



ORIGINAL

Improve customer service quality, reduce operating expenses, and improve energy sales by the K-MEANS method and path optimization algorithms

Mejora de la calidad del servicio al cliente, reducción de los gastos de explotación y mejora de las ventas de energía mediante el método K-MEANS y algoritmos de optimización de rutas

Redouane Touil¹  , Rachid Marrakh¹ , Taoufiq Belhoussine Drissi¹ , Bahloul Bensassi¹ 

¹Faculty of Science Ain Chock. University Hassan II. Casablanca, Morocco.

Cite as: Touil R, Marrakh R, Belhoussine Drissi T, Bensassi B. Improve customer service quality, reduce operating expenses, and improve energy sales by the K-MEANS method and path optimization algorithms. Data and Metadata. 2025; 4:489. <https://doi.org/10.56294/dm2025489>

Submitted: 24-03-2024

Revised: 21-07-2024

Accepted: 10-11-2024

Published: 01-01-2025

Editor: Adrián Alejandro Vitón Castillo 

Corresponding Author: Redouane Touil 

ABSTRACT

Managing consumer expectations was essential to maintaining customer satisfaction throughout the electricity contract. However, the service provided to customers was based on the location of electrical meters. In the absence of addressing in rural areas, it was too difficult to ensure a comprehensive survey of electrical meter indexes and intervene in time for troubleshooting. The method adopted was the choice of a site with a significant number of meters and energy transformers and the geolocation of electrical installations by a GPS that allowed the assignment of a universal address to electrical installations and facilitated the location of facilities for maintenance and emergency response. The study of optimization algorithms has directed the choice toward the algorithm of the nearest neighbor that remains fast and aims to other algorithms whose number of iterations can be exponential ($n!$ iterations) but less exact. The integration of route optimization algorithms has improved the reactivity of technicians, reduced operational costs, and ensured accurate reading of indexes and transparent billing. The study presents a specific case in Morocco, where route optimization based on GPS coordinates of electric meters in reading indexes has significantly improved efficiency and customer satisfaction. In addition, the K-Means method was used to determine the centroids and clusters that represent respectively the transformation stations and the groups of electrical meters. These groupings allow the calculation of energy sales by transformer station to increase them by reducing energy losses.

Keywords: Electric Meters Geolocation; K-Means; Path Optimization Algorithms.

RESUMEN

La gestión de las expectativas de los consumidores era esencial para mantener su satisfacción a lo largo de todo el contrato de electricidad. Sin embargo, el servicio prestado a los clientes se basaba en la ubicación de los contadores eléctricos. A falta de direccionamiento en las zonas rurales, era demasiado difícil garantizar un estudio exhaustivo de los índices de los contadores eléctricos e intervenir a tiempo para solucionar los problemas. El método adoptado fue la elección de un emplazamiento con un número significativo de contadores y transformadores de energía y la geolocalización de las instalaciones eléctricas mediante un GPS que permitió asignar una dirección universal a las instalaciones eléctricas y facilitó la localización de las instalaciones para el mantenimiento y la respuesta de emergencia. El estudio de los algoritmos de optimización ha orientado la elección hacia el algoritmo del vecino más próximo que sigue siendo rápido y apunta a otros algoritmos cuyo número de iteraciones puede ser exponencial ($n!$ iteraciones) pero menos

exacto. La integración de algoritmos de optimización de rutas ha mejorado la reactividad de los técnicos, ha reducido los costes operativos y ha garantizado una lectura precisa de los índices y una facturación transparente. El estudio presenta un caso concreto en Marruecos, donde la optimización de rutas basada en las coordenadas GPS de los contadores eléctricos en la lectura de índices ha mejorado significativamente la eficacia y la satisfacción de los clientes. Además, se utilizó el método K-Means para determinar los centroides y los conglomerados que representan respectivamente las estaciones de transformación y los grupos de contadores eléctricos. Estas agrupaciones permiten calcular las ventas de energía por centro de transformación para aumentarlas reduciendo las pérdidas de energía.

Palabras clave: Geolocalización de Contadores Eléctricos; K-Means; Algoritmos de Optimización de Rutas.

INTRODUCTION

Managing consumer expectations is essential to maintaining customer satisfaction throughout the contract. Effective interaction with clients, particularly in the face of their complaints and problems, directly influences their perception of service quality. The development of smart business halls promotes a customer-centric approach, allowing for better responsiveness to consumer concerns.⁽¹⁾ This technological evolution not only optimizes resources but also enhances the efficiency of claims responses. In addition, discovering consumer preferences through experimental protocols can enrich management systems, where integrating occupant experience is essential to optimizing connected communities.⁽²⁾ Ultimately, improving reactivity in the face of customer disturbances is necessary to ensure their continued confidence and satisfaction within the framework of electrical energy contracts. In the current context of energy transition, consumers are looking for more flexibility in contractual arrangements related to electricity supply. This flexibility is essential to adapt to variations in tariffs and consumption peaks, as a study's results show that optimization algorithms can facilitate energy demand management while reducing overall costs.⁽³⁾ At the same time, clarity of contractual arrangements is crucial to consumer satisfaction. A well-structured contract, with transparent pricing and service conditions elements, can help build user confidence in energy suppliers. Thus, by combining flexibility and clarity, contractual arrangements can improve energy supply systems' efficiency and encourage wider adoption of sustainable solutions within households.⁽⁴⁾

Faced with the multiple complaints of its customers, the electrical energy distributor always strives to find technical solutions, organizational solutions, or other solutions to meet the needs of consumers. By analyzing the different processes adopted by the National Office of Electricity and Drinking Water (ONEE) in MOROCCO for the distribution of electrical energy and based on bibliographical research, we took a site containing more than 830 electrical meters. By their geographic coordinates, we got through the actions necessary for customer satisfaction, employee satisfaction, and the satisfaction of the electricity distributor, who will be able to reduce the cost of distributing the kilowatt-hour. In this paper, we have clearly defined the methodology adopted to achieve the objectives, such as the exhaustiveness of the electrical meter reading, the distribution of energy bills to customers, and the grouping of electricity meters by HV/LV transformation station. The geolocation allowed us to assign an address to the electric meter based on GPS coordinates for the geographical grouping of meters by reading unit while respecting the number of electrical meters that the technician can read their index. The geographic ordering of these facilities allowed for a comprehensive record of the electric meter indexes.

Bibliographical study

Overview of route optimization in meter reading

The growing importance of route optimization is recognized in several areas, including meter reading, where efficiency and cost reduction are essential. In particular, traffic pattern problems, such as those addressed by methods inspired by the Chinese Crossroads problem, offer innovative solutions to improve data collection. Using technologies such as radio frequency identification (RFID), vehicles can get close enough to the meters without requiring direct access to each site, significantly optimizing the route. Therefore, implementing mathematical models and heuristic search algorithms shows significant promise in maximizing journey coverage while minimizing unnecessary vehicle movements. The combination of these approaches paves the way for optimal solutions for utility companies, helping to reduce collection times while improving operational efficiency.⁽⁵⁾

Recent developments in route optimization algorithms

Recent technological advances, such as sensorization and real-time data access, have been key in optimizing journeys. In particular, these innovations have benefited the routing problems on arcs and related algorithms. These developments have led to various practical applications, notably in automatic meter reading, where specialized algorithms make it possible to optimize the journeys made by vehicles equipped with RFID receivers.

For example, some work highlight the importance of optimization algorithms to maximize operational efficiency while minimizing costs associated with these essential services within urban infrastructure.⁽⁶⁾

By addressing optimization problems such as CEARP, this research paves the way for solutions with a significant economic impact, thus underlining the urgency of exploring more robust algorithmic approaches.⁽⁷⁾

Comparative analysis of algorithms inspired by the Chinese Postman problem

There has been growing interest in optimizing paths for specific tasks, such as automatic meter reading, and algorithms inspired by the Chinese factor problem (PFC) play a crucial role in this area. These algorithms aim to minimize the total cost of travel while ensuring that each street is visited effectively. Arc routing issues, which include PFC, have become critical in various operational contexts, including waste management and mail delivery.⁽⁶⁾ In addition, technological innovations such as RFID devices significantly improve the efficiency of algorithms by reducing the time and cost of reading operations.⁽⁷⁾ Therefore, a comparative analysis of the methods derived from the PFC demonstrates their effectiveness differences and their potential for practical application in the public services sector.

Applications of Route Optimization in Reading Electricity Meters

The optimization of travel in electricity meter reading has significant benefits in terms of efficiency and cost savings. By integrating technologies such as radio frequency identification (RFID), companies can now collect consumption data remotely, eliminating the need for a traditional door-to-door approach. This evolution,⁽⁶⁾ demonstrates how route optimization methods can improve productivity. In addition, appropriate routing models such as the closed intersection problem allow vehicles to approach only a specified distance from meters, thus optimizing routes while respecting operational constraints. Thus, adopting these modern algorithms, as highlighted by studies on the economic impact of logistics optimization,⁽⁷⁾ emphasizes the importance of rigorous planning to maximize operational efficiency while minimizing customer service costs.

Case studies demonstrating efficiency improvements.

The optimization of paths in the electrical meter data collection has demonstrated significant efficiency improvements, as illustrated by the Close Arc Turn (LRCAP) problem. In this context, case studies have highlighted the application of routing algorithms that reduce travel time and energy consumption, allowing companies to achieve significant savings while improving service quality. For example, accurate approaches and metaheuristics, including methods based on scalable local research, have been implemented to optimize vehicle routes, leading to measurable results in recent studies.⁽⁶⁾ This highlights the relevance of optimization methods to the industry and the need to adapt these complex algorithms to the realities of modern data networks, thus enhancing their adoption in daily operational practices.⁽⁸⁾

Problematic

The main complaints received by the Services Agency of the National Office of Electricity and Drinking Water Board are as follows:

- Successive estimates of consumption.
- Blocking of billing.
- Failure to receive energy bills.
- Delay in processing subscription or termination requests.

Analysis of the activities has shown that they are based on the location of the electric meter (figure 1):

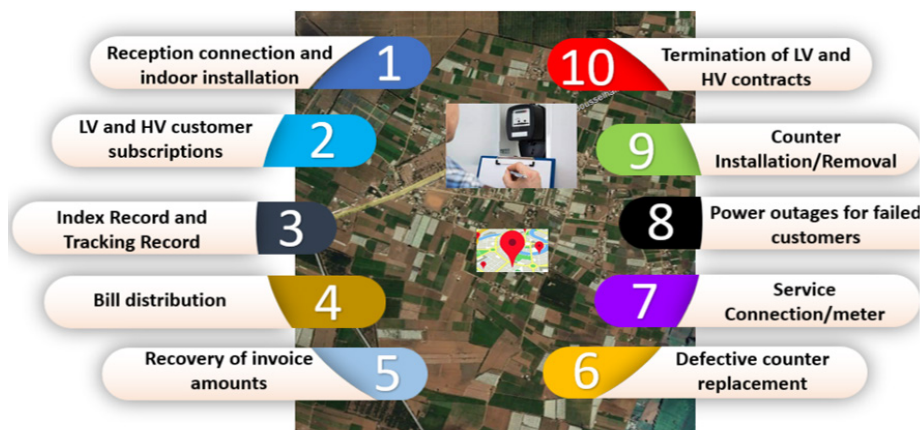


Figure 1. The main activities of the Services Agency are based on the location of the electric meter

It is essential to examine the impact of the non-localization of electricity meters. This could lead to significant inconveniences to the collective governance of the electricity grid. The analysis of the capacity to accommodate at the network level may highlight potential energy efficiency gaps if meter implantation is not considered.⁽⁹⁾ In addition, Shankar⁽¹⁰⁾ highlights the need for quality electrical energy, noting that the lack of adequate location of meters could lead to significant disruptions, directly affecting customer service reliability. These studies clearly illustrate the urgency of geolocating electricity meters to improve the quality of services and overall optimization of the power grid. Therefore, adopting this technology is not just beneficial but also urgent, as it could promote more efficient resource management and a faster response to consumer demands.

This research aimed to geolocate low-voltage meters and high-voltage/low-voltage electrical transformer stations to satisfy customers and reduce operating costs.

METHOD

In rural areas, the current location of the electric meter has limits because it is based on three numbers: tour, chronological order, and sub-order (TOS: ex 80 1200 05). The first number, “T,” represents the power line that can be changed with a novel HV/LV station insertion. The second number, “O,” represents the support number that is not always visible and can be changed with a novel HV/LV station insertion.

The location of the electric meter is used to satisfy customers through one or more of the above tasks:

- Acceptance of connection and installation inside.
- LV and HV customer subscriptions.
- reading and tracking the meter.
- Distribution of invoices.
- Collection of debt.
- Replacement of defective meters.
- Maintenance of connections and meters.
- Disconnection of defaulting customers.
- Meter installation/removal.

Choice of site

In Morocco, we chose LAANABSSA village as a sample for studying and resolving the problem. The choice of the site took into account the complaints received, the insertions of the HV/LV stations (4 stations), and the difficulty in locating the LV installation.

With an area of 6,26 km² and a radius of 2,14 km, the village of LAANABSSA (figure 2) was taken as a site to study the possibility:

- To ensure a comprehensive survey.
- To reduce intervention time and, therefore, reduce workloads.
- To satisfy customers by:
 1. The actual index readings.
 2. Distribution of consumption invoices.
 3. Reduction in response time.
 4. Make energy sales more reliable.
 5. Improve the recovery rate.



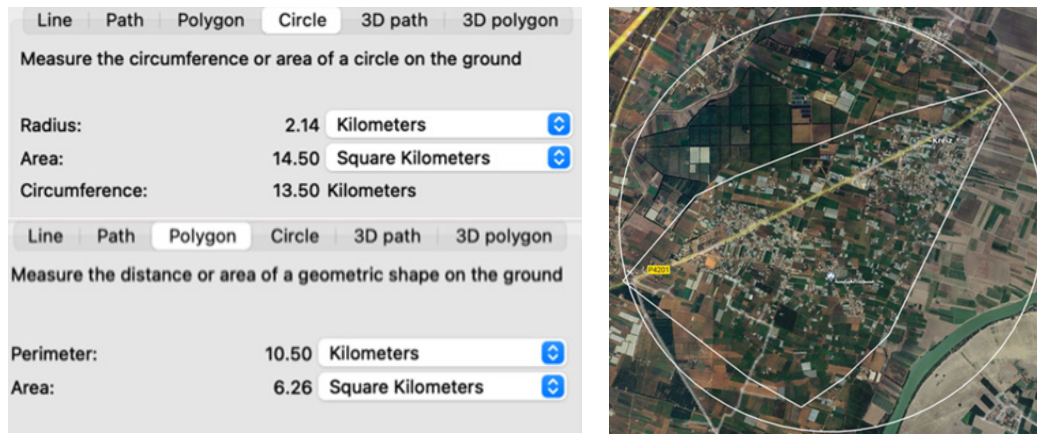


Figure 2. Location of the village LAANABSSA

SAP Data Extraction

ONEE uses SAP as a business process management software that allows data processing and information flow. A list of necessary data was prepared through SAP transactions and cross-referencing the resulting information.

Table 1. Number of electric meters/stations/ Units of reading

HV/LV Station Code	Unite 1	Unite 2	Unite 3	Unite 4	Total
Station 1	1		276	92	369
Station 2	336	120			456
Station 3	6				6
Total	343	120	276	92	831

Collection of the GPS information and coordinates of the low-voltage installations

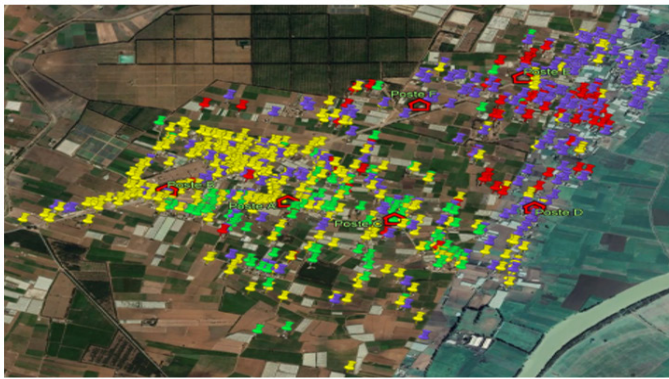
Over 15 days, a team of 3 agents, equipped with a GPS of the GARMIN brand and based on the prepared state, could collect the GPS coordinates of six high-voltage/low-voltage transformer stations and 831 low-voltage electric meters in the village LAANABSSA (figure 3, figure 4).

Based on the order initially assigned to electrical installations, we were able to design the initial scheduling of installations and the distance traveled by the different index readers (figure 5).

Projection of collected data and initial scheduling



Figure 3. Projection the coordinates of high-voltage/low-voltage transformer stations



-  First Index reader
-  Second index reader
-  Third index reader
-  Fourth index reader

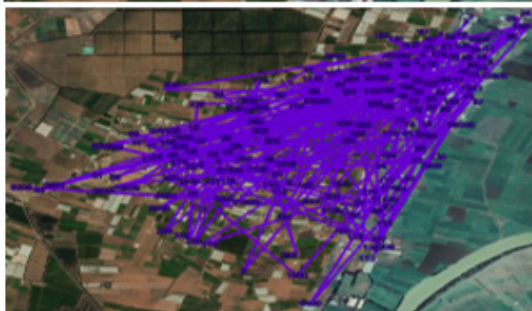
Figure 4. Projection of low voltage electric meters coordinates





To read 343 indexes, the first index reader must cover a distance of 254,699 km




To read 120 indexes, the second index reader must cover a distance of 47,475 km




To read 276 indexes, the third index reader must cover a distance of 274,597 km




To read 92 indexes, the fourth index reader must cover a distance of 43,560 km

Figure 5. The initial scheduling of installations and the distance traveled by the different index readers

Grouping of low voltage meters by station

On the system

We grouped the installations by station.

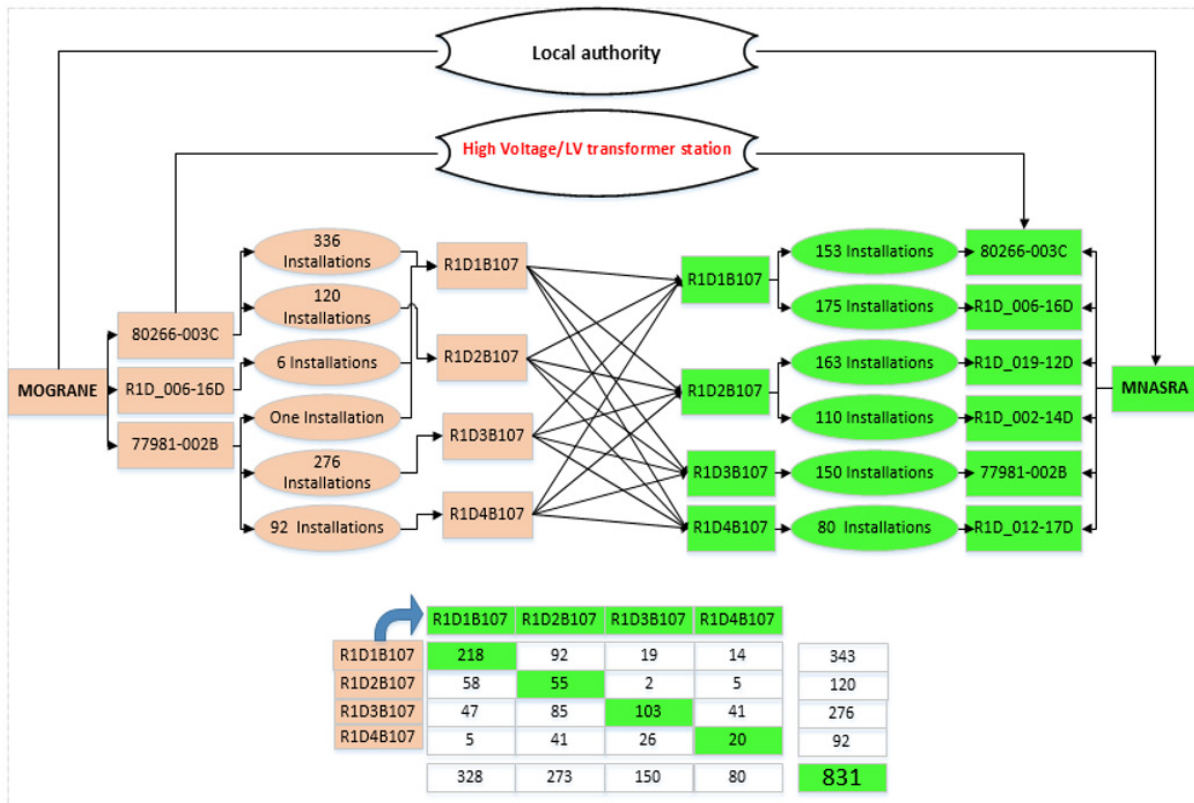


Figure 6. Grouping of low voltage meters by station On the system

On the ground

K-Means is particularly popular in data processing for its efficiency and ease of application in various sectors. K-means is considered one of the simplest unsupervised learning algorithms that solve the well-known clustering problem.⁽¹¹⁾ For example, in natural language processing, this clustering method is used to group documents according to their content, which facilitates the analysis of texts aligned in different languages, as was studied within the parallel corpus English-Persian, where the results show similar clustering behaviors.⁽¹²⁾ The K-Means algorithm is widely applied to identify big-data analytics industry consumer segments. It allows companies to refine their marketing strategies based on the common characteristics of identified groups. In addition, improving K-Means performance through the Advanced K-Mean Clustering Algorithm (AKMCA) demonstrates the continued importance of this method in discovering knowledge from unstructured datasets, Providing practical solutions to optimization and classification problems.⁽¹³⁾ In our case, it allowed us to project the GPS coordinates collected from the field (figure 7).

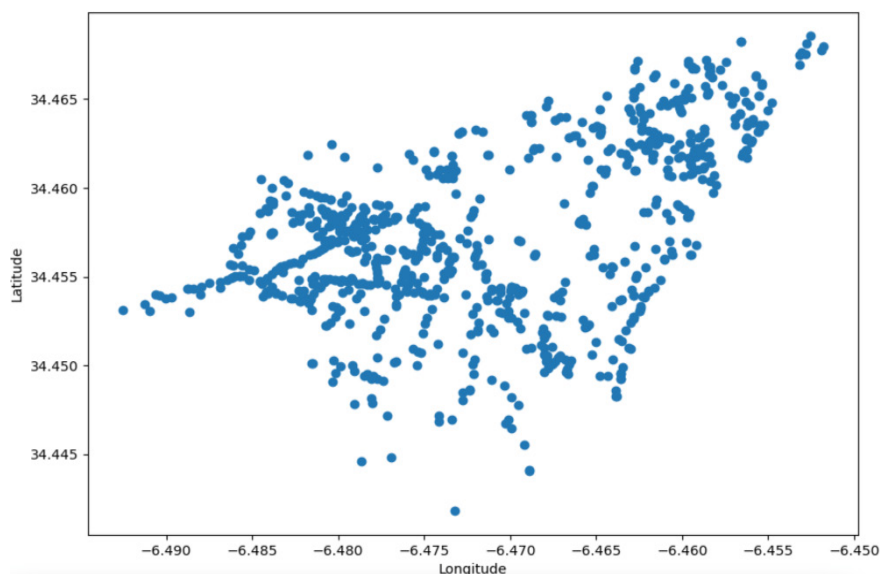


Figure 7. The projection of the GPS coordinates collected using the K-MEANS method

By the K-MEANS method, we have grouped the electrical meters into six clusters whose centroids present high-voltage, low-voltage transformers (figure 8).

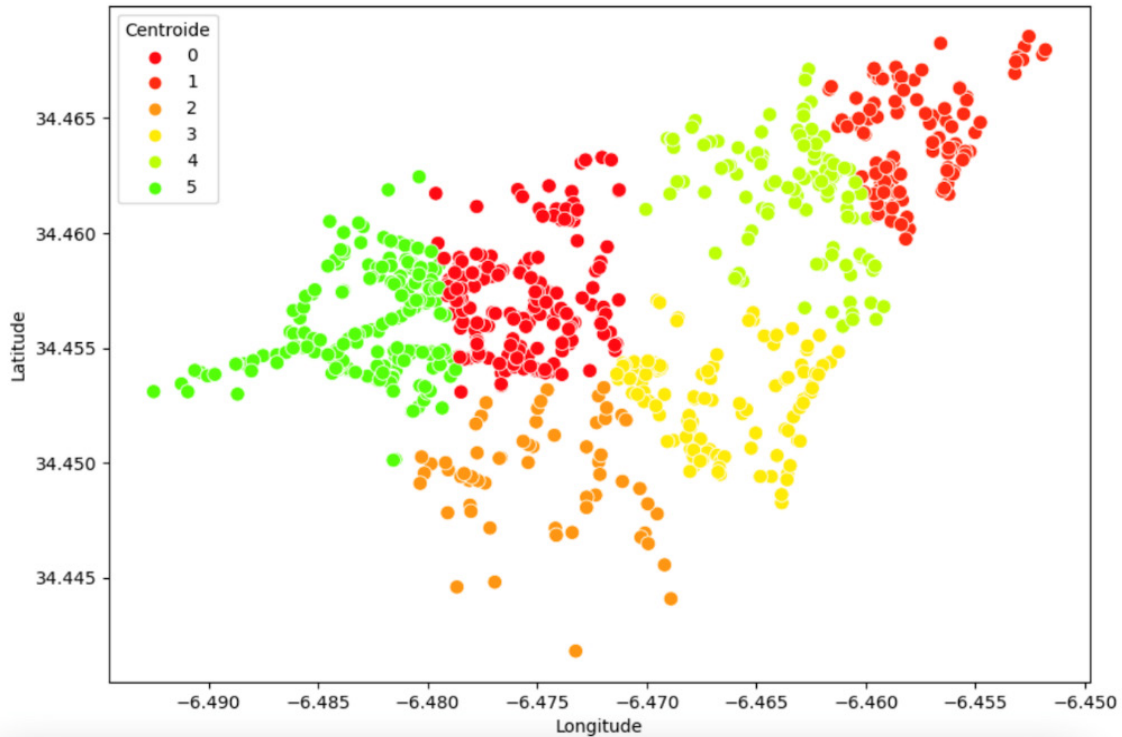


Figure 8. The formation of six clusters by the K-MEANS method

Consolidation of the field installations does not respect the K-MEANS method because the transformers were already installed between 2001 and 2018, and the technicians still play on the separation points between the transformation stations to ensure a balance between them (figure 8, figure 9).

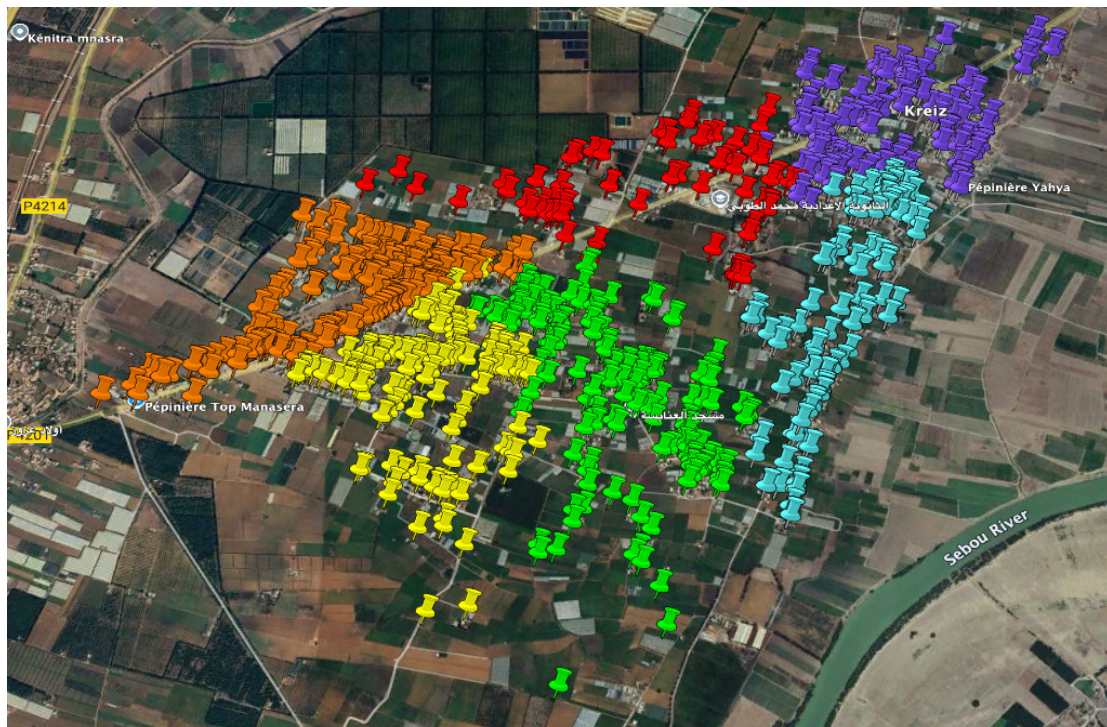


Figure 9. The projection of the GPS coordinates collected of electric meters grouping by transformer

In addition, the grouping for the reading of electrical meters is made by four units of reading (figure 10).

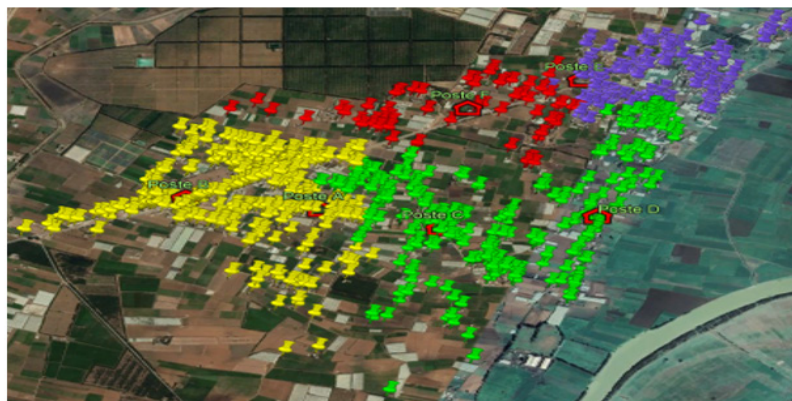


Figure 10. The projection of the GPS coordinates collected of electric meters grouped by unite of reading

Scheduling of electrical meters

Classical algorithmic approaches

Classical algorithmic methods for route optimization are characterized by their systematic approach to solving complex problems by exploiting pre-existing mathematical structures. These algorithms often exploit techniques such as linear programming or rules-based heuristics, thus allowing optimal or near-optimal solutions to be found within a reasonable time. For example, the importance of theoretical concept analysis and experimental evaluation has been highlighted in robust optimization, where recent work highlights the need to establish a stronger link between analysis and algorithm design.⁽¹⁵⁾ In addition, the increasing complexity of modern systems calls for a reassessment of these approaches, especially in the chemical lingerie sector, where the challenge of design for operability highlights crucial economic and environmental implications.⁽¹⁶⁾ Thus, further integration of these classical algorithmic approaches could lead to significant innovations in the field of journey measurement.

Brute force algorithm and its applications in shortest path problems

The importance of search algorithms in solving complex problems is particularly evident in the context of shortest-path problems. The brute force approach has exponential time complexity and has prohibitively long execution times, even for moderately sized models.⁽¹⁴⁾ Indeed, the brute force algorithm, although often criticized for its inefficiency, plays a crucial role in situations where exhaustive research is required to ensure an optimal solution. This approach is based on a solid foundation and allows all possible permutations of a set of points to be considered, thus ensuring that no option is overlooked. However, as recent work, such as those presented in the movement planning processes, indicates, brute force search and AI-based optimizations can significantly improve computing time while maintaining the quality of solutions obtained.⁽¹⁷⁾ In parallel, methods for revising specifications incorporating sub-objective bias, as demonstrated by research, can help further refine decision-making.⁽¹⁸⁾ These developments suggest a promising future for applying such algorithms in increasingly complex scenarios.

Code with example

```
import pandas as pd
import numpy as np
```

```

import plotly.express as px
from math import cos , sin , acos, radians
from __future__ import annotations
def generate_permutations(arr: list[int]) -> list[list[int]]:
    if len(arr) == 1:
        return [arr]
    permutations: list[list[int]] = []
    for i in range(len(arr)):
        extracted_element = arr[i]
        remaining_elements = arr[:i]+arr[i+1:]
        smaller_permutations = generate_permutations(remaining_elements)
        for permutation in smaller_permutations:
            permutation.insert(0, extracted_element)
            permutations.append(permutation)
    return permutations
def tsp_brute_force(graph: list[list[float]]) -> tuple[float,list[int]]:
    n: int = len(graph)
    vertices: list[int] = list(range(n))
    permutations: list[list[int]] = generate_permutations(vertices)
    min_cost: float = float('inf')
    min_tour: list[int] = []
    for permutation in permutations:
        permutation.append(permutation[0])
        cost: float = 0
        for i in range(n):
            cost += graph[permutation[i]][permutation[i+1]]
        if cost < min_cost:
            min_cost = cost
            min_tour = permutation
    return min_cost, min_tour
if __name__ == '__main__':
    Data1=[(-6,4823,34,454211), (-6,481987,34,454207), (-6,482039,34,453912),\
(-6,482177,34,453607), (-6,482252,34,453582), (-6,482657,34,453746),\
(-6,482735,34,453769), (-6,484358,34,453892), (-6,481701,34,453756),\
(-6,481593,34,453762)]
    graph : list[list[float]] = [[acos(sin(radians(y2)) * sin(radians(y1)) + cos(radians(y2)) * cos(radians(y1)) *
cos(radians(x2)- radians(x1))) * 6371 *1000 for x2, y2 in Data1] for x1, y1 in Data1]
    min_cost: float; min_tour: list[int]
    min_cost, min_tour = tsp_brute_force(graph)
    print(f'Minimum cost: {min_cost}')
    print(f'Minimum path: {min_tour}')

```

Result

Minimum cost: 560,4452657121925 meters

Minimum path: [3, 8, 9, 2, 1, 0, 7, 6, 5, 4, 3]:

Projection the result



Figure 11. Projection of brute force algorithm

The Backtracking algorithm and its applications in shortest-path problems

The applications of algorithmic programming in path optimization are diverse and demonstrate the effectiveness of methods such as backtracking. This approach, which comprehensively explores the different path possibilities, is particularly suited to scenarios where complex constraints must be considered. For example, backtracking helps solve the back-to-top problem in graphs, similar to the shortest path algorithm. Indeed, the Backtracking algorithm can lead to optimal solutions by systematically examining each option, although it may be subject to time limitations, especially in large graphs with strong connectivity. In addition, the complexity of graph representation, as in the case of the maximum common subgraph (MCS), highlights the need for a thorough algorithmic reflection to master the challenges associated with these complex problems.⁽¹⁹⁾

Code with example

```
import pandas as pd
import numpy as np
import plotly.express as px
from math import cos , sin , acos, radians
from __future__ import annotations
def rec(graph: list[list[float]], tour: list[int], cost: float, min_tour: float, min_cost: list[int]) -> tuple[float,list[int]]:
    n: int = len(graph)
    if len(tour) == n:
        tour.append(tour[0])
        cost += graph[tour[-2]][tour[-1]]
        if cost < min_cost:
            min_cost = cost
            min_tour = tour.copy()
        cost -= graph[tour[-2]][tour[-1]]
        tour.pop()
        return min_cost, min_tour
    for i in range(n):
        if i not in tour:
            tour.append(i)
            cost += graph[tour[-2]][tour[-1]]
            if cost < min_cost:
                min_cost, min_tour = rec(graph, tour, cost, min_tour, min_cost)
            cost -= graph[tour[-2]][tour[-1]]
            tour.pop()
    return min_cost, min_tour
def tsp_backtracking(graph: list[list[float]]) -> tuple[float,list[int]]:
    start: int = 0
    tour: list[int] = [start]
    cost: float = 0
    min_tour: list[int] = None
    min_cost: float = float('inf')
    min_cost, min_tour = rec(graph, tour, cost, min_tour, min_cost)
    return min_cost, min_tour
if __name__ == '__main__':
    Data1=[(-6,4823,34,454211), (-6,481987,34,454207), (-6,482039,34,453912),\
(-6,482177,34,453607), (-6,482252,34,453582), (-6,482657,34,453746),\
(-6,482735,34,453769), (-6,484358,34,453892), (-6,481701,34,453756),\
(-6,481593,34,453762)]
    graph : list[list[float]] = [[acos(sin(radians(y2)) * sin(radians(y1)) + cos(radians(y2)) * cos(radians(y1)) *
cos(radians(x2)- radians(x1))) * 6371 *1000 for x2, y2 in Data1] for x1, y1 in Data1]
    min_cost: float; min_tour: list[int]
    min_cost, min_tour = tsp_backtracking(graph)
    print(f'Minimum cost: {min_cost}')
    print(f'Minimum tour: {min_tour}')
```

Result

Minimum cost: 560,4452657121925 meters
Minimum tour: [0, 7, 6, 5, 4, 3, 8, 9, 2, 1, 0]

Projection the result

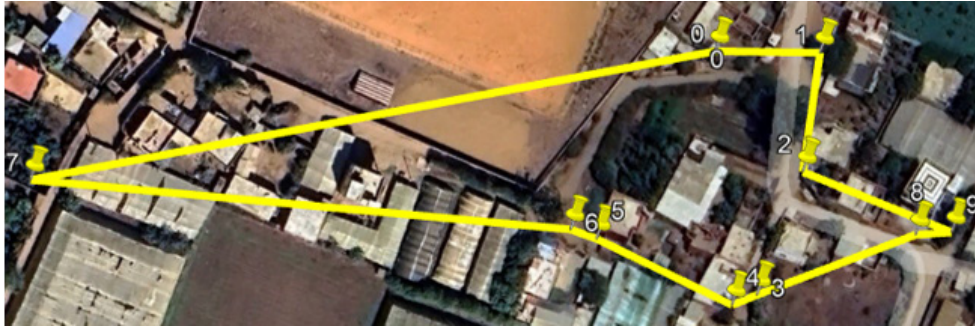


Figure 12. Projection of backtracking algorithm

The Dynamic Programming algorithm and its applications in shortest-path problems

The effectiveness of dynamic programming algorithms in solving shortest-path problems is based on their ability to break down a complex problem into simpler subproblems. This method, used in complex transport networks, allows for the rapid calculation of optimal routes, as shown by the ability to obtain driving directions in milliseconds at a continental scale.⁽²⁰⁾ In addition, tools like the JGraphT library offer remarkable flexibility to model various graphs, which is fundamental for social and transport network analysis.⁽²¹⁾ These algorithms are particularly effective when data changes dynamically, such as real-time traffic. Thus, dynamic programming is positioned as a pillar in path optimization, integrating both time complexity and practical application concerns.

Code with example

```
import pandas as pd
import numpy as np
import plotly.express as px
from math import cos , sin , acos , radians
from __future__ import annotations
def tsp(mat: list[list[float]], s: int, A: int, memo: dict[tuple[int,int],float], nxt: dict[tuple[int,int],int]) -> float:
    if (s, A) in memo:
        return memo[(s, A)]
    if not A:
        return mat[s][0]
    else:
        memo[(s, A)], nxt[(s, A)] = min([(tsp(mat, i, A^(1<<i), memo, nxt)+mat[s][i], i) for i in range(len(mat)) if A&(1<<i)])
        return memo[(s, A)]
def tsp_dp(mat: list[list[float]]) -> tuple[float,list[int]]:
    n: int = len(mat)
    s: int = 0
    A: int = (1<<n) - 1
    A ^= 1
    memo: dict[tuple[int,int],float] = {}
    nxt: dict[tuple[int,int],int] = {}
    min_cost: float = tsp(mat, s, A, memo, nxt)
    min_tour: list[int] = [s]
    while A:
        s = nxt[(s, A)]
        min_tour.append(s)
        A ^= 1<<s
    min_tour.append(0)
    return min_cost, min_tour
if __name__ == '__main__':
    Data1=[(-6,4823,34,454211), (-6,481987,34,454207), (-6,482039,34,453912),\
(-6,482177,34,453607), (-6,482252,34,453582), (-6,482657,34,453746),\
(-6,482735,34,453769), (-6,484358,34,453892), (-6,481701,34,453756),\
(-6,481593,34,453762)]
    graph : list[list[float]] = [[acos(sin(radians(y2)) * sin(radians(y1)) + cos(radians(y2)) * cos(radians(y1))) *
```

```

cos(radians(x2)- radians(x1)) * 6371 *1000 for x2, y2 in Data1] for x1, y1 in Data1]
min_cost, min_tour = tsp_dp(graph)
min_tour = [Data1[i] for i in min_tour]
print(f'Shortest tour: {min_tour} (total length: {min_cost})')

```

Result

Shortest tour: [(-6,4823, 34,454211), (-6,484358, 34,453892), (-6,482735, 34,453769), (-6,482657, 34,453746), (-6,482252, 34,453582), (-6,482177, 34,453607), (-6,481701, 34,453756), (-6,481593, 34,453762), (-6,482039, 34,453912), (-6,481987, 34,454207), (-6,4823, 34,454211)] (total length: 560,4452657121926 meters)

Projection the result

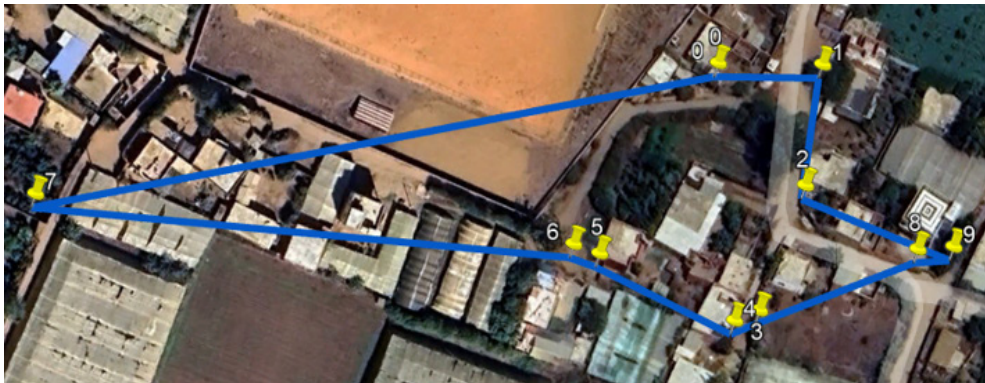


Figure 13. Projection of dynamic programming algorithm

The nearest neighbor algorithm and its applications in shortest-path problems

Recent research has highlighted the importance of efficient algorithms to solve complex path optimization problems. In particular, the nearest neighbor algorithm is distinguished by its simplicity and efficiency in addressing shorter-path problems in varied environments. The latter establishes local connections between nodes, allowing it to explore the research space quickly while ensuring convergence towards optimal or near-optimal solutions. For example, in transport or communication applications where minimization of travel time is crucial, methods such as those described in the network literature highlight the importance of quality of service and congestion.⁽²²⁾ In addition, the advent of asymptotically optimal algorithms such as LBT-RRT illustrates the continuity and evolution of optimization techniques that enrich the practical applications resulting from the nearest neighbor algorithm.⁽²³⁾ This analytical framework thus offers a clear vision of the potential to be exploited in complex systems.

Code with example

```

import pandas as pd
import numpy as np
import plotly.express as px
from math import cos , sin , acos, radians
from __future__ import annotations
from __future__ import annotations
def tsp_nearest_neighbor(graph: list[list[float]], start: int=0) -> tuple[float,list[int]]:
    n: int = len(graph)
    vertex: int = start
    tour: list[int] = [start]
    visited: set[int] = set()
    visited.add(start)
    total_cost: float = 0
    for _ in range(n-1):
        min_cost: float = float('inf')
        min_neighbor: int = -1
        neighbor: int
        for neighbor in range(n):
            if neighbor not in visited and graph[vertex][neighbor] < min_cost:
                min_cost = graph[vertex][neighbor]

```

```

        min_neighbor = neighbor
        total_cost += min_cost
        tour.append(min_neighbor)
        visited.add(min_neighbor)
        vertex = min_neighbor
        total_cost += graph[tour[-1]][tour[0]]
        tour.append(tour[0])
        return total_cost, tour
def tsp_repetitive_nearest_neighbor(graph: list[list[float]]) -> tuple[float,list[int]]:
    n: int = len(graph)
    min_cost: float = float('inf')
    min_tour: list[float] = []
    for start in range(n):
        total_cost: float; tour: list[int]
        total_cost, tour = tsp_nearest_neighbor(graph, start)
        if total_cost < min_cost:
            min_cost = total_cost
            min_tour = tour
    return min_cost, min_tour
if __name__ == '__main__':
    Data1=[(-6,4823,34,454211),\
(-6,481987,34,454207), (-6,482039,34,453912), (-6,482177,34,453607),\
(-6,482252,34,453582), (-6,482657,34,453746), (-6,482735,34,453769),\
(-6,484358,34,453892), (-6,481701,34,453756), (-6,481593,34,453762)]
    graph : list[list[float]] = [[acos(sin(radians(y2)) * sin(radians(y1)) + cos(radians(y2)) * cos(radians(y1)) *
cos(radians(x2)- radians(x1))) * 6371 *1000 for x2, y2 in Data1] for x1, y1 in Data1]
    min_cost: float; min_path: list[int]
    min_cost, min_path = tsp_repetitive_nearest_neighbor(graph)
    min_tour = [Data1[i] for i in min_path]
    print(f'Shortest tour: {min_tour} (total length: {min_cost})')

```

Result

Shortest tour: [(-6,4823, 34,454211), (-6,481987, 34,454207), (-6,482039, 34,453912), (-6,481701, 34,453756), (-6,481593, 34,453762), (-6,482177, 34,453607), (-6,482252, 34,453582), (-6,482657, 34,453746), (-6,482735, 34,453769), (-6,484358, 34,453892), (-6,4823, 34,454211)] (total length: 561,3655418653942 meters)

Projection the result



Figure 14. Projection of nearest neighbor algorithm

Chortest edges algorithm and its applications in shortest path problems

Recent advances in path optimization algorithms reveal promising applications of the Chortest edges algorithm in various contexts, including those related to transport networks. This algorithm excels in managing complex dynamics encountered in journey planning scenarios, where time and resources must be optimized simultaneously. Taking into account the multi-criteria aspects often related to public transport planning, it is essential to integrate robust optimization concepts, highlighting the increased relevance of this algorithm in real situations. Indeed, as indicated by research that highlights rapid processing methods on large-scale road

networks,⁽²⁰⁾ the combination of advanced algorithmic approaches, both theoretical and experimental, Provides viable and effective solutions for large-scale instances.⁽¹⁵⁾

Code with example

```
import pandas as pd
import numpy as np
import plotly.express as px
from math import cos , sin , acos, radians
from __future__ import annotations
from collections import defaultdict
class DisjointSet:
    def __init__(self, elems: list[any]):
        self.parent: dict[any,any] = {}
        self.size: dict[any,int] = {}
        elem: any
        for elem in elems:
            self.make_set(elem)
    def make_set(self, a: any) -> None:
        self.parent[a] = a
        self.size[a] = 1
    def find(self, a: any) -> any:
        if self.parent[a] == a:
            return a
        else:
            self.parent[a] = self.find(self.parent[a])
            return self.parent[a]
    def union(self, a: any, b: any) -> None:
        root_a = self.find(a)
        root_b = self.find(b)
        if root_a == root_b:
            return
        elif self.size[root_a] <= self.size[root_b]:
            self.parent[root_a] = root_b
            self.size[root_b] += self.size[root_a]
        else:
            self.parent[root_b] = root_a
            self.size[root_a] += self.size[root_b]
def tsp_sorted_edges(graph: list[list[float]]) -> tuple[float,list[int]]:
    n: int = len(graph)
    vertices: list[int] = list(range(n))
    edges: list[tuple[int,int]] = [(i, j) for i in range(n) for j in range(i+1, n)]
    edges.sort(key=lambda edge: graph[edge[0]][edge[1]])
    tour_graph: dict[int,list[tuple[int,float]]] = defaultdict(lambda: [])
    ds: DisjointSet[int] = DisjointSet(vertices)
    nb_edges: int = 0
    u: int; v: int
    for u, v in edges:
        w: float = graph[u][v]
        if ds.find(u) != ds.find(v) and len(tour_graph[u]) < 2 and len(tour_graph[v]) < 2:
            tour_graph[u].append((v, w))
            tour_graph[v].append((u, w))
            nb_edges += 1
            ds.union(u, v)
            if nb_edges == len(vertices)-1:
                break
    start: int; end: int
    start, end = [u for u in tour_graph if len(tour_graph[u]) == 1]
    tour_graph[start].append((end, graph[start][end]))
    tour_graph[end].append((start, graph[start][end]))
    tour: list[int] = [start]
```

```

total_cost: float = 0
vertex: int = start
for _ in range(n):
    neighbor: int; w: float
    neighbor, w = tour_graph[vertex][0]
    total_cost += w
    tour.append(neighbor)
    tour_graph[vertex].remove((neighbor, w))
    tour_graph[neighbor].remove((vertex, w))
    vertex = neighbor
return total_cost, tour
if __name__ == '__main__':
    Data1=[(-6,4823,34,454211), (-6,481987,34,454207), (-6,482039,34,453912),\
(-6,482177,34,453607), (-6,482252,34,453582), (-6,482657,34,453746),\
(-6,482735,34,453769), (-6,484358,34,453892), (-6,481701,34,453756),\
(-6,481593,34,453762)]

    graph : list[list[float]] = [[acos(sin(radians(y2)) * sin(radians(y1)) + cos(radians(y2)) * cos(radians(y1)) *
cos(radians(x2)- radians(x1))) * 6371 *1000 for x2, y2 in Data1] for x1, y1 in Data1]
    min_cost: float; min_tour: list[int]
    min_cost, min_tour = tsp_sorted_edges(graph)
    min_tour = [Data1[i] for i in min_tour]
    print(f'Shortest tour: {min_tour} (total length: {min_cost})')

```

Result

Shortest tour: [(-6,4823, 34,454211), (-6,481987, 34,454207), (-6,482039, 34,453912), (-6,481701, 34,453756), (-6,481593, 34,453762), (-6,482177, 34,453607), (-6,482252, 34,453582), (-6,482657, 34,453746), (-6,482735, 34,453769), (-6,484358, 34,453892), (-6,4823, 34,454211)] (total length: 561,3655418653942 meters)

Projection the result



Figure 15. Projection chortest edges algorithm

The Christofides algorithm and its applications in shortest-path problems

The optimization of paths in a graph is a significant issue in algorithmic research, and the algorithm of Christofides proves to be a powerful method, especially for solving the problem of the trade traveler in symmetric metrics. It is well known for its approach that combines the construction of a minimum covering shaft and the resolution of the perfect matching problem, thus guaranteeing an approximation ratio of $3/2$. However, recent improvements have overcome the known limitations of this algorithm. For example, the work of Hooegeveen introduced a variant that reached a ratio of $5/3$, but new adaptations have proposed even more precise approximations, such as the algorithm that achieves a ratio of $(1+ 5)/2$ for the s-t TSP path problem.⁽²⁴⁾ These advances demonstrate the importance of Christofides' algorithm in contemporary path optimization algorithms and open up interesting leads for related problems, such as T-join.⁽²⁵⁾

Code with example

```

import pandas as pd
import numpy as np
import plotly.express as px
from math import cos , sin , acos, radians

```



```

from __future__ import annotations
from collections import defaultdict
import random
import fibheap as fh
def prim(mat: list[list[float]]) -> dict[int,list[tuple[int,float]]]:
    n = len(mat)
    mst: dict[int,list[tuple[int,float]]] = defaultdict(lambda: [])
    start: int = 0
    cost: dict[int,float] = {}
    parent: dict[int,int] = {}
    nodes: dict[int,fh.Node] = {}
    queue: fh.Fheap = fh.makefheap()
    u: int
    for u in range(n):
        cost[u] = 0 if u == start else float('inf')
        parent[u] = None
        nodes[u] = fh.Node((cost[u], u))
        queue.insert(nodes[u])
    while len(mst) < n:
        u: int = queue.extract_min().key[1]
        if u != start:
            mst[u].append((parent[u], cost[u]))
            mst[parent[u]].append((u, cost[u]))
        v: int; w: float
        for v in range(n):
            w = mat[u][v]
            if v not in mst and w < cost[v]:
                cost[v] = w
                parent[v] = u
                queue.decrease_key(nodes[v], (cost[v], v))
    return mst
def min_weight_perfect_matching(mat: list[list[any]], odds: list[any]) -> list[tuple[any,any,float]]:
    random.shuffle(odds)
    matching: list[tuple[any,any,float]] = []
    while odds:
        u: any = odds.pop()
        closest: any = None
        min_dist: float = float('inf')
        for v in odds:
            if u != v and mat[u][v] < min_dist:
                min_dist = mat[u][v]
                closest = v
        matching.append((u, closest, min_dist))
        odds.remove(closest)
    return matching
def dfs(graph: dict[any,list[tuple[any,float]]], vertex: any, deg: dict[any,int], output: list[any], edges:
dict[tuple[any,any],int]) -> None:
    while deg[vertex] > 0:
        deg[vertex] -= 1
        neighbor: any = graph[vertex][deg[vertex]][0]
        if edges[tuple(sorted((vertex, neighbor)))] > 0:
            if vertex == neighbor:
                edges[tuple(sorted((vertex, neighbor)))] -= 1
            else:
                edges[tuple(sorted((vertex, neighbor)))] -= 2
        dfs(graph, neighbor, deg, output, edges)
    output.append(vertex)
def hierholzer(graph: dict[any,list[tuple[any,float]]]) -> list[any]:
    deg: dict[any,int] = {u: len(graph[u]) for u in graph}
    edges: dict[tuple[any,any],int] = defaultdict(lambda: 0)

```

```

for u in graph:
    for v, w in graph[u]:
        edges[tuple(sorted((u, v)))] += 1
start: any = 0
output: list[any] = []
dfs(graph, start, deg, output, edges)
output.reverse()
return output
def shortcutting(circuit: list[any]) -> list[any]:
    visited = set()
    output = []
    for vertex in circuit:
        if vertex not in visited:
            output.append(vertex)
            visited.add(vertex)
    output.append(circuit[0])
    return output
def tsp_christofides(mat: list[list[float]]) -> tuple[float, list[int]]:
    mst: dict[int, list[tuple[int, float]]] = prim(mat)
    odds: list[int] = [u for u in mst if len(mst[u]) % 2 == 1]
    matching: list[tuple[int, int, float]] = min_weight_perfect_matching(mat, odds)
    u: int; v: int; w: float
    for u, v, w in matching:
        mst[u].append((v, w))
        mst[v].append((u, w))
    eulerian: list[int] = hierholzer(mst)
    tour: list[int] = shortcutting(eulerian)
    cost: float = 0
    for i in range(len(tour)-1):
        cost += mat[tour[i]][tour[i+1]]
    return cost, tour
if __name__ == '__main__':
    Data1=[(-6,4823,34,454211), (-6,481987,34,454207), (-6,482039,34,453912),\
(-6,482177,34,453607), (-6,482252,34,453582), (-6,482657,34,453746),\
(-6,482735,34,453769), (-6,484358,34,453892), (-6,481701,34,453756),\
(-6,481593,34,453762)]
    graph : list[list[float]] = [[acos(sin(radians(y2)) * sin(radians(y1)) + cos(radians(y2)) * cos(radians(y1))) *
cos(radians(x2)- radians(x1))) * 6371 *1000 for x2, y2 in Data1] for x1, y1 in Data1]
    cost, tour = tsp_christofides(graph)
    print(f'Produced tour: {tour} (cost = {cost})')

```

Result

Produced tour: [0, 7, 6, 5, 4, 3, 2, 9, 8, 1, 0] (cost = 573,4039979456204 meters)

Projection the result

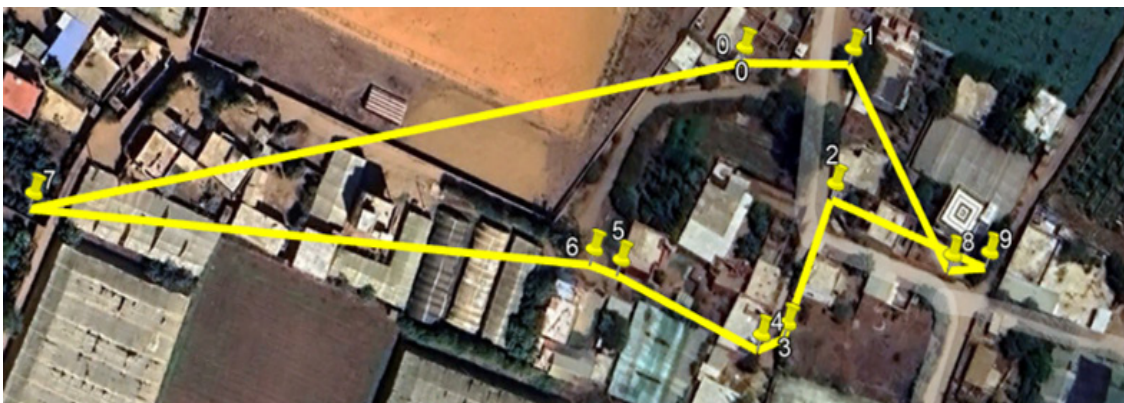


Figure 16. Projection christofides algorithm

RESULTS AND DISCUSSION

Synthesis

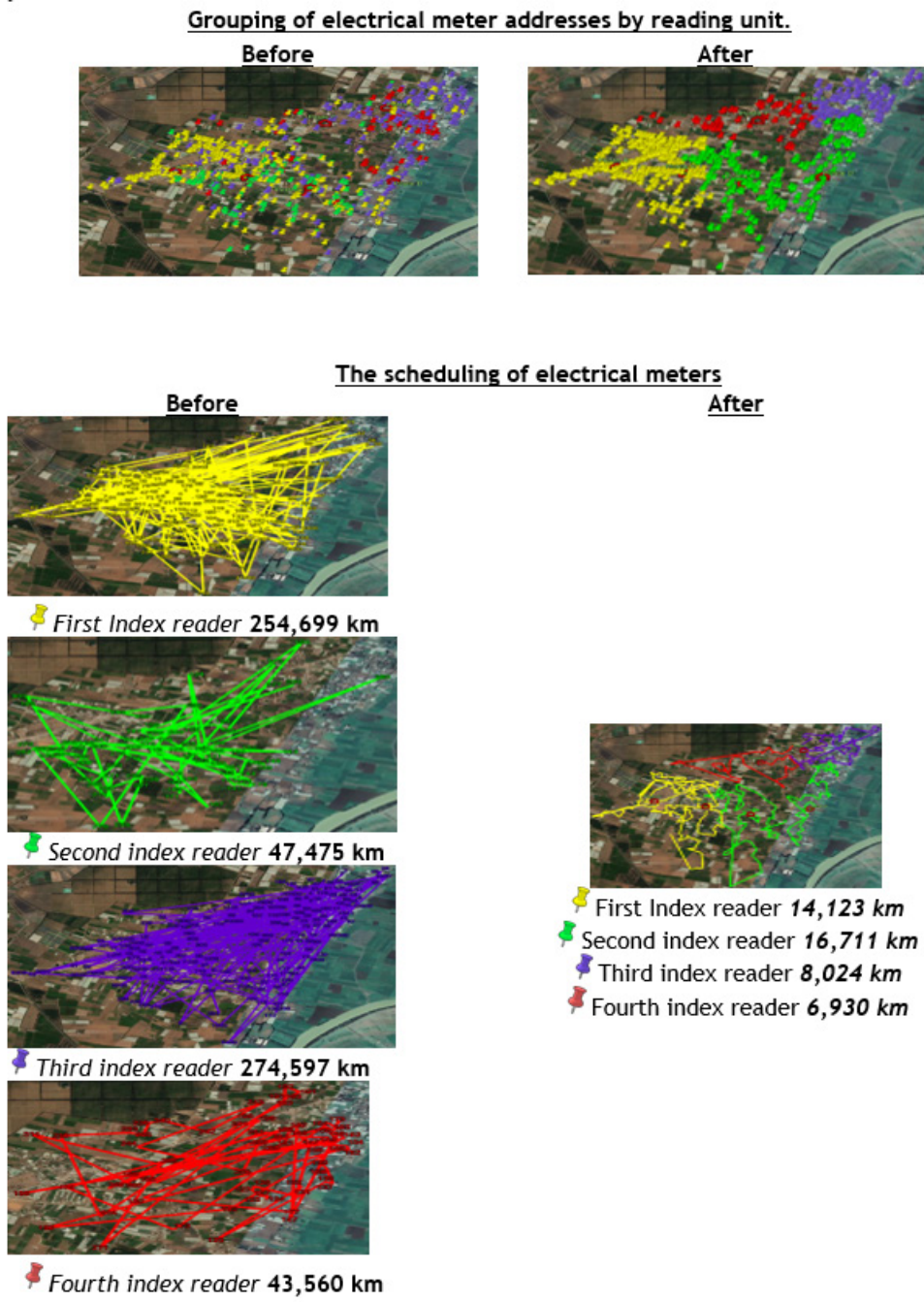


Figure 17. Synthesis of results

CONCLUSIONS

Improving customer service while reducing operating costs and increasing energy sales is a significant challenge in the contemporary energy sector. This study has shown that these objectives are compatible and can be achieved through integrated strategies focused on customer satisfaction and operational efficiency. Companies that take a proactive approach, using advanced technologies and optimized management processes, can achieve significant savings while enhancing the user experience.

The analysis of the results obtained during this research highlights the importance of a synergy between optimized customer service, a significant reduction in expenses, and an increase in energy sales. First, the satisfaction of electricity consumers begins with a regular survey of the electricity meter indexes for actual billing that does not exceed the consumer’s expectations and an emergency response following the electrical incident to ensure continuity of power supply.

All tasks performed by the electrical power distributor require a move to the point of consumption, and knowing that rural addresses are not arranged, it was not easy to locate electrical installations. As a result, operating expenses have not ceased to increase, especially with new customers that index readers or technicians do not yet know their location. This situation has penalized customers by the cumulating of consumption and has led non-payers to terminate their subscription contracts. All this has tarnished the energy distributor's brand image and reduced electric power sales.

Finally, this work of locating the electric meters using the GPS brand Garmin and their scheduling with optimization algorithms, mainly that of the nearest neighbors, allowed us to satisfy customers by conducting periodic surveys and actual invoicing to reduce the index reading time and the response time following an incident.

The work we did allowed us to achieve our goal, which was: improve customer service quality, reduce operating expenses, and improve energy sales by the K-MEANS method and optimization algorithms.

BIBLIOGRAPHIC REFERENCES

1. Yijun Wang. "Interactive display method of electric power business hall based on 3D technology". Computers and Electrical Engineering 2024.
2. Philip Ramsey. "Providing for Occupant Experience in Optimized Connected Energy Communities - A Systematic and Critical Review". Building and Environment 2024.
3. João Tabanêz Patrício, R. Lopes, N. Majdalani, D. Aelenei, João Martins. "Aggregated Use of Energy Flexibility in Office Buildings". Energies 2023.
4. L. Varga. "Guest editorial (17.2) OE Emergence: Complexity and Organization" 2019.
5. Alam, Mohammad Jahidul. Multiple Drone and Truck Arc Routing Problem 2022.
6. Corberán and al, Arc routing problems: A review of the past, present, and future 2021.
7. Reula Martín, Miguel. Contributions to Close-Enough Arc Routing Problems 2021.
8. BOSTEL, Nathalie, HA, Minh Hoang. (2012).
9. P. Muhlmeister, International Steve Fine. "The Locational Value of Energy Efficiency on the Distribution Grid" 2016.
10. Dr. Arathi R Shankar. "Measurement of power quality disturbances" 2015.
11. Diego A. Zaldivar, Andres A. Romero and Sergio R. Rivera, Risk Assessment Algorithm for Power Transformer Fleets Based on Condition and Strategic Importance, Algorithms 2021.
12. A Khazaei M, Ghasemzadeh. Comparing k-means clusters on parallel Persian-English corpus. 'International Digital Organization for Scientific Information 2015.
13. Boghey, Rajesh, Khan, Ali Z, Singh, Manjari. IMPROVEMENT OF DATA ANALYSIS BASED ON K-MEANS ALGORITHM AND AKMCA (2023).
14. Martin Gjoreski and al, "A Method for Generating Alternatives for Hierarchical Multi-Attribute Decision Models Using Bayesian Optimization", Algorithms 2022
15. A Agra and al., "Algorithm Engineering in Robust Optimization" 2015.
16. Bandoni and al., "Diseño para operabilidad: Una revisión de enfoques y estrategias de solución" 2004.
17. Schäfer and al., "Component-based synthesis of motion planning algorithms" 2021.
18. Fainekos and al., "Revision of Specification Automata under Quantitative Preferences" 2014.
19. Raymond and al., "Maximum common subgraph isomorphism algorithms for the matching of chemical structures" 2002.

20. Bast and al., “Route Planning in Transportation Networks” 2015.
21. Kinable and al., “JGraphT -- A Java library for graph data structures and algorithms” 2020.
22. Aldous D Fill J and al., “Networking - A Statistical Physics Perspective” 2012.
23. Halperin and al., “Asymptotically near-optimal RRT for fast, high-quality, motion planning” 2015.
24. An and al., “Improving Christofides’ Algorithm for the s-t Path TSP” .2011.
25. Sebö and al., “Improving Christofides’ Algorithm for the s-t Path TSP” 2012.

FINANCING

The authors did not receive financing for the development of this research.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

AUTHORSHIP CONTRIBUTION

Conceptualization: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Data curation: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Formal analysis: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Research: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Methodology: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Supervision: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Validation: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Drafting - original draft: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.

Writing - proofreading and editing: Redouane Touil, Rachid Marrakh, Taoufiq Belhoussine Drissi, Bahloul Bensassi.