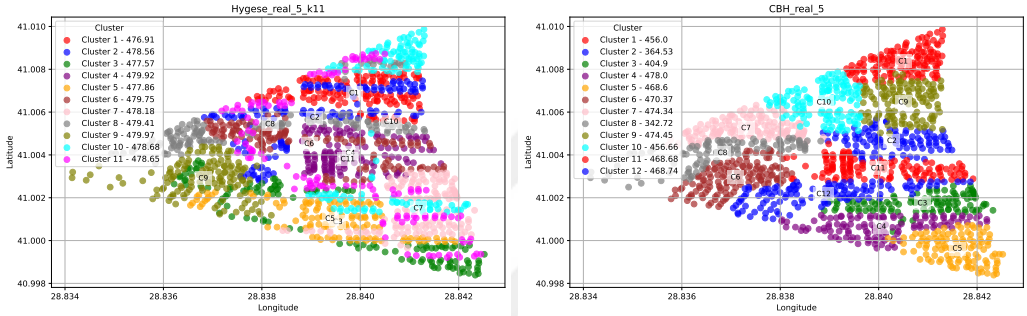


Figure A.5: *District A – Instance A5: HGS (left) vs CBH (right).*

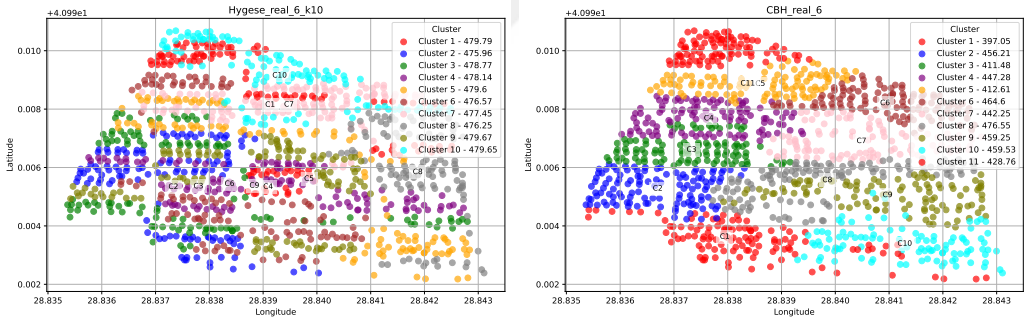


Figure A.6: *District A – Instance A6: HGS (left) vs CBH (right).*

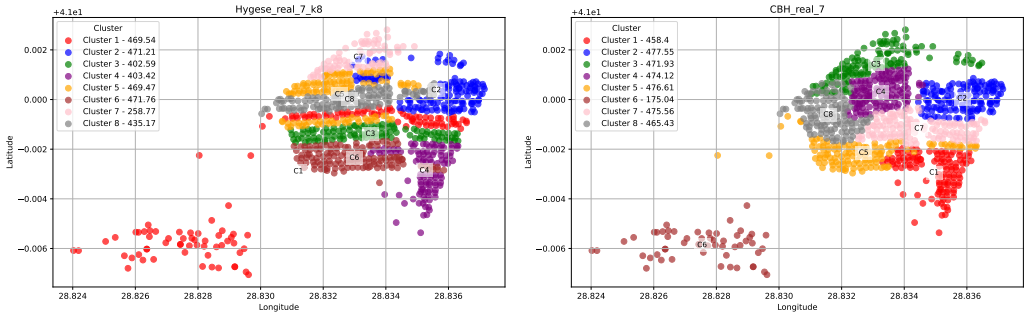


Figure A.7: *District A – Instance A7: HGS (left) vs CBH (right).*

APPENDIX B. PLOTS OF DISTRICT B

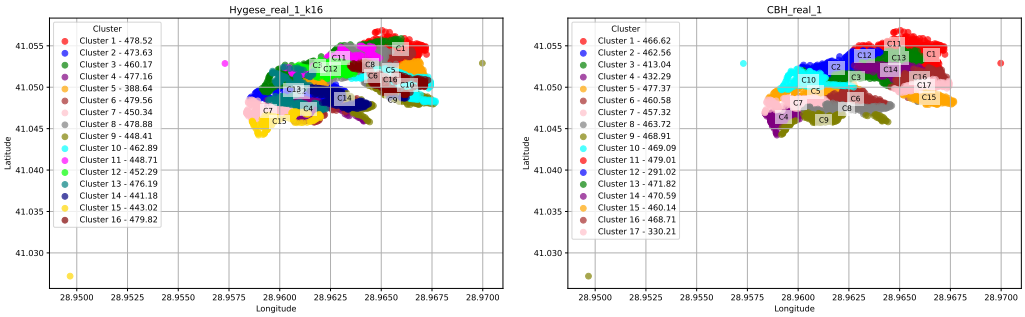


Figure B.1: *District B – Instance B1: HGS (left) vs CBH (right).*

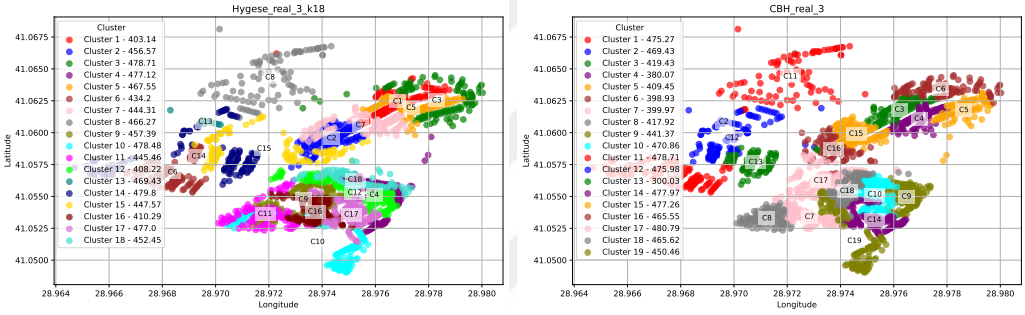


Figure B.2: *District B – Instance B2: HGS (left) vs CBH (right).*

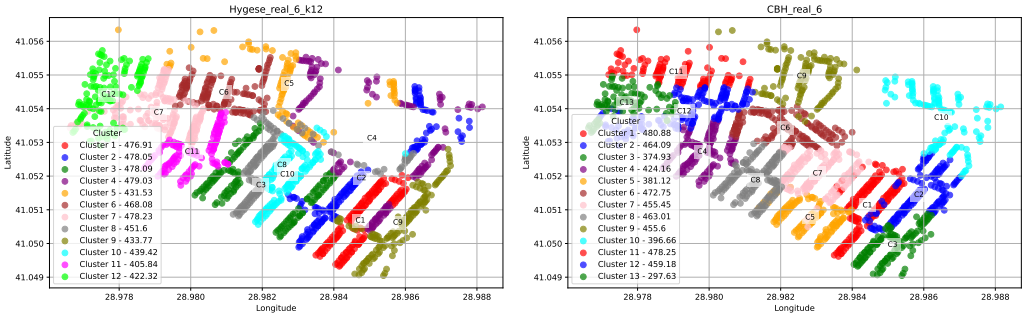


Figure B.3: *District B – Instance B3: HGS (left) vs CBH (right).*

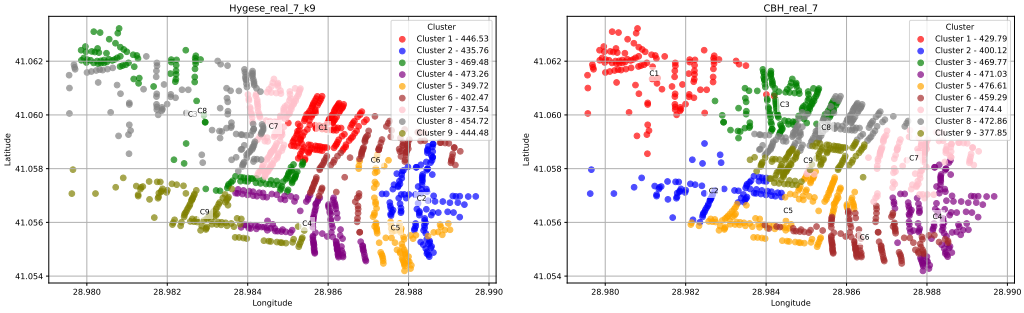


Figure B.4: *District B – Instance B4: HGS (left) vs CBH (right).*

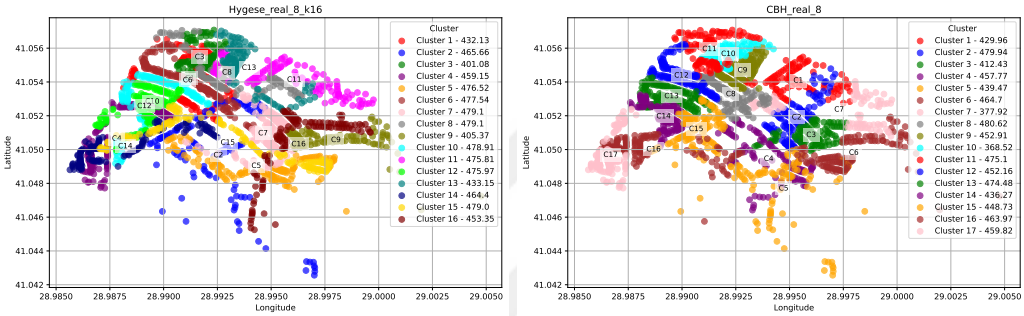


Figure B.5: *District B – Instance B5: HGS (left) vs CBH (right).*

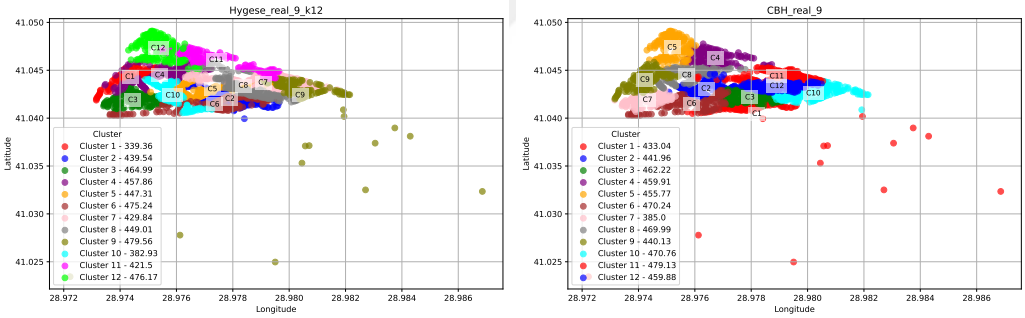


Figure B.6: *District B – Instance B6: HGS (left) vs CBH (right).*

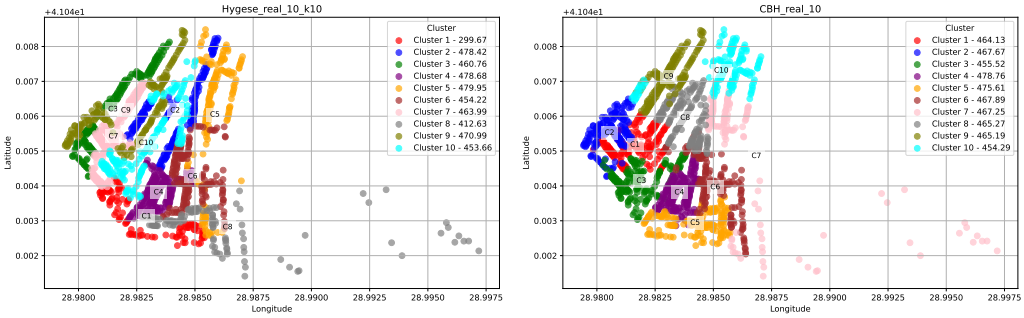


Figure B.7: *District B – Instance B7: HGS (left) vs CBH (right).*

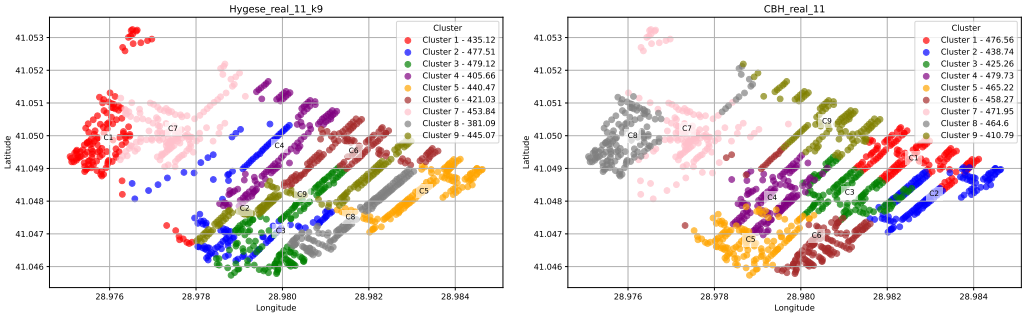


Figure B.8: *District B – Instance B8: HGS (left) vs CBH (right).*

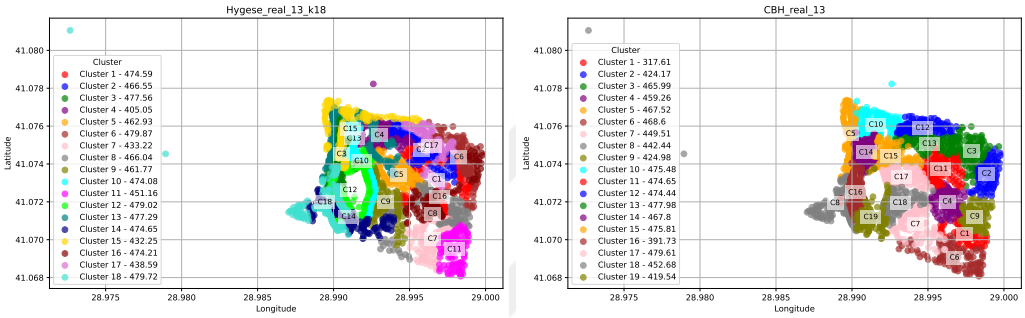


Figure B.9: *District B – Instance B9: HGS (left) vs CBH (right).*

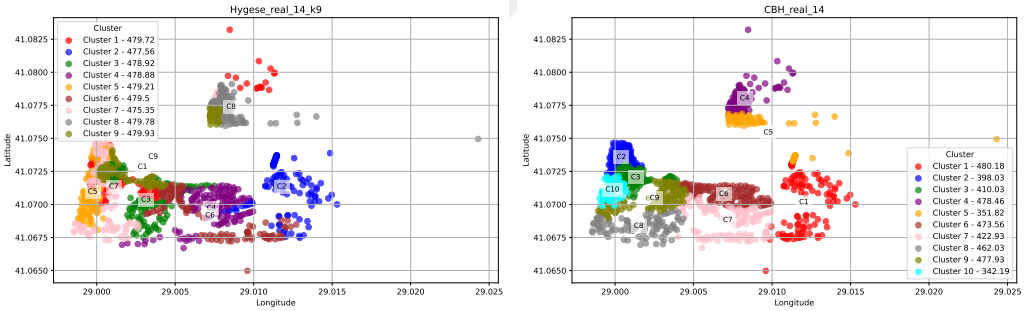


Figure B.10: *District B – Instance B10: HGS (left) vs CBH (right).*

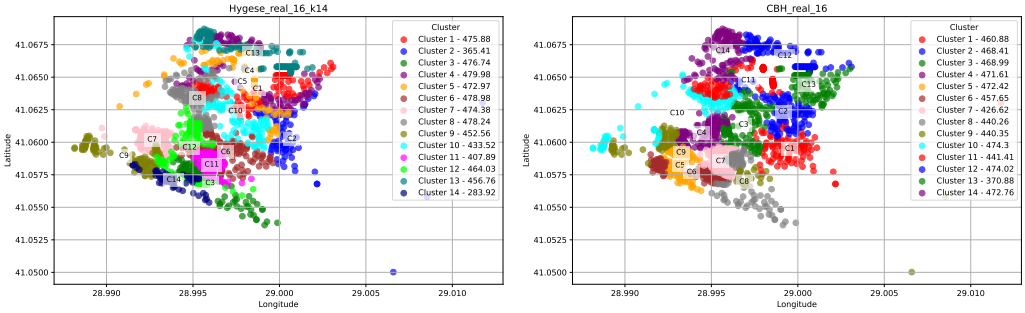


Figure B.11: *District B – Instance B11: HGS (left) vs CBH (right).*

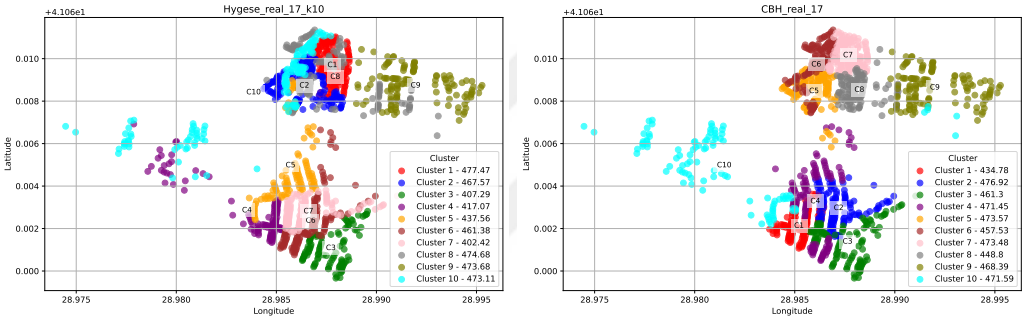


Figure B.12: *District B – Instance B12: HGS (left) vs CBH (right).*

APPENDIX C. OUTCOMES OF UCHOA BENCHMARKS

Table C.1: *Comparison of HGS and CBH results (1).*

Instance	Size	HGS #Routes	HGS Avg. Time	HGS Comp.	HGS Silh.	CBH #Routes	CBH Avg. Time	CBH Comp.	CBH Silh.	Diff
X-n101-k25_transformed	101	4	474,0	4,86	0,38	5	378,1	2,96	0,41	1
X-n106-k14_transformed	106	4	346,3	1,92	0,26	4	380,7	1,87	0,32	0
X-n110-k13_transformed	110	5	425,9	3,03	0,37	5	452,2	3,47	0,33	0
X-n115-k10_transformed	115	5	441,2	5,07	0,18	6	379,1	3,42	0,32	1
X-n120-k6_transformed	120	5	418,9	4,55	0,19	5	442,1	3,17	0,39	0
X-n125-k30_transformed	125	4	440,6	2,12	0,49	4	454,4	2,88	0,41	0
X-n129-k18_transformed	129	5	459,3	3,71	0,25	6	391,9	2,56	0,35	1
X-n134-k13_transformed	134	4	354,3	0,96	0,40	4	362,7	0,95	0,39	0
X-n139-k10_transformed	139	6	400,1	3,12	0,32	7	358,4	2,33	0,35	1
X-n143-k7_transformed	143	6	401,9	3,78	0,26	6	417,2	3,22	0,34	0
X-n148-k46_transformed	148	6	389,7	3,18	0,27	6	401,6	2,98	0,34	0
X-n153-k22_transformed	153	4	463,3	1,69	0,30	5	385,0	1,14	0,30	1
X-n157-k13_transformed	157	4	416,2	1,02	0,27	4	423,0	1,11	0,33	0
X-n162-k11_transformed	162	6	409,4	3,75	0,22	7	368,3	1,92	0,41	1
X-n167-k10_transformed	167	6	446,5	2,90	0,36	7	384,9	2,72	0,33	1
X-n172-k51_transformed	172	6	404,0	2,59	0,45	6	417,0	2,54	0,45	0
X-n176-k26_transformed	176	7	399,1	2,81	0,26	7	425,5	2,97	0,32	0
X-n181-k23_transformed	181	5	432,0	1,47	0,36	6	361,4	1,10	0,34	1
X-n186-k15_transformed	186	7	408,5	3,21	0,23	7	432,0	3,29	0,28	0
X-n190-k8_transformed	190	4	471,2	1,45	0,20	5	387,5	1,08	0,26	1
X-n195-k51_transformed	195	7	405,5	3,04	0,26	7	417,4	2,31	0,37	0
X-n200-k36_transformed	200	5	444,6	1,71	0,30	5	455,9	2,01	0,36	0
X-n204-k19_transformed	204	7	414,4	3,23	0,23	8	378,7	3,07	0,25	1
X-n209-k16_transformed	209	7	450,4	4,00	0,22	8	412,5	2,67	0,29	1
X-n214-k11_transformed	214	5	418,6	1,18	0,29	5	435,5	1,10	0,34	0
X-n219-k73_transformed	219	7	461,2	3,80	0,25	8	409,5	2,43	0,32	1
X-n223-k34_transformed	223	7	450,9	3,18	0,25	8	401,0	1,98	0,36	1
X-n228-k23_transformed	228	6	441,4	2,36	0,35	7	377,9	1,67	0,39	1
X-n233-k16_transformed	233	7	459,5	2,80	0,31	8	406,1	1,89	0,37	1
X-n237-k14_transformed	237	8	428,7	2,30	0,32	9	389,0	1,69	0,35	1
X-n242-k48_transformed	242	8	418,5	2,79	0,24	9	380,9	1,67	0,39	1
X-n247-k50_transformed	247	6	407,4	1,22	0,21	6	414,5	1,16	0,29	0
X-n251-k28_transformed	251	8	403,5	2,10	0,21	8	420,3	2,34	0,31	0
X-n256-k16_transformed	256	7	416,3	1,58	0,29	7	434,4	1,75	0,36	0
X-n261-k13_transformed	261	8	455,9	2,68	0,26	9	422,5	1,95	0,33	1
X-n266-k58_transformed	266	8	421,5	3,16	0,22	9	380,8	2,01	0,27	1
X-n270-k35_transformed	270	8	429,6	2,53	0,30	9	385,7	1,57	0,40	1
X-n275-k28_transformed	275	6	461,7	1,56	0,19	7	394,1	0,95	0,34	1
X-n280-k17_transformed	280	9	419,5	2,74	0,21	10	392,9	2,08	0,28	1
X-n284-k15_transformed	284	7	407,1	1,69	0,15	8	372,7	1,05	0,24	1
X-n289-k60_transformed	289	9	423,8	2,33	0,23	10	389,1	1,61	0,29	1
X-n294-k50_transformed	294	9	432,3	2,51	0,26	10	398,7	1,54	0,36	1
X-n298-k31_transformed	298	9	435,9	2,20	0,31	10	399,9	1,84	0,32	1
X-n303-k21_transformed	303	8	418,4	1,54	0,30	8	424,9	1,05	0,40	0
X-n308-k13_transformed	308	9	433,7	2,80	0,15	10	392,8	1,87	0,27	1
X-n313-k71_transformed	313	9	415,8	2,01	0,25	10	383,3	1,70	0,31	1
X-n317-k53_transformed	317	7	413,7	1,36	0,28	7	425,6	1,15	0,42	0
X-n322-k28_transformed	322	9	450,1	2,50	0,26	10	419,5	1,78	0,31	1
X-n327-k20_transformed	327	9	448,3	2,50	0,23	10	413,3	1,78	0,28	1
X-n331-k15_transformed	331	10	417,9	2,39	0,26	11	391,4	1,62	0,31	1

Table C.2: *Comparison of HGS and CBH results (2).*

Instance	Size	HGS #Routes	HGS Avg. Time	HGS Comp.	HGS Silh.	CBH #Routes	CBH Avg. Time	CBH Comp.	CBH Silh.	Diff
X-n336-k84_transformed	336	10	422,3	2,89	0,21	11	397,8	1,82	0,29	1
X-n344-k43_transformed	344	10	416,7	2,16	0,23	11	386,6	1,59	0,35	1
X-n351-k40_transformed	351	7	470,4	1,40	0,14	8	417,6	1,03	0,21	1
X-n359-k29_transformed	359	10	422,1	2,09	0,25	11	390,4	1,50	0,35	1
X-n367-k17_transformed	367	8	437,9	1,18	0,27	9	392,2	0,76	0,28	1
X-n376-k94_transformed	376	10	456,0	2,70	0,20	12	380,8	1,39	0,33	2
X-n384-k52_transformed	384	11	426,2	2,18	0,23	12	403,2	1,54	0,30	1
X-n393-k38_transformed	393	10	444,2	2,53	0,18	11	419,2	2,14	0,33	1
X-n401-k29_transformed	401	9	439,0	1,27	0,24	10	396,0	0,84	0,34	1
X-n411-k19_transformed	411	8	460,1	1,41	0,22	9	411,0	0,76	0,34	1
X-n420-k130_transformed	420	10	458,6	1,92	0,24	11	420,6	1,75	0,27	1
X-n429-k61_transformed	429	11	457,5	2,69	0,18	13	393,3	1,26	0,32	2
X-n439-k37_transformed	439	11	433,9	2,05	0,19	12	401,1	1,34	0,30	1
X-n449-k29_transformed	449	12	430,6	2,23	0,19	13	401,2	1,41	0,30	1
X-n459-k26_transformed	459	9	450,7	1,06	0,19	10	409,5	0,91	0,25	1
X-n469-k138_transformed	469	12	449,8	1,79	0,24	13	412,3	1,53	0,32	1
X-n480-k70_transformed	480	11	425,6	1,56	0,26	12	395,3	0,81	0,36	1
X-n491-k59_transformed	491	12	430,4	2,14	0,20	12	430,5	1,22	0,35	0
X-n502-k39_transformed	502	9	443,5	0,96	0,11	10	399,6	0,90	0,24	1
X-n513-k21_transformed	513	12	449,1	2,33	0,16	13	421,2	1,55	0,27	1
X-n524-k153_transformed	524	13	449,3	2,26	0,17	15	389,4	1,19	0,31	2
X-n536-k96_transformed	536	11	424,5	1,31	0,23	11	432,6	0,91	0,32	0
X-n548-k50_transformed	548	13	461,4	1,80	0,23	15	397,3	1,35	0,29	2
X-n561-k42_transformed	561	13	460,1	2,03	0,20	15	393,9	1,17	0,31	2
X-n573-k30_transformed	573	9	471,4	0,97	0,11	10	429,1	0,70	0,23	1
X-n586-k159_transformed	586	13	451,6	2,45	0,16	15	391,6	1,05	0,31	2
X-n599-k92_transformed	599	14	450,4	2,12	0,19	16	392,0	0,99	0,36	2
X-n613-k62_transformed	613	14	463,5	2,54	0,13	16	400,8	1,10	0,31	2
X-n627-k43_transformed	627	11	476,2	1,11	0,18	13	392,9	0,60	0,27	2
X-n641-k35_transformed	641	14	468,1	1,76	0,15	16	406,2	1,07	0,30	2
X-n655-k131_transformed	655	11	427,9	0,66	0,18	12	396,2	0,54	0,28	1
X-n670-k130_transformed	670	15	462,6	2,00	0,18	17	401,6	0,94	0,36	2
X-n685-k75_transformed	685	15	448,6	1,85	0,17	16	420,9	1,04	0,29	1
X-n701-k44_transformed	701	15	448,2	1,45	0,22	17	396,9	1,00	0,28	2
X-n716-k35_transformed	716	12	442,0	0,73	0,12	12	441,5	0,61	0,21	0
X-n733-k159_transformed	733	16	451,4	1,67	0,20	18	394,8	1,15	0,29	2
X-n749-k98_transformed	749	14	440,7	0,97	0,16	15	416,3	0,72	0,23	1
X-n766-k71_transformed	766	16	463,6	1,42	0,19	18	409,4	1,05	0,29	2
X-n783-k48_transformed	783	17	452,6	1,75	0,16	19	402,0	1,03	0,28	2
X-n801-k40_transformed	801	17	450,6	1,51	0,21	18	424,1	1,05	0,30	1
X-n819-k171_transformed	819	15	441,0	1,37	0,15	16	410,8	0,69	0,28	1
X-n837-k142_transformed	837	17	461,9	1,54	0,15	18	432,4	1,00	0,27	1
X-n856-k95_transformed	856	17	460,8	1,78	0,13	19	405,2	0,91	0,29	2
X-n876-k59_transformed	876	15	448,1	0,86	0,16	16	420,9	0,88	0,22	1
X-n895-k37_transformed	895	18	468,0	1,78	0,13	21	397,7	0,85	0,31	3
X-n916-k207_transformed	916	18	463,8	1,48	0,18	19	434,4	0,86	0,30	1
X-n936-k151_transformed	936	19	453,7	1,43	0,17	21	406,6	0,94	0,28	2
X-n957-k87_transformed	957	18	467,6	1,44	0,13	20	418,9	1,02	0,25	2
X-n979-k58_transformed	979	16	1,5	0,09	0,14	18	405,5	0,55	0,25	2
X-n1001-k43_transformed	1001	20	451,9	1,43	0,13	22	408,8	0,90	0,25	2