People's Democratic Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and Scientific Research
وزارة التعليم العالي و البحث العلمي

جامعة قاصدي مرباح ورقلة
Kasdi Merbah University of Ouargla

Academic Master Thesis

to obtain a master's degree in Computer Science

Major : Industrial Computing

Comparative study of Vehicle Routing Problem' solving methods

Realized by :
Zeineb Chaabena
Manel Khechiba

Supervised by :
Dr. Mourad Belhadj
(UKMO)

Presented in June 2023, in front of the jury composed of :

Dr.Abdelhakim Cheriet :	UKMO	- President
Dr. Meriem Khelifa :	UKMO	- Examiner

Promotion : 2022/2023

Dedication

“

To our parents

”

- Zeineb and Manel

Acknowledgment

First of all, we thank our Almighty God who helped us and gave us the patience and courage during our years of study and to achieve this work. we also thank our parents, who encouraged and motivated us to reach this level of study.

Our sincere thanks also go to our supervisor **Dr. Belhadj Mourad** for the continuous support, guidance, and motivation to us all the time for solving our research problems. And a special thanks for his confidence in us.

Likewise, we extend our respectful thanks to the members of the jury. The president **Dr. Abdelhakim Cheriet** and the examiner **Dr.Meriem Khelifa** , who has done the honor to participate in this jury and to examine this work.

we would be remiss in not mentioning all my teachers for their guidance in the last five years in Kasdi Merbah Ouargla University's Department of Information Technology.

A special thanks to our family members and friends for always standing by us. Finally, we would like to thank all those who have helped us from near or far during all our studies and in the preparation of this dissertation, our deep gratitude and respect.

Abstract

The decision-making process in distribution operations has a significant impact on minimizing distance, time, and costs, making it a crucial topic in the field of logistics. Efficient vehicle routing is essential in scenarios such as school bus routes, pizza delivery, and distribution of goods. The Vehicle Routing Problem (VRP) represents the core challenge we aim to address and solve. Understanding the complexities and nuances of VRP allows us to develop strategies and methodologies to optimize routing decisions and achieve minimal distances, reduced time, and lowered costs.

The objective of this dissertation is to conduct a comprehensive study of methods to solve the Vehicle Routing Problem (VRP) and evaluate their effectiveness and suitability.

In this dissertation, we employ a mixed-methods approach. Firstly, we conduct a literature review to understand the definition and characteristics of VRP and gather existing methodologies. We then develop a methodology by creating a dataset and modifying existing codes. These modifications aim to address specific VRP requirements and optimize route planning, resource utilization, and cost efficiency. The proposed methodology serves as a practical approach to solving VRP. For the experimental part, we evaluate the effectiveness of ACO, NN algorithm, and PABCW using benchmark datasets provided by [9] and a real-world case. We compare the solutions obtained by these algorithms across different instances and evaluate their performance.

The comparative study reveals that the modified version of the k-Nearest Neighbors (kNN) algorithm achieves the highest accuracy percentage among NN, ACO, and PABCW. The evaluation based on benchmark datasets and a real-world case demonstrates the effectiveness of the modified kNN algorithm in solving the VRP. However, further evaluation and validation are necessary to confirm its effectiveness and robustness in real-world applications, considering potential cost savings and improved operational efficiency.

This dissertation provides valuable insights into the performance and suitability of different algorithms for addressing the Capacitated Vehicle Routing Problem. The study highlights the significance of efficient vehicle routing in minimizing distance, time, and costs in distribution operations. The developed methodology, based on a modified kNN algorithm, demonstrates promising results in solving the VRP. Future research should

focus on further evaluation and validation of the algorithm in real-world applications, considering the potential for cost savings and improved operational efficiency.

Keywords : Vehicle Routing Problem, k-Nearest Neighbour, Ant Colony Optimization, Proposed Algorithm, benchmark, dataset.

Résumé

Le processus de prise de décision dans les opérations de distribution a un impact significatif sur la minimisation de la distance, du temps et des coûts, ce qui en fait un sujet crucial dans le domaine de la logistique. Un routage efficace des véhicules est essentiel dans des scénarios tels que les itinéraires des bus scolaires, la livraison de pizzas et la distribution de marchandises. Le Problème de Routage de Véhicules (VRP) représente le défi central que nous cherchons à résoudre. Comprendre les complexités et les nuances du VRP nous permet de développer des stratégies et des méthodologies pour optimiser les décisions de routage et atteindre des distances optimales, un temps réduit et des coûts réduits.

L'objectif de ce travail est de réaliser une étude approfondie des méthodes de résolution du Problème de Routage de Véhicules (VRP) et d'évaluer leur efficacité et leur pertinence, en plus d'appliquer ses méthodes sur des cas d'utilisation réels.

Dans cette dissertation, nous utilisons une approche mixte. Tout d'abord, nous réalisons une revue de littérature pour comprendre la définition et les caractéristiques du VRP et recueillir les méthodologies existantes. Nous développons ensuite une méthodologie en créant un ensemble de données et en modifiant des codes existants. Ces modifications visent à répondre aux exigences spécifiques du VRP et à optimiser la planification des itinéraires, l'utilisation des ressources et l'efficacité des coûts. La méthodologie proposée sert d'approche pratique pour résoudre le VRP. Pour la partie expérimentale, nous évaluons l'efficacité de l'ACO, de l'algorithme NN et de l'PABCW en utilisant des ensembles de données de référence fournis par [9] ainsi qu'un cas réel. Nous comparons les solutions obtenues par ces algorithmes sur différentes instances et évaluons leurs performances.

L'étude comparative révèle que la version modifiée de l'algorithme des k plus proches voisins (kNN) atteint le pourcentage de précision le plus élevé parmi NN, ACO et PABCW. L'évaluation basée sur des ensembles de données de référence et un cas réel démontre l'efficacité de l'algorithme kNN modifié dans la résolution du VRP. Cependant, des évaluations et des validations supplémentaires sont nécessaires pour confirmer son efficacité et sa robustesse dans des applications du monde réel, en tenant compte des économies potentielles de coûts et de l'amélioration de l'efficacité opérationnelle.

En conclusion, ce modeste travail offre des perspectives précieuses sur les performances et la pertinence de différentes algorithmes pour aborder le Problème de Routage de Véhicules à Capacité Limitée. L'étude met en évidence l'importance d'un routage efficace des véhicules dans la minimisation de la distance, du temps et des coûts dans les opérations de distribution. La méthodologie développée, basée sur un algorithme kNN modifié, démontre des résultats prometteurs dans la résolution du VRP. Les futures recherches devraient se concentrer sur une évaluation et une validation supplémentaires de l'algorithme dans des applications du monde réel, en tenant compte du potentiel d'économies de coûts et de l'amélioration de l'efficacité opérationnelle.

Mots clés : Problème de Routage de Véhicules, k plus proches voisins, Optimisation par Colonie de Fourmis, Algorithme Proposée, référence, ensemble de données.

ملخص

عملية اتخاذ القرار في عمليات التوزيع لها تأثير كبير في تقليل المسافة والوقت والتكاليف، مما يجعلها موضوعاً حاسماً في مجال الخدمات اللوجستية. يعد التوجيه الفعال للمركبات ضرورياً في سيناريوهات مثل مسارات حافلات المدارس وتوصيل البيتزا وتوزيع السلع. تمثل مشكلة توجيه المركبات التحدي الأساسي الذي نهدف إلى معالجته وحله. فهم تعقيدات وجوانب مشكلة توجيه المركبات يسمح لنا بتطوير استراتيجيات ومنهجيات لتحسين قرارات التوجيه وتحقيق أدنى مسافات وتقليل الوقت وتخفيض التكاليف.

هدف هذه الدراسة هو إجراء دراسة شاملة للأساليب التي تحل مشكلة توجيه المركبات وتقييم فعاليتها وملاءمتها.

في هذه الدراسة، نستخدم نهجًا يعتمد على تنوع الطرق. أولاً، نقوم بمراجعة الأدبيات لفهم تعريف وخصائص مشكلة توجيه المركبات وجمع الأساليب الموجودة بالفعل. ثم نقوم بتطوير منهجية من خلال إنشاء مجموعة بيانات وتعديل رموز موجودة نحصل عليها من مصادر موثوقة. تهدف هذه التعديلات إلى معالجة متطلبات محددة لمشكلة توجيه المركبات وتحسين تخطيط الطرق واستخدام الموارد وكفاءة التكلفة. تعد منهجية المقترح نهجًا عمليًا لحل هذه المشكلة. بالنسبة للجزء التجريبي، نقوم بتقييم فعالية خوارزمية النمل المستعرض، وخوارزمية الجار الأقرب، والطريقة المقترحة باستخدام مجموعة بيانات المرجع التي تم توفيرها من قبل taregua وحالة حقيقية. نقارن الحلول التي تم الحصول عليها بواسطة هذه الخوارزميات عبر مختلف الحالات ونقيم أدائها

يكشف الدراسة المقارنة أن النسخة المعدلة من خوارزمية أقرب الجيران تحقق أعلى نسبة دقة بين أقرب الجيران، وأمثلة تحسين النمل، والأسلوب المقترح. يوضح التقييم القائم على مجموعة بيانات الاختبار القياسية وحالة عمل حقيقية فعالية الخوارزمية المعدلة لأقرب الجيران في حل المشكلة اللوجستية للمركبات. ومع ذلك، يجب إجراء تقييم وتحقق إضافي للتأكد من فعالية الخوارزمية وقوتها في تطبيقات العالم الحقيقي، مع النظر في التوفير في التكاليف المحتملة وتحسين الكفاءة التشغيلية.

في الختام، يوفر هذا البحث إدراكًا قيمًا حول أداء وملاءمة الخوارزميات المختلفة للتعامل مع مشكلة التوزيع للمركبات ذات السعة المحدودة. يسلط البحث الضوء على أهمية توجيه السيارات الفعال في تقليل المسافة والوقت والتكاليف في عمليات التوزيع. تظهر المنهجية المطورة، المستندة إلى خوارزمية أقرب الجيران المعدلة، نتائج واعدة في حل مشكلة توزيع المركبات. ينبغي أن يركز البحث المستقبلي على تقييم وتحقق إضافي للخوارزمية في تطبيقات العالم الحقيقي، مع النظر في إمكانية توفير التكاليف وتحسين الكفاءة التشغيلية.

كلمات مفتاحية :

مشكلة توزيع المركبات، أقرب الجيران، تحسين النمل، الخوارزمية المقترحة، مجموعة بيانات الاختبار القياسية، مجموعة البيانات.

Contents

List of Figures

List of Tables

List of Algorithms

List of abbreviations

VRP	*Vehicle Routing Probelm*
TSP	*Traveling Salesman Problem*
VRPTW	*Vehicle Routing Problem with Time Windows*
CVRP	*Capacitated Vehicle Routing Problem*
SWOT	*Strenght / Weakness / Opportunities / Threats*
k-NN	*The k-Nearest Neighbors*
GPHH	*Genetic Programming Hyper-heuristic*
UCARP	*Incapacitated Arc Routing Problem*
EAs	*Evolution based Algorithms*
GA	*genetic algorithm*
SI	*Swarm intelligence*
PSO	*Particle Swarm Optimization*
ACO	*Ant Colony Optimization*
PAs	*Physics-based algorithms*
SA	*Simulated Annealing*
HAs	*Human-based metaheuristic algorithms*
TS	*Tabu search*

LS *Local search*

TSP *Traveling Salesman Problem*

VRP *Vehicle Routing Problem*

GIS *geographic information system*

DE *Differential Evolution*

ABC *Artificial Bee Colony*

CS *Cuckoo Search*

NN *Nearest Neighbor*

PABCW *Proposed Algorithm Based on Clarke-wright*

General introduction

The optimization of vehicle routing plays a crucial role in various industries, including transportation, logistics, and delivery services. Efficiently planning routes for vehicles can lead to significant cost savings, improved resource utilization, and enhanced customer satisfaction. The Vehicle Routing Problem (VRP) is a complex optimization problem that involves determining the most optimal routes for a fleet of vehicles to serve a set of customers while satisfying various constraints. This dissertation aims to address the challenges associated with VRP and propose effective algorithms to solve them.

VRP has been extensively studied in operations research and has garnered considerable attention due to its practical applications. The problem encompasses various types, such as the Capacitated VRP, Vehicle Routing Problem with Time Windows (VRPTW), and the Multiple Depot VRP, each with unique characteristics and constraints. Understanding the different types of VRP and their applications in industries is crucial for developing appropriate algorithms and solutions.

A comprehensive literature review will be conducted to explore the existing research and solutions in the field of VRP. This review will cover traditional approaches and state-of-the-art techniques, including heuristic and metaheuristic algorithms. By analyzing the strengths and limitations of these approaches, this dissertation aims to identify gaps in the literature and propose novel modifications to existing algorithms, as well as a new algorithm specifically tailored for VRP.

The primary objective of this dissertation is to conduct a comparative study of three algorithms for solving VRP. Two of the algorithms will be modified to adapt to VRP based on the characteristics of the provided dataset. In contrast, the third algorithm will be a proposed approach designed to address the limitations of existing methods. The study aims to evaluate and compare the performance of these algorithms in terms of solution quality, computational efficiency, and scalability.

The methodology employed in this research will involve several steps. Firstly, a comprehensive dataset will be collected, which will include details such as customer locations, vehicle capacities, delivery time windows, and distance constraints. The adapted versions of the Ant Colony Optimization (ACO) and K-Nearest Neighbor (KNN) algorithms will

be developed, considering the specific requirements of VRP. The proposed algorithm will combine elements from both ACO and KNN to devise a novel approach to solving VRP.

This dissertation will be structured into several chapters to ensure a logical flow of information. Chapter 1 will provide an introduction to VRP, its applications in various industries, and the need for efficient algorithms to solve these optimization problems. Chapter 2 will present an overview of heuristic and metaheuristic algorithms, discussing their relevance in solving VRP. Chapter 3 will focus on the methodology, explaining the dataset used, the adaptations made to ACO and KNN algorithms, and the proposed algorithm. Chapter 4 will present the comparative analysis of the three algorithms based on benchmark datasets and the provided dataset. Finally, We will summarize the findings, conclude, and provide recommendations for future research.

In conclusion, this dissertation aims to contribute to the field of VRP by conducting a comparative study of three algorithms for solving this challenging optimization problem. By proposing adaptations to existing algorithms and introducing a novel approach, the research aims to provide valuable insights into the performance and suitability of different algorithms in real-world VRP scenarios. The findings of this study can help industries optimize their vehicle routing processes, leading to improved efficiency, reduced costs, and enhanced customer satisfaction.

Chapter 1

Vehicle Routing Problem

1.1 Introduction

The decision-making process in distribution operations significantly impacts minimizing distance, time, and costs, making it a crucial topic in logistics. Examples of such decision-making scenarios include school bus routes for student transportation, pizza delivery routes, and the distribution of goods from warehouses to multiple retail outlets. These scenarios affect vehicle routing to ensure optimal use of resources and minimize associated expenses. In this chapter, we will focus on the Vehicle Routing Problem (VRP) and its characteristics. VRP represents the core challenge that we seek to address and solve. By examining the complexities and nuances of VRP, we can develop strategies and methodologies to optimize routing decisions and achieve minimal distances, reduced time, and lowered costs. Through a comprehensive exploration of VRP in this chapter, we will gain insights into its different variations, key challenges, and potential solutions.

1.2 Definition and Characteristics of Vehicle Routing Problem

Since its introduction by Dantzig and Ramser, the vehicle routing problem (VRP), also called Capacitated Vehicle Routing Problem (CVRP difficulty), has attracted significant attention from researchers due to its practicality and inherent difficulty[20]. VRP depots encompass a broad range of problems involving the optimal visitation of multiple customers by a given number of vehicles from one or more depots[18]. The primary objective of VRP is to determine the optimal route for each vehicle while minimizing total distance or total costs and adhering to specified constraints. The key components of VRP include depots, vehicles, customers, road networks, and drivers. The combinations and arrangements of these components give rise to numerous variations of VRP. Before exploring the different variants, let us briefly define and characterize the main components:

- Depots: Depots serve as the starting and ending points for the VRP. In some cases, multiple depots may exist.
- Customer: Customers represent the entities that must be serviced in the VRP. They are located around the depots and may have deterministic or stochastic demands.
- Vehicles: VRP involves a predetermined number of vehicles, each with specific characteristics such as maximum travel time, capacity, cost, or time constraints (The Traveling Salesman Problem emerges when there is just one car).
- Routes: Routes refer to the paths that vehicles take to serve customers. These routes can have varying costs, travel times and can be one-way or two-way.

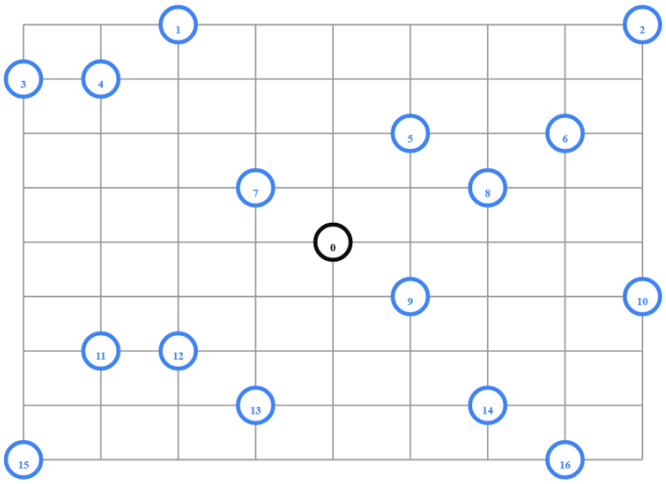


Figure 1.1: Customers and depots
[62]

Understanding these fundamental components of VRP and considering the specified conditions is crucial for developing effective strategies and algorithms to optimize the routing process, ultimately minimizing costs and improving efficiency.

It is worth noting that the vehicle routing problem is classified as NP-hard [6], meaning there is currently no efficient solution to this problem. While it can be precisely solved for a limited number of occurrences, finding the best routes for several cars traveling to a set of places introduces additional complexities. For instance, the traveling salesman problem emerges when only one car is involved.

In the subsequent sections, we will delve into different variants of VRP, exploring their unique characteristics and challenges. Additionally, we will discuss the specific considerations and adaptations required for various industries and domains where VRP finds application.

1.3 Vehicle Routing Problem's history

Dantzig and Ramser, in 1959, first publicly presented the CVRP. For its resolution, these authors suggested a straightforward matching-based heuristic and demonstrated it using a miniature example. Heuristics based on various concepts, such as cost savings, closeness to customers' locations, customer matching, and intra- and inter-route optimization steps, began to develop in the following years. The Clarke and Wright 1964 savings heuristic, which has survived the test of time due to its speed, simplicity, and moderate accuracy, is possibly the most well-known heuristic in this category. After two studies by Christofides, Mingozzi, and Toth were published in Networks by Christofides [46], the development of precise algorithms for the VRP took off.

While the second study presented two mathematical formulations using q-paths and k-shortest spanning trees, the first publication proposed an approach based on dynamic programming with state-space relaxation. A few years later, the first cutting plane technique for a VRP was put out by Laporte, Desrochers, and Nobert[45], based on the linear relaxation solution of an integer model. Some of the more modern algorithms have included these fundamental ideas.

Since then, several precise algorithms built on formulations from mathematical programming have been released. Some formulations, which frequently need branch-and-cut solutions, include variables for vehicle flow or commodity flow. The VRP may be expressed as a set partitioning problem with some extra legitimate inequalities. This concept has been used in some of the most effective implementations by Baldacci [10]and Fukasawa [23].

They typically use many operators, as in adaptive extensive neighborhood search [51] or combine genetic search with local search, as in the recently suggested hybrid genetic algorithm by [46].

However, The current VRP models vastly differ from those introduced by Dantzig and Ramser, Clarke, and Wright, as they increasingly aim to incorporate real-life complexities. For example, time-dependent travel times (reflecting traffic congestion), time windows for pickup and delivery, and input information (e.g., demand information) that changes dynamically over time are just a few examples. These features bring along substantial complexity[21].

1.4 Difference between Vehicle Routing Problem and Travelling Salesman Problem

The Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) are central to distribution management and have attracted the attention of researchers for more than 50 years.

The traveling salesman problem (TSP) has the same objective as the VRP. The TSP aims to find the shortest optimal route regarding distance, time, or cost for a traveling salesman [13]. The traveling salesman has to start from one depot, visit each city once, and return to the beginning point, the depot, in one single closed tour. The difference between the two problems is that VRP has more than one vehicle, while the TSP has only one vehicle. Both have the same goal which is either traveling the least total distance or minimizing the total cost.

Listing all the possible tours and choosing the one with the lowest cost may sound

logical and correct. However, in the TSP, the number of tours grows exponentially. So, the possible number of solutions is n! for n cities [47].

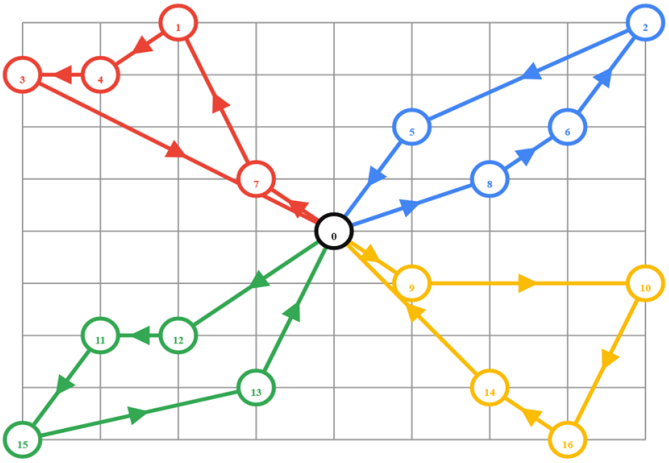


Figure 1.2: Illustration of VRP
[62]

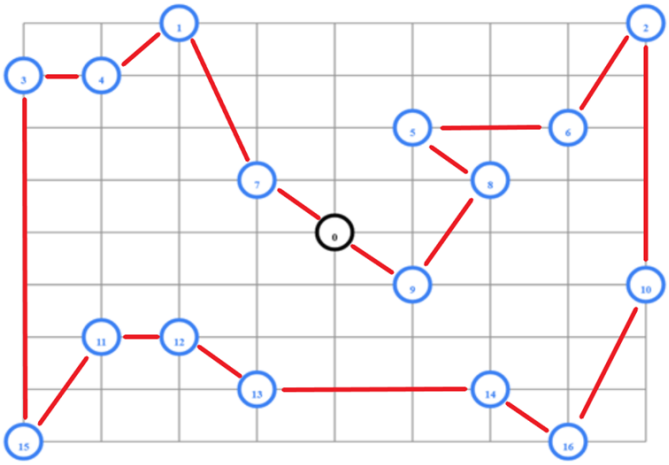


Figure 1.3: Illustration of TSP

1.5 Mathematical Model

Let $G = (V, A)$ be a complete and directed graph, where $V = 0, 1, \ldots, n$ is the set of nodes and A is the set of arcs. The nodes $i = 1, 2, .., n$ correspond to customers, each with a deterministic demand $d_i \geq 0$.

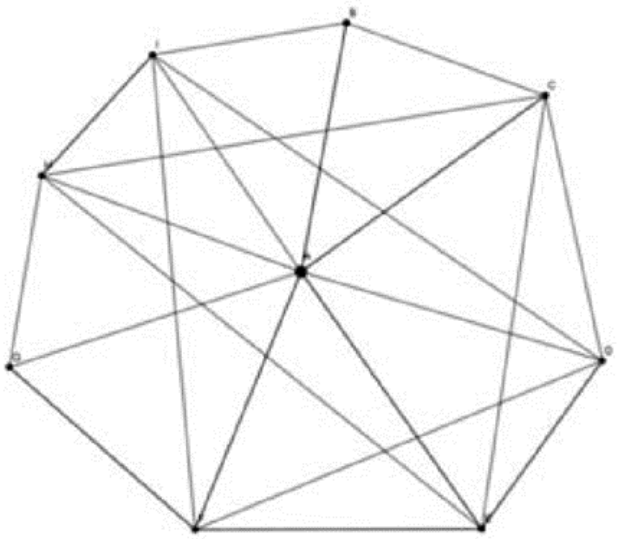


Figure 1.4: The graph model of VRP

The node $i = 0$ is the zero-demand depot, *i.e.*, $d_0 = 0$. Let $m > 0$ be the number of different types of vehicles. For the k^{th} type of vehicles($k = 1, 2, \ldots, m$), let $V_k = \{1, 2, \ldots, v_k\}$ be the set of vehicles k, with load capacity $Q_k > 0$. Let c_{ij} be the non-negative distance-based cost associated with the arc $(i, j) \in A$, and $c_{ij} = + \propto, \forall_I \in V$. In the case of calculating the actual distance between pairs of locations (i, j), the distance matrix may be asymmetric, *i.e.*, it can happen that $c_{ij} \neq c_{ji}$.

The VRP goal is to find the minimum distance or cost roundtrip routes starting from and ending at the depot, which satisfy all customers' demands, visit each customer only once, and do not exceed the load capacity of each type of vehicle.

The following expression gives the objective function to be minimized:

$$Min \sum_{\forall_I \in V} \sum_{\forall_J \in V} c_{ji} \sum_{k=1}^{m} x_{ijk}$$

Subject to:

$$\sum_{\forall_I \in V \setminus \{0\}} x_{i0k} = \sum_{\forall_J \in V \setminus \{0,i\}} x_{j0k}, \ \forall k \in \ \{1, , m\} \tag{1.1}$$

$$\sum_{k=0}^{m} \sum_{\forall_i \in V} x_{j0k} = 1, \ \forall j \in \ V \setminus \{0\} \tag{1.2}$$

$$\sum_{\forall_I \in V \setminus \{h\}} x_{ihk} = \sum_{\forall_J \in V \setminus \{h,i\}} x_{jhk}, \ \forall h \in v \setminus \{0\}, \ \forall k \in \{1, , m\} \tag{1.3}$$

$$\sum_{\forall_J \in V \setminus \{0\}} x_{0jk} \leqslant v_k, \ \forall k \in \{1, , m\} \tag{1.4}$$

$$\sum_{\forall_I \in V} y_{ij} + d_j = \sum_{\forall_I \in V} y_{ji}, \ \forall j \in v \setminus \{0\} \tag{1.5}$$

$$0 \leqslant d_i x_{ijk} \leqslant y_{ij} \leqslant (Q_K - J) x_{ijk}, \ \forall (i,j) \in v \setminus \{0\}, \ \forall k \in \ \{1, , m\} \tag{1.6}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in \ v \setminus \{0\}, \ \ \forall k \in \ \{1, , m\} \tag{1.7}$$

In this formulation, for both symmetrical and asymmetrical issues and for both homogeneous and heterogeneous fleets, $O(n_K^2)$ binary variables x are used. The binary variable x_{ijk} in 1.7 indicates whether or not the arc $(i,j) \leqslant A$ is traveled by a vehicle of type $k(k = 1, 2, \ldots, m)$. In addition, there are $O(nK)$ variables y where y_{ij} represents the load in the truck arriving at customer j after visiting customer i in terms of units of commodity

The objective function is minimizing the total distance-based cost of the arcs used by all m routes generated.

Constraint (1.1) implies that the number of vehicles of each type leaving the depot is the same as the number of vehicles of that type returning to it.

Constraints (1.2) and (1.3) require that each customer is visited exactly once and that the same type of vehicle k arrives and leaves each h customer location, respectively.

Constraint (1.4) imposes that the number of used vehicles of each type does not exceed the number of available vehicles of that type.

Constraint (1.5) states that the quantity of products y_{ij} in the truck leaving customer j plus the demand of that customer equals the number of products in the truck leaving it after the service has been completed.

Constraint (1.6) guarantees lower and upper bounds ensuring that: the quantity of products y_{ij} in the truck leaving customer i is equal to or greater than its demand, d_i; and the total demand served by each vehicle k does not exceed the service capacity Q_k [32].

1.6 SWOT analysis

To analyze the vehicle routing problem, it is important to examine its strengths, weaknesses, opportunities, and threats.

- Strengths: The VRP has many great qualities, for instance, one of its strengths is finding the best route for transportation companies, which helped to reduce transportation costs, including vehicle costs, driver wages, and penalties for late deliveries, ultimately leading to the satisfaction of customers [16].
- Weaknesses: The VRP contains flaws that need to be fixed. Only small-scale issues can be solved, and finding a workable solution takes time during the execution and running of the solvers. Additionally, it might not work well in real-world situations. In other words, although the outcomes were great, they were never perfect [16].
- Opportunities: Include creating new software to address the issue and produce precise answers as well as enhancing an existing program while taking into account practical limitations [16].
- Threats: Threats to the validity of the optimality of the solution that may lower its effectiveness include changes in the weather, the mechanical condition of the cars, and traffic [16].

1.7 Vehicle Routing Problem' Types

VRP issues may be categorized by constructing a taxonomy or a generalized framework that enumerates the models now in use, the goals pursued, and the theories related to the investigation of the issue, the following table 1.1shows the most popular VRP's types [2]:

Name	Description
Node components	
Traveling Salesman Problem	A single agent (vehicle) visits the cities (customers). The vehicle makes a trip. The objective is to minimize the distance the vehicle travels. Cities (customers) have no product demand
VRP with Single Depot	The vehicles leave a common depot, visit the customers (deliver products to the customers) and then return to the depot (after the customers have visited)
Vehicle Component	
Electric VRP	Electric vehicles travel, the vehicles visit recharger stations after a certain distance
Fuel Efficient Green VRP	The objective is to minimize fuel emissions from vehicles
Time Component	
VRP with Time windows	Customers may have different time windows. The product demands of the customers must be served within the time window. The customer can have single or multiple time windows
Periodic VRP	Customers do not have to be visited once, but periodically, even several times within a (predefined) period
Functional parameter component	
Open VRP	One or more depots are in the system from which vehicles departed to visit customers. However, the vehicles after visiting the customers do not return to the depot
Multi-Depot VRP with Pickup and Delivery	One or more depots are in the system from which vehicles departed to visit customers. Once the customers are visited, vehicles can return to any depot.

Table 1.1: The main types of VRP
[2]

1.8 Applications

Olli Bäysy and Geir Hasle conducted a thorough investigation of the usage and relevance of vehicle routing in several global companies and sectors. They came to the conclusion that there are some daily tasks where using Vehicle Routing solutions can make them a little less difficult [60]

- Waste collection: In situations like this, when it may be used often, vehicle routing can be quite helpful. Simply said, waste collection is VRP with several distribution points or nodes where trash has to be gathered. Giving them an ordered route without leaving behind a node or by preventing disturbance may be quite helpful in instances like these [41] [35]. The itineraries of five garbage collection trucks are shown in Figure 06 below, together with the number of stops they will make and the overall distance they will travel.

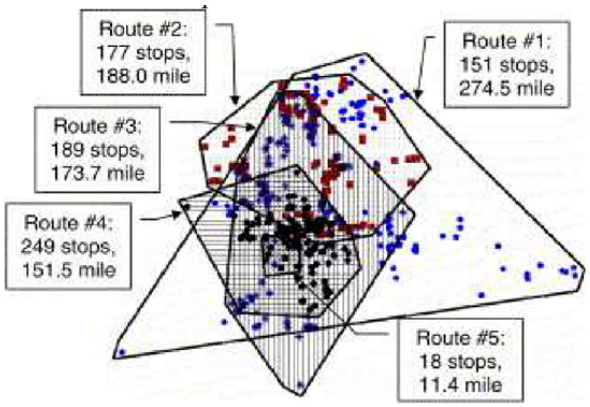


Figure 1.5: VRP (Waste collection)
[57]

Real-world applications typically include a variety of factors and limitations that simple VRP can not always account for, leading to the addition of restrictions like capacity, time frames, etc. Additionally, [60] conducted research on several VRP uses, such as disaster relief [60].

- Primary healthcare services: In the event of an unplanned calamity, such as a natural disaster or a terrorist attack, there might be significant destruction, loss of life, and damage to both person and property. In this situation, the medical preparations might not always be enough. To assist in getting people to safety in these situations, scheduled vehicle dispatch might offer emergency logistical support[31]. Additional restrictions, such as the availability of medical services within the cars, etc., might be added to this issue.

  When considering additional modes of transportation like water (ships and ferries), aerial (airplanes and drones), and other comparable circumstances like school bus

pickup and drop-off, VRP may also be connected to them. Applications might change based on the circumstance, and constraints for solutions can change depending on the issue statement since two problem statements could not call for the same constraints. Drones are a significant, recently emerging VRP solution application. These mobile bots are fully employed in ways that decrease traffic, time, and fuel use.

- Drones: Drones are being more widely utilized for both commercial and non-commercial purposes, and they are also being used to implement VRP solutions known as VRPD [65]. A select few businesses, like Amazon, DHL, Federal Express, etc., employ drones with the intention of directly or indirectly delivering items. A VRPD algorithm was proposed by Xingyin Wang. in which delivery cars with a fleet of drones carrying items are driven to specific locations, where the drones assume control and proceed to distribute the goods to the adjacent residences before returning to the vehicle. Once all the allotted supplies have been delivered, the truck then makes its way back to the depot.

1.9 Vehicle Routing Problem's solutions

There are several ways to solve the VRP, however, the precise and heuristic techniques are the two that are most frequently employed. The most popular precise and heuristic approaches for solving the CVRP[59] are displayed in Table 1.2. When combined with extra personal logic, these strategies produce various outcomes. These two categories best describe how to find solutions. The study of operations research (OR), which aids in the implementation of vehicle routing solutions, is widely used.

Type of Approach	Solution procedure
Exact Solutions	Branch and cut Brunch and cut Pricing Pricing
Heuristic	Simple heuristic Meta heuristic Meta heuristic

Table 1.2: Proposed solutions to solve VRP.

- Exact solutions: Exact answers were the first VRP solutions, and it is well known that they handle the issue almost perfectly [17]. Exact solutions have the disadvantage of not operating effectively on large-scale problems [43]. But that took into

account the conventional precise answers. Large-scale difficulties are successfully solved by the new precise solutions with appropriate cuts and prices.

- Heuristic: One of the methods frequently utilized to address different VRP issues is heuristics. Finding the outcome that comes the closest to the ideal answer is the fundamental aspect of this methodology. According to Cordeau, the VRP does not have a precise ideal solution, hence the alternative is to search for responses that are near to the optimal ones [19]. Large issues can be solved using heuristic approaches rather than exact ones. The accuracy, speed, simplicity, and adaptability of the heuristics are their key characteristics. First, unlike the Exact Approach, which takes longer to present a complete answer, heuristics first present a good solution before beginning to improve it[19].

  Additionally, several delivery issues need quick fixes. For instance, Gendereau contends that while certain problems require long-term planning, the position of ambulances must be determined every three minutes[26]. In conclusion, the "Heuristic" method is crucial since truck routing difficulties call for quick, practical answers. Furthermore, heuristic algorithms like the Clarke and Wright method are simple to use but algorithms with several parameters might be challenging to comprehend. The heuristic approach is very adaptable since it works within a variety of limitations[19]. Simple heuristics and metaheuristics are the two subclasses of heuristics[19].

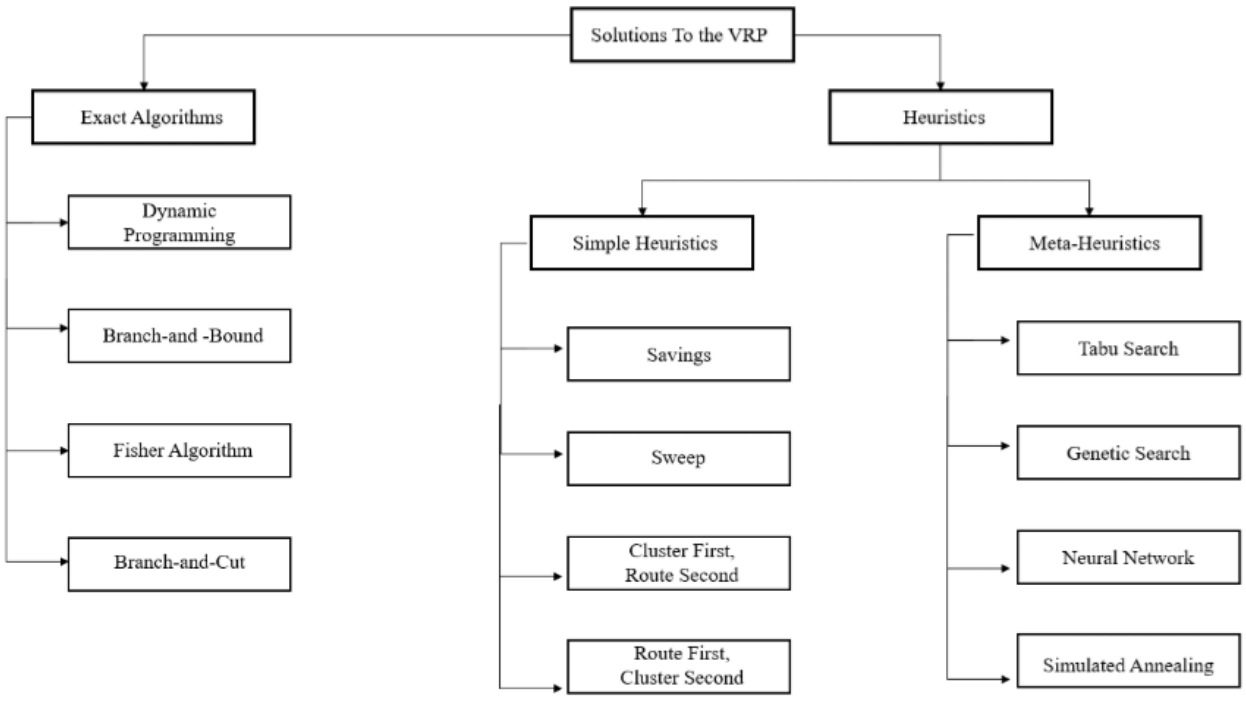


Figure 1.6: Types of algorithms solving VRP

1.10 Conclusion

In conclusion, the VRP is a challenging optimization problem that has many applications in the real world. There are a number of heuristic and metaheuristic methods that can be used to find good solutions to the VRP. The next chapter will discuss these methods in more detail.

Chapter 2

Overview of Simple heuristic and metaheuristic

2.1 Introduction

Optimization is an area of study within mathematics and computational science that focuses on developing methods and solving problems that aim to either minimize or maximize one or more objective functions. These functions are based on one or more dependent variables, which can have values that are either real or integer[24].

When engineers and researchers tackle optimization problems, they need to consider several factors based on the specific problem at hand. However, the two primary objectives that often take precedence are minimizing execution time and obtaining solutions that meet acceptable criteria[15]. therefore, these conventional optimization techniques do not guarantee global optimum performance and have led to the quest for new methods[44], one of which is simple heuristics and metaheuristics.

This chapter aims at giving an overview of metaheuristics and heuristics, so we will cover the history of these two methods. Then the important differences between them, After the definition and the classification with an explanation of each algorithm in every classification of each method, and Finally the comparison between them.

2.2 Historical study

In history Several problem-solving techniques tend to be metaheuristic; however, heuristic as a scientific technique for optimization is a modern phenomenon. metaheuristic algorithms or heuristic algorithms are developed in different historical periods. Between the 1940s and 1960s, heuristic methods were widely used in different applications, Researchers began developing new heuristics for various optimization problems, including the traveling salesman problem, the knapsack problem, and the quadratic assignment problem but the main achievement was in 1963 with the introduction of evolutionary algorithms (EA) by [8].

Between the 1960s and 1970s, Researchers began using mathematical programming techniques, such as linear programming and dynamic programming, to analyze heuristics and develop new ones. In addition to the development of Genetic algorithms by [28] in 1975.

Between the 1980s and 1990s found one of the biggest steps in the metaheuristic algorithms was the simulated annealing (SA) in 1983 by [42]. Another significant step was the development of artificial immune systems in 1986 by Farmer, Packard, and Perelson[22].

In 1986, for the first-time memory was used in metaheuristics by Fred Glover in the Tabu search (TS) algorithm in which the previous search moves are stored in a Tabu-list, and upcoming moves must avoid revisiting previous moves[27].

The 1990s and 2000s, were an exciting period for metaheuristic algorithms, as it saw

the development of several important algorithms such as ant colony optimization (ACO) by [5] in 1992, and particle swarm optimization (PSO) by [5] in 1995, and later differential evolution (DE) was developed by [56] in 1997.

From 2000 until nowadays, metaheuristic algorithms have been widely used in many applications, and many new significant algorithms have been developed. In 2001 Zong Woo Geem, Joong Hoon Kim, and G. V. developed the harmony search (HS) algorithm[25]. In 2002, a bacteria foraging algorithm was developed by [50]. In 2005 D. Karaboga developed the artificial bee colony (ABC). In 2009, Xin-She Yang and Suash Deb developed a cuckoo search (CS) algorithm[68].

Generally, there are many significant meta-heuristic algorithms that have been developed to solve real-life and optimization problems[5]

2.3 What is a heuristic method?

In computer science and artificial intelligence, heuristics serve as "rules of thumb" employed in algorithms to aid in finding approximate solutions for intricate problems. When confronted with an overwhelming amount of data that hinders quick resolution, heuristic algorithms are utilized to prioritize speed over exactness, As this definition say: "It is a rule for reducing the number of mental operations (or information-processing steps) taken to solve a problem" [53]. Nevertheless, since heuristics are derived from problem-specific rules, the details of these heuristics differ from one problem to another[63].

These methods strive to generate solutions within a reasonable timeframe that are sufficiently effective for solving the given problem. While the solutions obtained through heuristics may not be perfect or exact, they hold value as approximate or best-guess solutions. In situations where obtaining an exact solution would take hundreds of thousands of years, heuristics allow us to swiftly generate an approximate solution [63], "A heuristic is a technique aimed to solve a problem faster when traditional techniques are too slow" [33].

2.4 Why use a heuristic solution method?

Until now, the discussion in this section has primarily centered around a key rationale for employing heuristics: the difficulty, and sometimes impossibility, of finding the optimal solution for the mathematical representation of the given situation. However, there exist additional justifications for utilizing heuristic solution methods, they include :

- Facilitation of implementation: 'People would rather live with a problem they cannot solve than accept a solution they cannot understand' [67]. Decision-makers are

more likely to embrace and utilize decision rules when they possess a basic, intuitive understanding of how these rules function, particularly in terms of how crucial parameters influence the selected actions. This comprehension helps facilitate the acceptance and practical implementation of decision rules [55].

- Show improvement over current practices: Related to the previous point, managers may be quite satisfied with a heuristic solution that produces better results than those currently achieved [55].
- Fast results: Sometimes fast, reasonable, results are needed and heuristics can be more quickly developed and used than optimization routines [55].
- Robustness: Heuristics can be less sensitive to variations in problem characteristics and data quality. In the words [12]'Optimal solutions are fragile in the sense that they can be exquisitely sensitive to changes in the data'. In cases where there is a slight modification in the problem description, achieving an optimal solution often necessitates solving the entire problem again, which can be computationally intensive and time-consuming, especially if the initial problem was already challenging to solve [55].
- Use within optimization routines: Heuristics can be effectively incorporated into optimization routines through two approaches. Firstly, they can contribute by providing favorable initial solutions within an iterative framework. Secondly, they can offer bounds that aid in eliminating portions of the solution space, particularly in optimization methods involving partial enumeration[55].

2.5 Classification of heuristic methods

There are several classifications of heuristic methods, each with its strengths and weaknesses. In this discussion, we will explore each of these classifications of heuristics in the figure below:

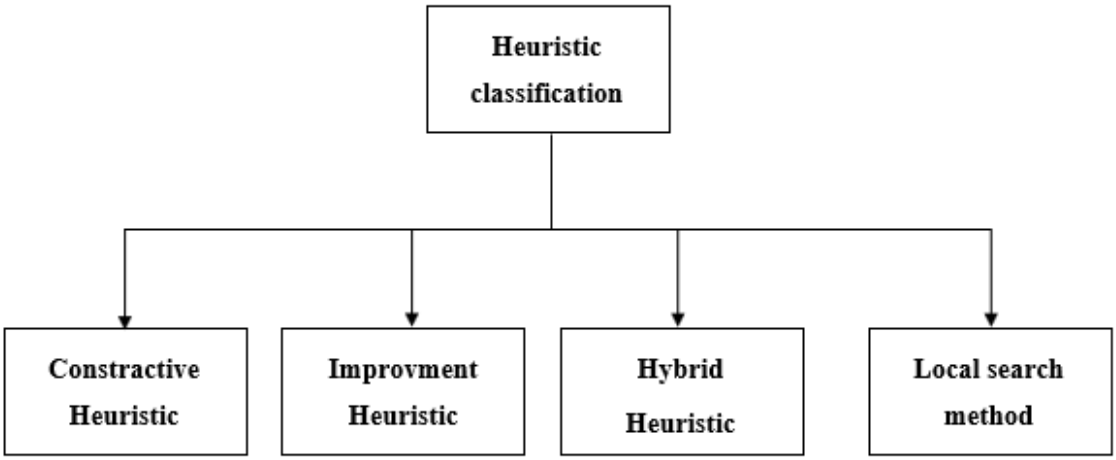


Figure 2.1: Classification of heuristic methods

2.5.1 Constructive heuristic

Constructive methods, as the name implies, use the data of the problem to construct a solution, step by step. Typically, no solution is obtained until the procedure is complete (in contrast with improvement methods to be discussed in the next subsection). A special constructive approach is the so-called greedy method, where, at each step, the next element of the solution is chosen so as to give the best immediate benefit (highest profit contribution or lowest cost) [55], one of the most common algorithms in this class is KNN algorithm.

- The k-Nearest Neighbors:(k-NN) algorithm is a popular machine learning technique extensively employed in data classification research. However, with the advent of big data, the conventional k-NN algorithm's performance and efficiency have become significant concerns. Specifically, when dealing with large-scale multi-categorical training datasets, the traditional k-NN algorithm proves to be inefficient. It encounters difficulties in effectively filtering the training dataset to extract the most relevant data for the given test dataset or file[4].

2.5.2 Improvement heuristic

Improvement heuristics start with an initial solution, i.e., a complete solution obtained either by a constructive heuristic or randomly generated. This initial solution is then improved by applying small consecutive changes until a stopping criterion is met. The goal is to keep the computational times fairly low while improving the quality of the solutions when compared to constructive heuristics by [49], sweep algorithm is one of the common algorithms used in this class.

- Sweep algorithm: The heuristic proposed here uses different procedures to generate

a large set of good routes and then chooses those that satisfy the problem constraints at the lowest cost using a polynomial set partitioning algorithm. Specifically, the proposed heuristic uses five subordinate procedures called: Order, 1-petal, 2-petal, Petals Selection, and Improve[52].

2.5.3 Hybrid Heuristics

These methods combine two or more heuristic methods to take advantage of their strengths and overcome their weaknesses[58], the GPHH one of the best examples.

- The Genetic Programming Hyper-heuristic:(GPHH) is a promising approach for effectively handling the Incapacitated Arc Routing Problem (UCARP) by automatically evolving efficient routing policies. However, one drawback of GPHH is that the evolved routing policies can often be overly complex, making it difficult for human users to understand and trust them[64].

2.5.4 Local Search Method

In this method, the most feasible way of solving a problem is to search and used. Continuous improvement is made in the method during the solving process, and when there is no more scope for improvement, the method gets to the end, and the final result is the answer to the problem [33].

These are just a few examples of the classifications of heuristics methods that are commonly used in optimization problems. The choice of heuristic method depends on the specific problem, its characteristics, and the available computational resources.

It is important to recognize that the various classifications of heuristics are not mutually exclusive. In fact, there are often advantages to integrating multiple heuristic methods when addressing a particular category of problems. Furthermore, using multiple distinct methods simultaneously to solve the same problem can lead to more fruitful results, allowing for the selection of the best solutions[55].

2.6 What is a metaheuristic method?

Metaheuristics are not specifically focused on solving any kind of problem, but they propose simple ideas with high applicability to a wide number of problems[48]. “A metaheuristic is a higher-level technique or heuristic that seeks, generates, or selects a heuristic that may provide a sufficiently good solution to an optimization problem “[44].

“Metaheuristics were conceived as high-level problem-solving strategies to coordinate

the cooperation between other search methods, including heuristics and/or traditional search techniques" [48]. These simple procedures are usually based on emulating natural or physical phenomena, such as the behavior of flocks of birds and insects, cooling procedures in metals, or the natural evolution, among many others[48].

2.7 Why use a metaheuristic solution method?

- Meta-heuristics can lead to good enough solutions for computationally easy (technically, P class) problems with large input complexity, which can be a hurdle for classical methods.
- Meta-heuristic can lead to good enough solutions for the NP-hard problems, i.e. problems for which no known exact algorithm exists that can solve them in a reasonable amount of time.
- Unlike most classical methods, meta-heuristics require no gradient information and therefore can be used with non-analytic, black-box, or simulation-based objective functions.
- Most meta-heuristics have the ability to recover from local optima due to inherent stochasticity or deterministic heuristics specifically meant for this purpose.
- Because of the ability to recover from local optima, meta-heuristics can better handle uncertainties in objectives.
- Most meta-heuristics can handle multiple objectives with only a few algorithmic changes[11]

2.8 Classification of metaheuristic methods

Here, we present a possible classification of the most common combinatorial optimization methods applied in science and engineering.

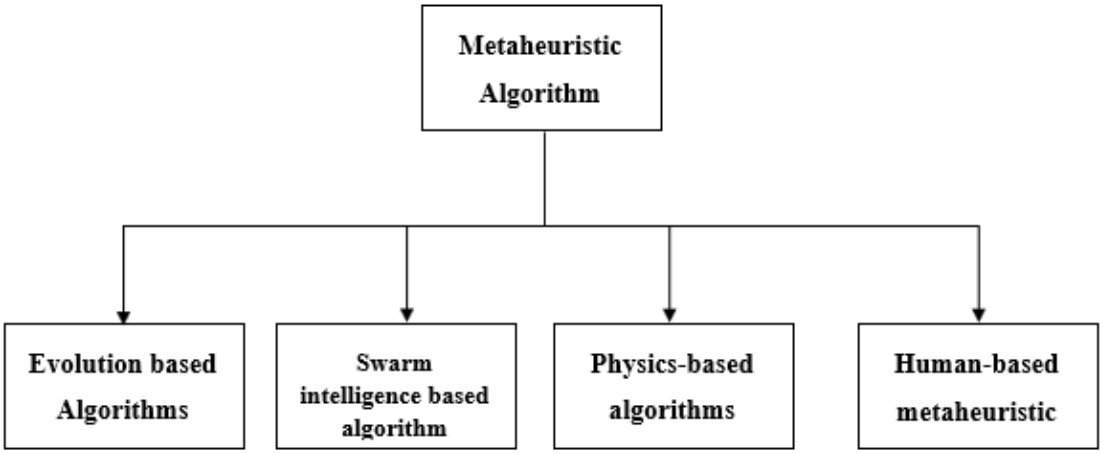


Figure 2.2: Classification of metaheuristic methods
[3]

2.8.1 Evolution-based Algorithms (EAs)

Simulate the biological progression of evolution at the cellular level employing selection, crossover, mutation, and reproduction operators to generate increasingly better candidate solutions (chromosomes). For evolutionary computation, there are four historical paradigms: evolutionary programming, evolutionary strategies, genetic algorithms, and genetic programming [1], The most EAs algorithm is the genetic algorithm.

- The genetic algorithm (GA): This algorithm is a random and nonlinear search method based on the principles of natural selection. This method is found by John Holland in around 1960. The proofs of GA start from the coding or parameters coding. Next, the chromosomal generation is done. A chromosomal generation is a group of genetics[61].

2.8.2 Swarm intelligence (SI)

Mimics are the collective behavior of agents in a community, such as birds and insects. SI mainly depends on the decentralization principle i.e.; the candidate solutions are updated through the local interaction with each other and with their environment. The most popular SI algorithms are Particle Swarm Optimization (PSO)[61], Ant Colony Optimization (ACO) [1]

- Ant Colony Optimization (ACO): Ant Colony Optimization is inspired by ants and their behavior of finding the shortest paths from their nest to sources of food. Without any leader that could guide the ants to optimal trajectories, the ants

manage to find these optimal trajectories over time, by interacting with their local environment. The ants initially search for food in a random fashion, but when they have found some, they return home while depositing chemicals, called pheromones. These pheromones attract other ants to follow the same path, and they in turn also deposit pheromones on their way back. Over time, this behavior leads to the emergence of paths, that can be shown to be near-optimal[61].

2.8.3 Physics-based algorithms (PAs)

This family of algorithms is based on the behaviors observed in nature that are not related to biological processes. In general, they are based on physical observation and experimentation[14]. Some of the most recognized and widely used physics-inspired optimization algorithms is simulated annealing

- Simulated Annealing (SA): This one is an effective and general form of optimization. It is useful in finding global optima in the presence of large numbers of local optima. "Annealing" refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal. Simulated annealing uses the objective function of an optimization problem instead of the energy of a material[7].

2.8.4 Human-based meta-heuristic algorithms (HAs)

Are search optimization methodologies that emulate human interactions in different environments [54]. These algorithms exploit the behavior of humans in groups in order to solve real-life problems and learn new skills. The most common socially inspired algorithm is tabu search.

- Tabu search (TS): This is an iterative neighborhood search algorithm, where the neighborhood changes dynamically. TS enhances local search by actively avoiding points in the search space already visited. By avoiding already visited points, loops in search trajectories are avoided and local optima can be escaped. TS can be considered as the combination of local search (LS) and memory structures. The main feature of TS is the use of explicit memory. The uses of memory have two goals: to prevent the search from revisiting previously visited solutions and to explore the unvisited areas of the solution space[40].

2.9 Comparison of heuristic and metaheuristic Methods

The table below provides an overview of the characteristics of heuristic and metaheuristic:

characteristics	Heuristic methods	Metaheuristic methods
Nature	Deterministic	Randomization+heuristic
Type	Algorithmic Iterative	Nature inspired Iterative
Nature of solution	Inexact, Near-Optimal solution	Inexact, Near-Optimal solution
Optimal Result	Not guaranteed	Not guaranteed
Correct Result	Guaranteed	Guaranteed
Execution Time	Typically fast	Tuning-dependent but typically fast

Table 2.1: Comparison between heuristic and metaheuristic .
[34]

2.10 Conclusion

In conclusion, heuristic and metaheuristic methods are powerful tools for solving the Vehicle Routing Problem (VRP). These methods can find good solutions to the VRP in a reasonable amount of time, even when the VRP is very large and complex. The next chapter will discuss the methodology used in this study to solve the VRP.

Chapter 3

Methodology

3.1 Introduction

This chapter presents a methodology for solving the Vehicle Routing Problem (VRP) using modified existing codes. In this study, we build upon original codes obtained from reputable sources such as GitHub and adapt them to address our specific VRP requirements. By making these modifications, we aim to develop a tailored solution that optimizes route planning, resource utilization, and cost efficiency. The proposed methodology offers a practical approach for efficiently solving the VRP and has the potential to bring significant improvements to real-world operations.

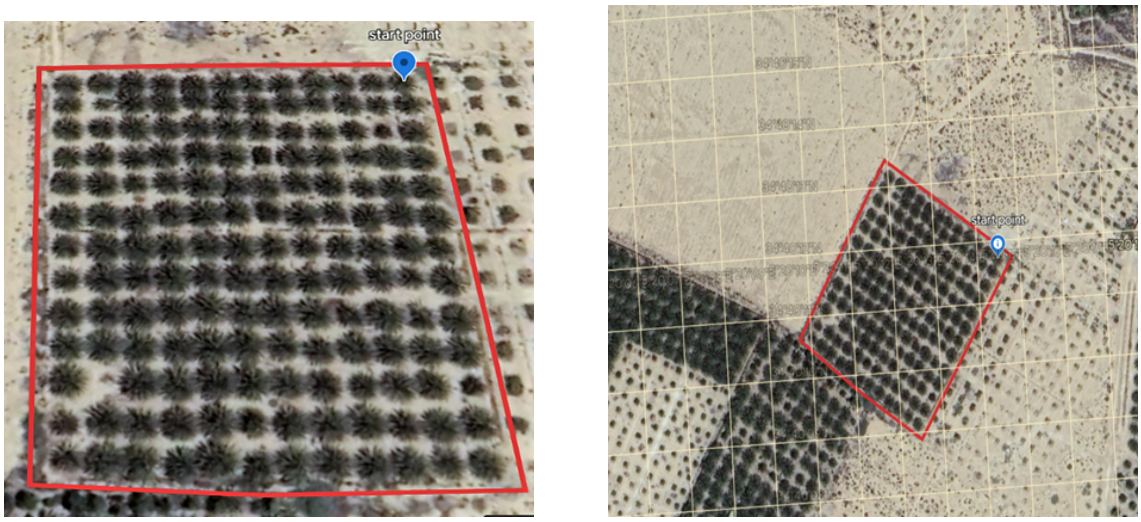
3.2 Data collection

To apply our study in a real word case, We have chosen a specific piece of land for analysis, and the information presented in this assignment is based on observations and analysis conducted using Google Earth.

- Google Earth: This tool is a virtual globe and geographic information system (GIS) software developed by Google. It provides a 3D representation of the Earth based on satellite imagery, aerial photography, and GIS data[66]. Google Earth is available as a desktop application for Windows, macOS, and Linux, as well as a web-based version called "Google Earth Web." in our work, we use the application google earth pro version 7.3
- Google Earth Pro: This is a professional version of Google Earth, provided by Google, that ties extensive satellite data together into one system to visualize the earth and study various geographic aspects. It offers enhanced capabilities for data analysis, visualization, and presentation, making it a valuable tool for geospatial professionals and researchers [29]

By utilizing Google Earth's satellite imagery and mapping capabilities, we were able to access detailed information about the selected area, allowing for a comprehensive assessment of its features, characteristics, as follows:

- Location: We have chosen a specific palm tree forest in Biskra, and the coordinates of the selected land parcel are as follows: latitude 34.6697561, longitude 5.3368609.
- Provide the coordinates of starting point: The coordinates of the selected piece of land, after converting them from geographical coordinates to Cartesian coordinates, are as follows: x= 714187.08 and y= 3838925.34.
- Describe the geographical location: The number of trees in length is 13, and in width, it is 12. The perimeter of the zone is 0.37 km, which is equivalent to 374.26 m, the area of the zone is 8675.45 m^2.



(a) (b)

Figure 3.1: Piece of land chosen in various positions

Figure 3.1 is a captivating photo, taken from Google Earth, that showcases our forest study area. To obtain the dataset, we employ the grid coordinate method and we adapted this method to our work:

3.2.1 The grid coordinate method

Is a surveying technique used to establish coordinates for specific points on a piece of land. It involves dividing the land into a grid-like pattern of squares or rectangles and assigning coordinates to the corners of these squares to calculate the coordinates of the grid squares[30], a reference point is established. By moving eastward or northward from this reference point, the coordinates of each grid point can be determined based on the chosen side length. To adapt this method to our specific situation, we developed the following steps:

- Selecting a reference point: The first step is to choose a reference point or start point within the study area. This point typically has coordinates (714187.08, 3838925.34).
- Establishing grid dimensions: Determine the size of the grid squares or rectangles based on the desired accuracy and the characteristics of the terrain depending on the application. in this situation, we determine the number of trees in length and latitude to get a matrix of points
- Assigning coordinates: if the grid dimension is 10 meters, the second point may have coordinates (714187.08, 3838935.34), the third point (714187.08, 3838935.34).

For applied this method we create the following algorithm:

Algorithm 1 Function *Get − Address* using grid method

Input: x0,y0,m,n

Output: X,Y

function *Get − Address (x0,y0,m,n) : table*

```
    x← [x0]
     y← [y0]
     for <i from 1 to m > do
        a ← y[i-1]+10.0
         b ← x[1]
         y ← y+[a]
         x ← x+[b]

        for <i from 1 to n-1 > do
            a ← X[j]+10.0
             b ← Y[1]
             x ← X+[a]
             y ← Y+[b]

        end
    end
    df ← Empty DataFrame
     df[x] ← x
     df[y] ← y

    return X,Y
end
```

We obtained 157 sets of coordinates as the output, which were saved in a CSV file. The figure below illustrates little bit of the output

X	Y
714187.1	3838925
714187.1	3838935
714197.1	3838935
714207.1	3838935
714217.1	3838935
714227.1	3838935
714237.1	3838935
714247.1	3838935
714257.1	3838935
714267.1	3838935
714277.1	3838935

Figure 3.2: example of Output the grid method

3.3 Algorithms for solving VRP

3.3.1 Nearest Neighbour (NN)

NN works by finding the k most similar instances (neighbors) to a new instance and then predicting the label of the new instance based on the labels of the k neighbors. The value of k is a hyperparameter that must be chosen by the user.

To find the k most similar instances, KNN calculates the distance between the new instance and all of the instances in the training dataset. The distance can be calculated using any distance metric, such as the Euclidean distance, the Manhattan distance, or the Chebyshev distance.

Once the distances have been calculated, the k instances with the smallest distances are considered to be the k nearest neighbors. The label of the new instance is then predicted by a majority vote. For classification tasks, the label of the new instance is the most common label among the k nearest neighbors. For regression tasks, the label of the new instance is the average of the values of the k nearest neighbors.

Original code

We obtained the original code from GitHub [38], which was written by an author identified as “Vivek Kulkarni“ [36]. The original code focuses on implementing the K Nearest

Neighbors (KNN) algorithm for classification in Python using the $scikit-learnlibrary$. It begins by importing necessary libraries such as pandas, seaborn, matplotlib. pyplot, and numpy. The code loads a dataset using pandas from a CSV file named "Classified Data" and performs some preprocessing steps. It standardizes the variables using the StandardScaler from the $scikit-learn$ library to ensure that variables on different scales do not affect the KNN classifier disproportionately. Then, it splits the data into training and testing sets using the $train-test-splitfunction$.

Next, the code trains a KNN classifier with $k = 1$ ($n-neighbors = 1$) using the $KNeighbors-Classifier$from $scikit-learn$. It fits the classifier to the training data and predicts the testing data. The code also calculates and prints the confusion matrix and classification report, which provide insights into the performance of the KNN model.

Furthermore, the code applies the elbow method to determine a suitable value of k. It iterates over a range of k values and calculates the error rate for each value. The error rates are then plotted against the k values to visualize the elbow point, which indicates the optimal k value where the error rate stabilizes. The code then retrains the KNN model with $k = 1$ and $k = 23$, printing the confusion matrix and classification report for both cases.

Modified code

[1] The modified code introduces a different approach for solving a different problem. It defines several functions to calculate the distance between two points, find the k nearest neighbors of each location, and obtain routes for multiple vehicles. It also includes a local search algorithm to improve the obtained solution. The main part of the code initializes the number of vehicles, and the coordinates of locations, and obtains the nearest neighbors. It then executes the algorithm to find a good solution using the nearest neighbors and local search. The code calculates the total distance of the best solution, prints the distance, and measures the runtime. Finally, it visualizes the solution by plotting the locations and routes of the vehicles using *matplotlib*.

The modification involves a complete change in the problem being solved and the approach used. The original code dealt with classification using the KNN algorithm on a dataset, while the modified code focuses on solving a vehicle routing problem using a different algorithmic approach.

As for the difference between functions and parameters, functions are blocks of reusable code that perform a specific task when called, while parameters are variables that are used to pass information to functions. In the original code, functions such as fit, transform, predict, and various functions from $scikit-learn$ libraries are used. Parameters are passed to these functions to specify their behavior, such as the number of neighbors (k)

[1]The modified code is in QR code

in the$KNeighbors - Classifier$.

The modification introduces custom functions, including $get-distance$, $get-nearest-neighbors$, and $get - routes$. These functions take specific parameters, such as coordinates of points, the number of nearest neighbors, and the list of locations, to perform computations and return results.

It should be noted that the time complexity of both the original and modified code is $O(k * n2)$, where k is the number of vehicles and n is the number of locations. This is because both algorithms involve iterating over all locations for each vehicle, which results in quadratic time complexity.

Algorithm 2 Function *get − good − solution*(locations, k, *nearest − neighbors*)

Input: locations , k , *nearest − neighbors*

Output: list of routes for each vehicle

function *get − good − solution (locations, k,nearest − neighbors) : list*

routes ← an empty list

unvisited locations ← set of locations excluding the depot (location 0)

location − per − vehicle ← integer division of total locations by *k*

remaining − locations ← total locations excluding the depot

for `<i from 0 to k-1 >` **do**

while *(unvisited − locations* ***and*** *remaining − capacity*0*)* **do**

`distances ← [(`*get − distance*`(location[0], location[1], route[-1][0], route[-1][1]), location)`

nearest − location ← `location with minimum distance to the last location in the route`

for *location in unvisited − locations* **do**

nearest − location ← `location with minimum distance to the last location in the route`

route`.append(`*nearest − location*`)`

unvisited − locations`.remove(`*nearest − location*`)`

remaining − capacity ← *remaining − capacity* − 1

end

end

route.append(depot location)

routes.append(*route*)

remaining − locations ← *remaining − locations − locations − per − vehicle*

end

if *there are remaining locations* **then**

for *i* ***from*** *0* ***to*** *remaining − locations* − 1 **do**

location ← *unvisited − locations*.pop()

routes[*i*%*k*].insert(-1, *location*)

end

end

return *routes*

end

Overall, the original code focused on classification using the KNN algorithm, while the modified code tackles a vehicle routing problem using a different algorithmic approach which is a combination of NN and local search.

3.3.2 Ant Colony Optimization (ACO)

The ACO algorithm works by simulating the behavior of ants. A population of ants is created, and each ant starts at the depot. The ants then randomly visit locations, and the cost of traveling between locations is calculated using the distance between the locations. After visiting a location, an ant deposits pheromones on the path it took. The more ants that travel along a path, the stronger the pheromone trail becomes. This encourages other ants to follow the same path.

The ACO algorithm repeats this process for a number of iterations. After each iteration, the pheromone trails are updated. The stronger the pheromone trail, the more likely an ant is to travel along that path.The algorithm terminates when a termination criterion is met, such as a maximum number of iterations or a certain level of improvement in the solution

Original code

We obtained the original code from GitHub [39], which was written by an author identified as "Kirill Temlyakov" [37], and focuses on implementing the Ant Colony Optimization (ACO) algorithm for solving a specific problem. It defines the "AntColony class", which initializes various parameters and data structures, including the distances matrix and the pheromone matrix.

The algorithm runs for a specified number of iterations and performs key operations such as generating all possible paths, depositing pheromones on the best paths, and updating the best solution found so far. It uses techniques like path generation, pheromone spreading, and path selection based on pheromone levels and distances. The code also tracks and prints the shortest path found during each iteration and maintains an overall best path. However, it lacks certain features, such as runtime calculation and result visualization.

Modified code

In the modified code, several changes and additions are made to solve a different problem using the ACO algorithm. The code starts by importing necessary libraries such as *numpy*, *random*,*time*, and *matplotlib.pyplot*. It defines a new class called ACO, which encapsulates the ACO algorithm's implementation. The ACO class includes various parameters such as the number of ants, number of iterations, evaporation rate, and weights for pheromone and distance calculations. It also introduces a $calculate_distance()$ method using $numpy's linalg.norm()$ function to compute the distance between two points.

To solve the Vehicle Routing Problem (VRP), the code further extends the ACO class by adding a solve() method. This method takes inputs such as locations, demand,

capacity, and the number of vehicles. It initializes a distance matrix by calculating the distances between each pair of locations. It then initializes variables for tracking the best solution and its corresponding distance. The code proceeds to solve the VRP by applying the ACO algorithm, and modifying the original code to fit the VRP context. The modified code iteratively generates solutions, spreads pheromones, and updates the best solution based on the distances and pheromone levels. It utilizes techniques like path generation, pheromone spreading, and path selection using probabilities based on pheromone levels and distances.

Additionally, the modified code introduces new functionalities to enhance the overall analysis. It calculates the runtime of the ACO algorithm by measuring the execution time between start and end timestamps using the time module. Furthermore, the code includes result visualization using *matplotlib.pyplot*, allowing the plotted visualization of the obtained routes on a 2D coordinate system. This provides a clear representation of the VRP solution and allows for the visual inspection of the routes and their spatial distribution.

It should be noted that both the original and modified code have a time complexity of $O(k * n2)$, where n is the number of locations, depending on the problem being solved. This is because both algorithms involve iterating through all locations.

Algorithm 3 Function *Solve*(locations, demand, capacity, $num - vehicles$)

Input: locations, demand, capacity, $num - vehicles$

Output: $best_solution, best_distance$

function *Solve (locations, demand, capacity, num − vehicles): list*

- $num\_cities \leftarrow$ number of cities
- $distance - matrix \leftarrow$ a matrix of size $num - citiesnum - cities$ filled with zeros
- **for** *location in unvisited − locations* **do**
  - **for** *location in unvisited − locations* **do**
    - $distance - matrix[i][j] \leftarrow calculate - distance$(locations[i], locations[j])
  - **end**
  - $best - distance \leftarrow$ infinity
  - $best - solution \leftarrow$ an empty list
  - $best - solution \leftarrow$ a list of $num - vehicles$ empty lists
  - $visited - locations \leftarrow$ a list of $num - vehicles$ empty lists
  - **for** i ***from*** *0* ***to*** $num - cities - 2$ **do**
    - $assigned \leftarrow$ false
    - **while** *assigned is false* **do**
      - $vehicle \leftarrow$ choose a vehicle randomly between 0 and $num - vehicles - 1$
      - **if** *i is not in visited − locations*[*vehicle*] **then**
        - $best - solution[vehicle]$.append(i)
        - $visited - locations[vehicle]$.append($i$)
        - $assigned \leftarrow$ true
      - **end**
    - **end**
  - **end**
  - **for** *vehicle* ***from*** *0* ***to*** $num - vehicles - 1$ **do**
    - $best - solution$[vehicle] $\leftarrow [0] + best - solution$[vehicle] + [0]
  - **end**
- **end**
- $best - distance \leftarrow$ sum of all elements in $distance - matrix$
- **return** $best - solution$ and $best - distance$

end

In summary, the original code implements the ACO algorithm for a specific problem, while the modified code adapts it to solve the VRP. The modifications introduce additional functionalities, such as runtime calculation and result visualization, to enhance the analysis and understanding of the VRP solution.

3.3.3 Proposed Algorithm Based on Clarke-Wright (PABCW)

Proposed Algorithm code

The proposed algorithm to solve VRP is based on the Clarke-Wright Algorithm. It is an algorithm that iteratively constructs routes by adding nodes to existing routes in a way that minimizes the total distance traveled. The algorithm works as follows:
1. Start with a single route that includes the depot.

2. For each node that is not yet assigned to a route:

- Calculate the distance from the node to each existing route.

- Assign the node to the route with the shortest distance.

3. Repeat steps 2 and 3 until all nodes are assigned to a route.

The algorithm has a complexity of $O(nlogn + n)$, where n is the number of nodes in the VRP problem.

It begins by importing necessary libraries such as math, random, time, and $matplotlib-pyplot$. These libraries provide functions for mathematical calculations, random number generation, timing, and plotting, respectively.

Next, the code defines a Node class to represent a node in the VRP. Each node is defined by its x and y coordinates, as well as its demand (the number of goods to be delivered or collected at that node).

The $calculate-distance$ function calculates the Euclidean distance between two nodes based on their coordinates. This distance calculation is used to determine the distance traveled between nodes in the VRP. The $get-anglefunction$ calculates the angle between a given node and the depot (the starting and ending point of the routes). This angle is used to sort the nodes in a clockwise manner around the depot.

The $get - sorted_nodes$ function takes a list of nodes and the depot as input and returns the nodes sorted in a clockwise order around the depot. This sorting is based on the calculated angles using the $get - anglefunction$.

The $create - route$ function takes a list of nodes, the depot, and the vehicle capacity as input and creates a route by adding nodes to the route until the vehicle capacity is reached. The remaining capacity is updated accordingly, and the depot is added as the last node in the route.

The main function $solve - vrp$ takes a list of nodes, the depot index, the vehicle capacity, and the number of vehicles as input. It first removes the depot from the list of

nodes. Then, it sorts the remaining nodes based on their angles around the depot using the $get - sorted - nodes$ function.

Next, it initializes an empty list called routes to store separate routes for each vehicle. It iterates through the sorted nodes and assigns them to vehicles in a round-robin fashion based on their capacity constraints. If a node can be added to a vehicle's route without exceeding the vehicle's capacity, it is added to that route.

After assigning the nodes to the vehicles, the depot is added as the first and last node in each route using the insert function. Finally, the function returns the routes.

The $calculate - total - distance$ function calculates the total distance traveled in a given route by summing up the distances between consecutive nodes using the $calculate - distance$ function.

The code then defines the locations and demand for the VRP instance. The locations are represented as x and y coordinates, and the demand represents the number of goods to be delivered or collected at each location. Next, the code initializes variables for the vehicle capacity, depot index, and number of vehicles. It creates a list of Node objects based on the provided locations and demand.

The code measures the runtime of the algorithm by recording the start time using $time.time()$, running the $solve - vrp$ function, and then calculating the elapsed time. A delay of 1 second is introduced using $time.sleep()$ to simulate a longer runtime. After that, the code extracts the coordinates of the routes for plotting purposes. It creates a separate list for each vehicle's route and appends the (x, y) coordinates of each node in the corresponding route.

Finally, the code plots the routes using $matplotlib.pyplot$, labels each route with the corresponding vehicle number and displays the runtime on the plot. The plotted routes are shown in a 2D coordinate system, with the $x - coordinate$representing the horizontal position and the $y - coordinate$representing the vertical position. Additionally, the code prints the routes and calculates the total distance traveled for each route. It keeps track of the best route with the lowest total distance. The code also performs additional calculations such as the average distance per vehicle, maximum distance per vehicle, average locations per vehicle, standard deviation of locations per vehicle, and standard deviation of the solution (stability).

The code displays the runtime and additional statistics, including the average distance per vehicle, maximum distance per vehicle, average locations per vehicle, the standard deviation of locations per vehicle, the standard deviation of the solution, and the best distance achieved.

Algorithm 4 Function *solve − vrp*(nodes, *depot − index* , *vehicle − capacity* , *num − vehicles*)

Input: nodes, *depot − index*,*vehicle − capacity*, *num − vehicles*)
Output: routes
function *solve − vrp(nodes, depot − index,vehicle − capacity, num − vehicles) : list*
    depot ← node at *depot − index*
    Remove *depot* from the nodes list
    nodes ← nodes sorted by distance from *depot*
    routes ← a list of empty lists for each vehicle
    for *i* ***from*** *0* ***to*** *len(nodes) − 1* **do**
        route − index ← *i*%num-vehicles
        current − route ← route for the current vehicle
        current − capacity ← sum of demand of nodes in the current route
        if *current-capacity + node.demand* ≤ *vehicle-capacity* **then**
            current_route.append(node)
        end
    end
    for *each route in routes* **do**
        route.insert(0, depot)
        route.append(depot)
        return routes
    end
end

Once the data has been collected, the dataset created, and the methods adapted to address our specific VRP problem, the subsequent chapter will commence the comparative analysis.

3.4 Conclusion

In conclusion, we adapted the K-nearest neighbors (KNN) and Ant Colony Optimization (ACO) methods to solve the Vehicle Routing Problem (VRP). We also proposed a new algorithm for solving the VRP. The experimental results will be discussed in the next chapter.

Chapter 4

Experimental Setting

In addition to the benchmark datasets provided by [9], this chapter extends the evaluation by incorporating a real-world case dataset. By including a real-world case, the study aims to provide a more realistic assessment of the Ant Colony Algorithm (ACO), Nearest Neighbors (NN) algorithm, and a proposed algorithm (PABCW).

The comparison entails analyzing and comparing the solutions obtained by the benchmark dataset, NN algorithm, Ant Colony algorithm, and the proposed algorithm across multiple instances. Various criteria, such as runtime and complexity, are employed to assess the performance of these algorithms thoroughly.

The overarching objective of this chapter is to offer valuable insights into the effectiveness and applicability of different algorithms in addressing the Capacitated Vehicle Routing Problem (CVRP). To achieve this, the analysis leverages both benchmark datasets and a real-world case dataset, providing a robust foundation for conducting a comprehensive comparison. By considering real-world scenarios, the study aims to enhance the practical relevance of the findings and facilitate informed decision-making in real-world routing problems.

4.1 Comparative study

To test the effectiveness of our approach based on ACO, NN, and the PABCW, we compare our results where to apply the dataset of benchmark in this algorithm with the optimal solutions of benchmark by augerat[9], then according to our dataset.

4.1.1 CVRP Benchmark Instances

The benchmark consists of problem instances designed to represent different CVRP scenarios. These instances vary regarding the number of customers, vehicle capacities, and distances between locations, providing diverse problem instances for evaluating CVRP algorithms. The CVRP. benchmark typically includes the following components:

- Problem Instance Format: Problem instances are represented using text files or sets of files containing essential information such as the number of customers(n), number of vehicles (k), vehicle capacity, coordinates of the depot and customers, and the specific demand associated with each customer.

- Customer Demand: Each customer within the benchmark has an individual demand, indicating the quantity of goods or services to be delivered to that customer. The demands can vary across different problem instances, reflecting diverse delivery requirements.

- Vehicle Capacity: The benchmark encompasses instances with varying vehicle capacities, enabling the evaluation of algorithms under different constraints related to the maximum load each vehicle can carry.
- Solution Quality Measures: The benchmark may provide known optimal or best-known solutions for the given problem instances. These solutions serve as benchmarks to measure the quality of solutions produced by algorithms, facilitating performance evaluation and comparison between different algorithmic approaches.

In the table below 4.1 we present the instances chosen of augerat benchmark dataset (While n is nodes' number, k is vehicles' number, and Q is vehicles' capacity).

Instance	Q	Tightness	Opt
P-n20-k2	160	0.97	216
P-n21-k2	160	0.93	211
P-n22-k2	160	0.96	216
P-n101-k4	400	0.91	681

Table 4.1: Instances of the set P [9]

4.1.2 First comparison(with benchmark)

In this section, we compare Augerat benchmark solutions with the outcomes of the NN algorithm, ACO, and the PABCW. Through this comparison, we aim to evaluate the performance and accuracy of the algorithms. The result of this comparison is shown in the table 4.2 :

4.1.3 Second comparison (built dataset)

In this section, we will compare the performance of the previous methods on our dataset based on two criteria: runtime and complexity. By evaluating these factors, we can better understand how each method performs on our specific dataset and determine which best suits our needs; the result is shown in table 4.3.

Instance	Best solution	NN	ACO	PABCW
P-n20-k2	Route 1: 19 5 14 16 9 7 2 10 1 Route 2: 6 13 8 17 18 3 12 15 11 4	Route 1: 19 5 7 2 13 9 16 14 18 Route 2: 1 10 12 15 4 11 3 8 17	Route 1: 0 1 3 7 8 10 15 Route 2: 2 4 5 6 9 11 12 13 14 16 17 18	Route 1: 14 5 7 6 2 17 3 1 Route 2:19 16 9 13 8 18 10 12 15 4 11
p-n21-k2	Route 1: 20 5 14 17 9 13 2 7 6 Route 2: 16 1 10 8 18 19 3 12 15 11 4	Route 1: 6 20 5 7 2 13 9 17 14 19 Route 2: 16 1 10 12 15 4 11 3 8 18	Route 1: 4 5 6 7 11 12 17 18 Route 2: 0 1 2 3 8 9 10 13 14 15 16 19	Route 1:14 5 7 6 2 18 19 10 12 Route 2:20 17 9 13 8 16 3 1 15 4 11
P-n22-k2	Route 1: 6 2 13 9 7 21 17 14 5 20 Route 2: 16 1 10 8 18 19 3 12 15 11 4	Route 1: 6 20 5 21 7 2 13 9 17 14 Route 2: 16 1 10 12 15 4 11 3 8 18 19	Route 1: 0 2 3 5 6 7 9 13 14 15 17 19 20 Route 2: 1 4 8 10 11 12 16 18	Route 1: 14 5 21 9 13 8 16 3 1 15 4 11 Route 2: 20 17 6 2 18 19 10 12
P-n101-k4	Route 1: 6 96 99 59 92 93 98 37 100 91 85 61 16 86 38 44 14 42 43 15 57 2 87 97 95 94 Route 2: 53 26 12 80 68 29 24 54 4 55 25 39 67 23 56 75 22 41 74 72 73 21 40 58 13 Route 3: 28 76 77 3 79 78 34 35 71 65 66 20 32 90 63 64 49 36 47 46 8 45 17 84 5 60 83 18 89 Route 4: 27 69 1 50 33 81 9 51 30 70 10 62 11 19 48 82 7 88 31 52	Route 1: 89 6 94 95 97 92 59 99 96 93 85 98 37 100 91 16 61 86 44 12 42 87 2 57 15 Route 2:13 21 73 72 74 22 75 56 39 23 67 25 54 4 24 29 32 63 64 7 56 38 43 41 Route 3: 27 28 76 50 1 69 52 18 83 60 5 84 17 45 8 82 48 47 36 49 19 11 62 10 90 Route 4: 53 58 40 26 12 80 68 77 3 7 9 78 34 81 33 51 9 71 35 65 66 20 30 70 31 88	Route 1: 2 6 9 15 16 18 21 23 29 33 38 39 40 42 44 49 50 52 55 58 61 63 64 65 73 81 92 96 97 Route 2: 12 13 20 22 24 27 28 30 37 43 51 53 54 59 60 72 78 79 80 83 85 86 87 89 90 91 94 95 98 Route 3: 1 5 8 10 11 17 19 25 32 35 36 41 46 47 66 67 68 69 74 77 84 88 99 Route 4: 0 3 4 7 14 26 31 34 45 48 56 57 62 70 71 75 76 82 93	Route 1: 5 16 85 98 92 79 43 58 40 75 25 12 29 76 33 9 1 69 64 19 47 46 83 Route 2: 84 6 96 91 38 95 57 22 74 23 21 39 55 80 77 78 50 65 20 32 31 11 52 36 8 Route 3: 60 61 99 59 94 100 13 15 41 73 72 67 26 24 28 79 81 71 66 70 10 62 49 48 18 89 Route 4: 17 86 93 44 37 14 87 2 53 56 4 54 3 34 35 51 30 27 63 88 7 82 45

Table 4.2: Comparison of Solutions Obtained by Different Algorithms for Various Instances

Criteria	Vehicles number	NN	ACO	PABCW
Complexity	∀	$O(k*n2)$	$O(k*n2)$	$O(nlogn+n)$
Runtime (s)	4	0.0080	0.0868	1.002
	10	0.0081	0.0987	1.013
Best Distence	4	2417.5233	1611446.6124	1309.1702
	10	4157.0434	1611446.6124	619.9003

Table 4.3: Comparison according to dataset

4.2 Results and discussion

The NN method works by finding the nearest neighbors of each customer and then creating a route for each vehicle by visiting the nearest neighbors of each customer. The optimal solution is the solution that minimizes the total distance traveled by all of the vehicles while satisfying all of the constraints. It is a simple algorithm, but it is not very efficient.

ACO is a metaheuristic algorithm, meaning it is not guaranteed to find the optimal solution. However, the ACO algorithm can often find good solutions to problems that are difficult to solve using other methods. In the case of the VRP, the ACO algorithm works by creating a population of ants. Each ant starts at the depot and randomly visits locations. The cost of traveling between locations is calculated using the distance between the locations. The ants then deposit pheromones on the paths they take. The more ants travel along a path, the stronger the pheromone trail. This encourages other ants to follow the same path. The ACO algorithm repeats this process for a number of iterations. After each iteration, the pheromone trails are updated. The stronger the pheromone trail, the more likely an ant will travel along that path.

A PABCW algorithm is a simple algorithm that makes the locally optimal choice at each step, hoping to arrive at a globally optimal solution. In the case of the VRP, the locally optimal choice is to assign the next location to the vehicle that will minimize the total distance traveled.

4.2.1 Result of 1^{st} comparison

The provided4.2 table presents the comparison results of three algorithms, where two have been modified, while the other is a proposed algorithm. The analysis indicates that the modified version of k-Nearest Neighbors (kNN) achieves a higher percentage of accuracy

compared to the other algorithms in all cases where the number of vehicles is fixed at 2 and the number of coordinates is increasing ($n20 - k2$), ($n21 - k2$), and ($n22 - k2$). Furthermore, the modified kNN algorithm also outperforms the other algorithms in a case where the number of vehicles exceeds four and the number of coordinates is more than 100 ($n101 - k4$). These conclusions are derived from the comparison against the benchmark instance, demonstrating that the modified kNN algorithm is closer to the optimal solution with an estimated percentage of 80.75%.

On the other hand, the proposed algorithm based on clark and wright, although ranked second, demonstrates an overall accuracy of 76%. It performs well in the instances $n20-k2$ and $n21-k2$ with percentages of 63% and 90%, respectively. However, in certain instances of the proposed algorithm, the results were not as favorable compared to the Ant Colony Optimization (ACO) algorithm, which approximated the optimal solution by 54.5%. For example, in the cases of $n22-k2$ and $n101-k4$, the PABCW achieved percentages of 50% and 25%, respectively, while ACO achieved percentages of 77% and 37%, respectively.

Considering these results, it can be concluded that the PABCW provides a good solution. However, the modified k-Nearest Neighbors algorithm (NN) achieves more accurate results closer to the optimal solution in our specific case.

4.2.2 Result of 2^{nd} comparison

NN

As you can see in the table 4.3, The complexity is $O(k * n2)$ where k represents the number of vehicles and n is the number of nodes, the runtime is very fast, taking only 0.0080 seconds for 4 vehicles and 0.0081 seconds for 10 vehicles. This is because the NN algorithm is able to learn the optimal routes from a training set of data. The algorithm uses a neural network to learn the relationships between the locations and the distances between them. The best distance found by the NN algorithm is also very good. For 4 vehicles, the best distance is 2417.5233, and for 10 vehicles, the best distance is 4157.0434. The reason for this improvement is that the NN algorithm is able to learn the optimal routes from a training set of data. The algorithm uses a neural network to learn the relationships between the locations and the distances between them. This allows the algorithm to find better solutions than a greedy algorithm, which only considers the locally optimal choice at each step.

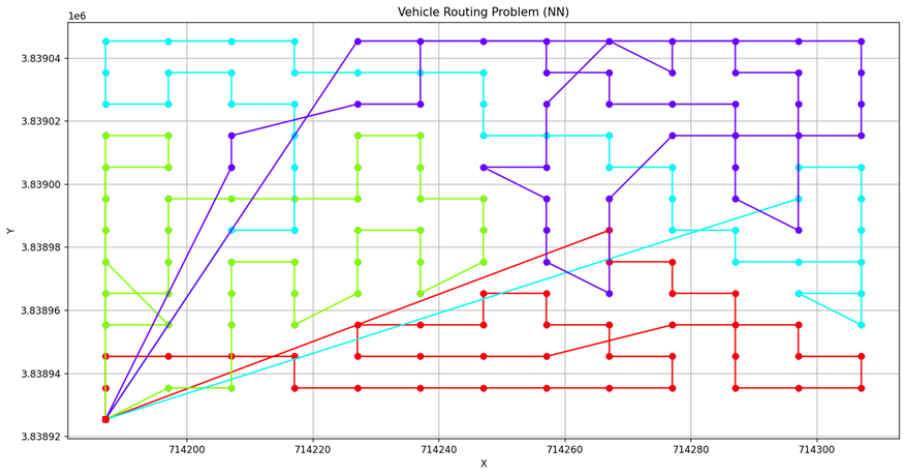


Figure 4.1: Output of NN using 4 vehicles

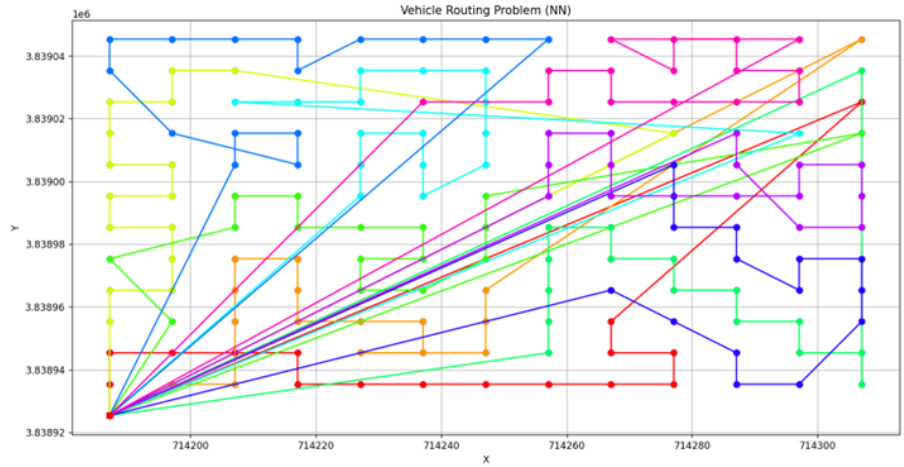


Figure 4.2: Output of NN using 10 vehicles

ACO

As you can see, its complexity is $O(k * n^2)$ where n is the number of nodes. The runtime increases slightly from 0.0868 seconds to 0.0987 seconds when the number of vehicles increases from 4 to 10. This is because the algorithm has to visit more customers when there are more vehicles. However, the best distance does not change significantly. This is because the ACO algorithm is able to find good solutions to the VRP, even for large problems. Still, it cannot find significantly better solutions as vehicles increase.
The reason for this is that the ACO algorithm is probabilistic. This means that it does not guarantee to find the optimal solution. Instead, it finds a solution that is likely to be good. The probability of finding a good solution depends on a number of factors, including the number of iterations, the number of ants, and the pheromone evaporation rate.

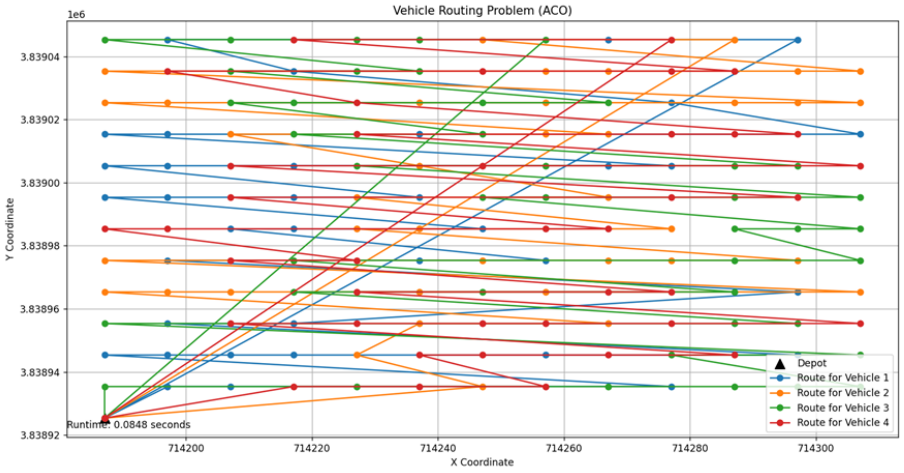


Figure 4.3: Output of ACO using 4 vehicles

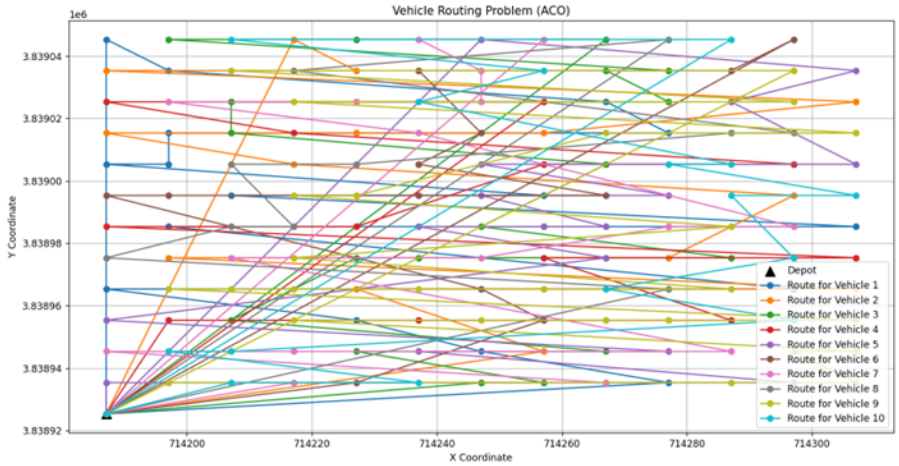


Figure 4.4: Output of ACO using 10 vehicles

PABCW

As we can see, its complexity is $O(nlogn + n)$ where n is a number of nodes, and the runtime increases slightly from 1.002 seconds to 1.013 seconds when the number of vehicles increases from 4 to 10. This is because the algorithm has to visit more customers when there are more vehicles. However, the best distance decreases significantly from 1309.1702 to 619.9003. This is because the algorithm is able to find better solutions even for large problems when the number of vehicles is increased.
The reason for this improvement is that the algorithm is able to divide the problem into smaller subproblems when there are more vehicles. This allows the algorithm to find better solutions for each subproblem, which in turn leads to a better solution for the overall problem.

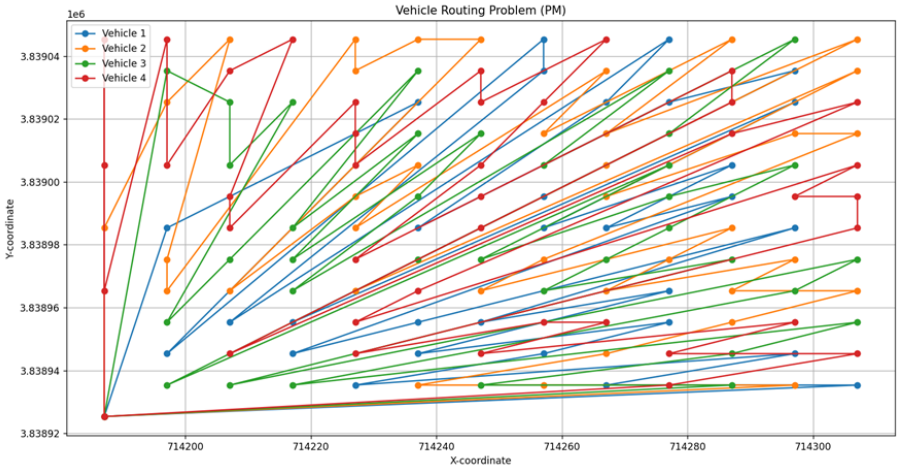


Figure 4.5: Output of PABCW using 4 vehicles

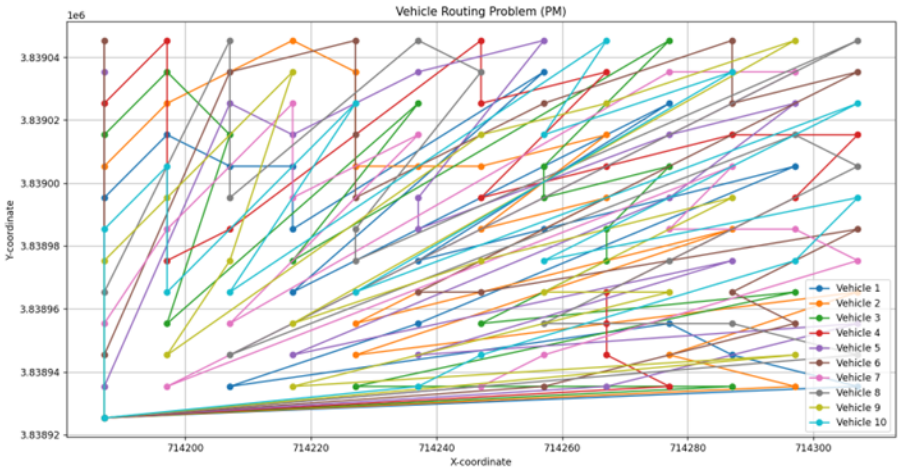


Figure 4.6: Output of PABCW using 10 vehicles

4.2.3 Discussion

Based on previous results of the 1^{st} the 2^{nd} comparison, the best method for solving VRP depends on a number of factors, including the size of the problem, the accuracy of the solution required, and the time available to solve the problem.

1. The NN algorithm has the fastest runtime, taking only 0.0080 seconds for 4 vehicles and 0.0081 seconds for 10 vehicles. It also produces significantly better distances than the greedy and ACO algorithms. Therefore, regarding runtime and solution quality, the NN algorithm seems to be the best choice for solving the VRP. Additionally, the NN algorithm has a solution quality of 80.75%, which is considered good quality. This means that the NN algorithm is able to find solutions that are within 80.75% of the optimal solution.

The NN algorithm is a good choice for solving VRP problems when the following condi-

tions are met:

- The problem is not too large.
- The optimal solution is not required.
- A fast solution is required.

The NN algorithm is not a good choice for solving VRP problems when the following conditions are met:

- The problem is large.
- The optimal solution is required.
- A slow solution is acceptable

Overall, the results show that the NN algorithm is able to find good solutions to the VRP even for large problems. The algorithm can also scale to larger problems as vehicles increase. The NN algorithm is a good choice for solving the VRP when the accuracy of the solution is essential. The algorithm is able to find good solutions to large problems, and it is very fast.
2. The ACO algorithm's runtime increases slightly as the number of vehicles increases, but the best distance does not change significantly. While the ACO algorithm can find good solutions for the VRP, it doesn't appear to find significantly better solutions as the number of vehicles increases. Therefore, the ACO algorithm can be considered if runtime is not a major concern and finding good solutions is sufficient. Additionally, the ACO algorithm has a solution quality of 54.5%, considered an average quality. This means that the ACO algorithm is able to find solutions that are within 54.5% of the optimal solution.

The ACO algorithm is a good choice for solving the VRP:

- When the problem is large and complex.
- When the problem is difficult to solve using other methods.
- When a good solution is required quickly.

The ACO algorithm is a bad choice for solving the VRP:

- When the problem is small and simple.
- When the problem can be solved using other methods quickly and easily.
- When an optimal solution is required.

Overall, the results show that the ACO algorithm is able to find good solutions to the VRP even for large problems. The algorithm can also scale to larger problems as vehicles increase. However, the algorithm cannot find significantly better solutions as vehicles increase.

3. The PABCW runtime increases slightly with more vehicles but significantly improves the best distance. This algorithm is able to find better solutions for larger problems when the number of vehicles is increased. However, the best distance achieved by the greedy algorithm is still not as good as the NN algorithm. So, if solution quality is a priority, the NN algorithm is still the better choice. Additionally, the PABCW algorithm has a solution quality of 76%, which is considered good quality. This means that the PABCW algorithm is able to find solutions that are within 76% of the optimal solution. The PABCW is a good choice for solving the VRP when the following conditions are met:

- The problem is small.
- The problem is not time-critical.
- The accuracy of the solution is not critical.

The PABCW is a bad choice for solving the VRP when the following conditions are met:

- The problem is large.
- The problem is time-critical.
- The accuracy of the solution is critical.

Overall, the results show that the PABCW is able to find good solutions to the VRP, even for large problems. The algorithm can also scale to larger problems as vehicles increase.

In conclusion, based on the given results, the NN algorithm appears to be the best method for solving the VRP due to its fast runtime and significantly better solution quality compared to the other two algorithms.

4.3 Conclusion

In conclusion, the comparative study between the Nearest Neighbor (NN), Ant Colony Optimization (ACO), and the proposed Algorithm (PABCW) for solving the Vehicle Routing Problem (VRP) reveals that the modified version of the k-Nearest Neighbors (kNN) algorithm achieves the highest accuracy percentage. Further evaluation and validation are required to confirm its effectiveness and robustness in real-world applications, considering cost savings and improved operational efficiency.

Conclusion and perspectives

General Conclusion

In this thesis, we introduce a review of VRP. Then, the characteristic of heuristic and metaheuristic methods to solving this problem, we chose one of the algorithms in each method, and we proposed an algorithm too It looks to be a straightforward application of a constructive heuristic strategy. The nodes are subsequently assigned to different routes based on their demands and vehicle capacity after being sorted according to their angle with regard to the depot. We compare these three algorithms, The primary objective of this study is to offer valuable insights into the performance and suitability of different algorithms in addressing the Capacitated Vehicle Routing Problem (CVRP). This analysis is carried out by leveraging benchmark datasets as a foundation for conducting a comprehensive comparison.

These results allow us to conclude that, while the Nearest Neighbor algorithm may be appropriate for smaller VRP instances, the proposed Algorithm offers a viable option for bigger issue sizes. Given that it can obtain considerably higher best distances, it may be able to deliver more efficient routes, which could result in financial savings and increased operational effectiveness in real- world VRP applications. To establish the efficacy and robustness of the suggested strategy, additional testing and validation on a variety of datasets and problem scenarios would be beneficial.
In conclusion, the best method for solving VRP depends on a number of factors, including the size of the problem, the accuracy of the solution required, and the time available to solve the problem.

perspectives

Considering temperature and wind as criteria in solving CVRP is crucial for achieving optimal vehicle performance. Integrating weather data and real-time updates into the algorithms can enable dynamic routing adjustments based on temperature variations and wind conditions. By doing so, the algorithms can adapt and generate routes that minimize the impact of adverse weather conditions, resulting in improved vehicle performance and more efficient delivery operations.

Accounting for criteria such as temperature and wind in algorithms for solving CVRP is essential for optimizing vehicle performance. By considering these factors and incorporating them into route planning and optimization, the algorithms can reduce fuel consumption, enhance efficiency, and ensure successful and timely deliveries even in varying weather conditions.

One perspective is to explore advanced optimization techniques and metaheuristic algorithms to enhance the algorithm's performance and address the issue of finding the

optimal solution. Another perspective is investigating the impact of different parameter settings on the algorithm's performance. Additionally, advancements in computing power and algorithms can help alleviate the computational burden of the algorithm for large-scale problems. Furthermore, incorporating real-time data and dynamic elements into the algorithm can make it more adaptable and responsive to changing conditions. In conclusion, the proposed algorithm offers opportunities for future research and improvement.

Bibliographie

[1] Mohamed Abdel-Basset, Laila Abdel-Fatah, and Arun Kumar Sangaiah. "Metaheuristic algorithms: A comprehensive review." In: *Computational intelligence for multimedia big data on the cloud with engineering applications* (2018), pp. 185–231.

[2] Anita Agárdi, László Kovács, and Tamás Bányai. "Mathematical Model for the Generalized VRP Model." In: *Sustainability* 14.18 (2022), p. 11639.

[3] Prachi Agrawal et al. "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)." In: *Ieee Access* 9 (2021), pp. 26766–26791.

[4] Munwar Ali et al. "Semantic-k-NN algorithm: An enhanced version of traditional k-NN algorithm." In: *Expert Systems with Applications* 151 (2020), p. 113374.

[5] Saman M Almufti and Awaz A Shaban. "U-turning ant colony algorithm for solving symmetric traveling salesman problem." In: *Academic Journal of Nawroz University* 7.4 (2018), pp. 45–49.

[8] Anne Auger. "Convergence results for the (1, λ)-SA-ES using the theory of ϕ-irreducible Markov chains." In: *Theoretical Computer Science* 334.1-3 (2005), pp. 35–69.

[9] Philippe Augerat et al. "Computational results with a branch and cut code for the capacitated vehicle routing problem." In: (1995).

[10] Emanuele Baldacci et al. "Social spending, human capital, and growth in developing countries." In: *World development* 36.8 (2008), pp. 1317–1341.

[11] Sunith Bandaru and Kalyanmoy Deb. "Metaheuristic techniques." In: *Decision sciences* (2016), pp. 693–750.

[12] John J Bartholdi III and Loren K Platzman. "Heuristics based on spacefilling curves for combinatorial problems in Euclidean space." In: *Management Science* 34.3 (1988), pp. 291–305.

[13] Tolga Bektas. "The multiple traveling salesman problem: an overview of formulations and solution procedures." In: *omega* 34.3 (2006), pp. 209–219.

[14] Anupam Biswas et al. "Physics-inspired optimization algorithms: a survey." In: *Journal of Optimization* 2013 (2013).

[15] Christian Blum and Andrea Roli. “Hybrid metaheuristics: an introduction.” In: *Hybrid metaheuristics: an emerging approach to optimization* (2008), pp. 1–30.

[16] Fatima-Ezzahra Boughazroun. *SOLVING VEHICLE ROUTING PROBLEM USING OPTIMIZATION TOOLS*. 2022.

[17] Der-San Chen, Robert G Batson, and Yu Dang. *Applied integer programming: modeling and solution*. John Wiley & Sons, 2011.

[18] Nicos Christofides. “The vehicle routing problem.” In: *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle* 10.V1 (1976), pp. 55–70.

[19] Jean-Francois Cordeau et al. “A guide to vehicle routing heuristics.” In: *Journal of the Operational Research society* 53.5 (2002), pp. 512–522.

[20] Farzaneh Daneshzand. “The vehicle-routing problem.” In: *Logistics Operations and Management* 8 (2011), pp. 127–153.

[21] Nathalie De Jaegere, Mieke Defraeye, and Inneke Van Nieuwenhuyse. “The vehicle routing problem: state of the art classification and review.” In: *FEB Research Report KBI_1415* (2014).

[22] J Doyne Farmer, Norman H Packard, and Alan S Perelson. “The immune system, adaptation, and machine learning.” In: *Physica D: Nonlinear Phenomena* 22.1-3 (1986), pp. 187–204.

[23] Ricardo Fukasawa et al. “Robust branch-and-cut-and-price for the capacitated vehicle routing problem.” In: *Mathematical programming* 106 (2006), pp. 491–511.

[24] Mihai Gavrilas. “Heuristic and metaheuristic optimization techniques with application to power systems.” In: *Proceedings of the 12th WSEAS international conference on Mathematical methods and computational techniques in electrical engineering*. 2010, p. 9.

[25] Zong Woo Geem, Joong Hoon Kim, and Gobichettipalayam Vasudevan Loganathan. “A new heuristic optimization algorithm: harmony search.” In: *simulation* 76.2 (2001), pp. 60–68.

[26] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. “A dynamic model and parallel tabu search heuristic for real-time ambulance relocation.” In: *Parallel computing* 27.12 (2001), pp. 1641–1653.

[27] Fred Glover. “Future paths for integer programming and links to artificial intelligence.” In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[28] David E Golberg. “Genetic algorithms in search, optimization, and machine learning.” In: *Addion wesley* 1989.102 (1989), p. 36.

[31] Çağlar Utku Güler and Murat Ermiş. "OR/MS studies on post-disaster stage relief item logistics: Complementary review." In: *Journal of Aeronautics and Space Technologies* 10.1 (2017), pp. 1–20.

[32] Rosa Herrero et al. "Solving vehicle routing problems with asymmetric costs and heterogeneous fleets." In: *International Journal of Advanced Operations Management* 6.1 (2014), pp. 58–80.

[35] Maaike Hoogeboom et al. "Erratum—Exact Algorithms for the Clustered Vehicle Routing Problem." In: *Operations Research* 64.2 (2016), pp. 456–457.

[41] Byung-In Kim, Seongbae Kim, and Surya Sahoo. "Waste collection vehicle routing problem with time windows." In: *Computers & Operations Research* 33.12 (2006), pp. 3624–3642.

[42] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. "Optimization by simulated annealing." In: *science* 220.4598 (1983), pp. 671–680.

[43] Lamiaa Korayem, M Khorsid, and SS Kassem. "Using grey wolf algorithm to solve the capacitated vehicle routing problem." In: *IOP conference series: materials science and engineering*. Vol. 83. 1. IOP Publishing. 2015, p. 012014.

[44] Vijendra Kumar and SM Yadav. "A state-of-the-Art review of heuristic and metaheuristic optimization techniques for the management of water resources." In: *Water Supply* 22.4 (2022), pp. 3702–3728.

[45] Gilbert Laporte, Martin Desrochers, and Yves Nobert. "Two exact algorithms for the distance-constrained vehicle routing problem." In: *Networks* 14.1 (1984), pp. 161–172.

[46] Gilbert Laporte, Paolo Toth, and Daniele Vigo. "Vehicle routing: historical perspective and recent contributions." In: *EURO Journal on Transportation and Logistics* 2 (2013), pp. 1–4.

[47] Pedro Larranaga et al. "Genetic algorithms for the travelling salesman problem: A review of representations and operators." In: *Artificial intelligence review* 13 (1999), pp. 129–170.

[48] Sergio Nesmachnow. "An overview of metaheuristics: accurate and efficient methods for optimisation." In: *International Journal of Metaheuristics* 3.4 (2014), pp. 320–347.

[49] José Fernando Oliveira et al. "A survey on heuristics for the two-dimensional rectangular strip packing problem." In: *Pesquisa Operacional* 36 (2016), pp. 197–226.

[50] Kevin M Passino. "Biomimicry of bacterial foraging for distributed optimization and control." In: *IEEE control systems magazine* 22.3 (2002), pp. 52–67.

[51] David Pisinger and Stefan Ropke. "A general heuristic for vehicle routing problems." In: *Computers & operations research* 34.8 (2007), pp. 2403–2435.

[52] Jacques Renaud and Fayez F Boctor. "A sweep-based algorithm for the fleet size and mix vehicle routing problem." In: *European Journal of Operational Research* 140.3 (2002), pp. 618–628.

[54] Sinan Q Salih and AbdulRahman A Alsewari. "A new algorithm for normal and large-scale optimization problems: Nomadic People Optimizer." In: *Neural Computing and Applications* 32 (2020), pp. 10359–10386.

[55] Edward Allen Silver. "An overview of heuristic solution methods." In: *Journal of the operational research society* 55 (2004), pp. 936–956.

[56] Rainer Storn and Kenneth Price. "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces." In: *Journal of global optimization* 11.4 (1997), p. 341.

[58] TO Ting et al. "Hybrid metaheuristic algorithms: past, present, and future." In: *Recent advances in swarm intelligence and evolutionary computation* (2015), pp. 71–83.

[59] Paolo Toth and Daniele Vigo. "An overview of vehicle routing problems." In: *The vehicle routing problem* (2002), pp. 1–26.

[60] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[61] Jelmer Van Ast, Robert Babuska, and Bart De Schutter. "Fuzzy ant colony optimization for optimal control." In: *2009 American Control Conference*. IEEE. 2009, pp. 1003–1008.

[64] Shaolin Wang, Yi Mei, and Mengjie Zhang. "A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems." In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2020, pp. 1–8.

[65] Xingyin Wang, Stefan Poikonen, and Bruce Golden. "The vehicle routing problem with drones: several worst-case results." In: *Optimization Letters* 11 (2017), pp. 679–697.

[67] RED Woolsey and HS Swanson. "MOST PEOPLE WOULD RATHER LIVE WITH A PROBLEM THEY CAN'T SOLVE, THAN ACCEPT A SOLUTION THEY CAN'T UNDERSTAND." In: ().

[68] Xin-She Yang and Suash Deb. "Cuckoo search via Lévy flights." In: *2009 World congress on nature & biologically inspired computing (NaBIC)*. Ieee. 2009, pp. 210–214.

Webographie

[6] Opex Analytics. *Vehicle Routing Problems 101*. http:https://medium.com/opex-analytics/opex-101-vehicle-routing-problems-262a173f4214. Accessed: 2023-06-02. (Visited on 05/08/2019).

[7] Brigham Anderson. *Simulated Annealing*. http:https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html. Accessed: 2023-05-16.

[29] *google earth pro*. http:https://www.androidpolice.com/google-earth-pro-explainer/. Accessed: 2023-05-16. (Visited on 02/22/2023).

[30] *Grid Survey - Calflora*. http:https://www.calflora.org/entry/help/gridsurvey.html. Accessed: 2023-05-29.

[33] *Heuristic Method*. http:https://www.javatpoint.com/heuristic-method. Accessed: 2023-05-02.

[34] *Heuristics vs. Meta-Heuristics vs. Probabilistic Algorithms*. http:https://www.baeldung.com/cs/heuristics-vs-meta-heuristics-vs-probabilistic-algorithms.

[36] *K-Nearest*. http:https://github.com/vivek2319. Accessed: 2023-05-29.

[37] *K-Nearest*. http:https://github.com/Akavall. Accessed: 2023-05-29.

[38] *K-Nearest-Neighbors-with-Python*. http:https://github.com/vivek2319/K-Nearest-Neighbors. Accessed: 2023-05-29.

[39] *K-Nearest-Neighbors-with-Python*. http:https://github.com/Akavall/AntColonyOptimizatic Accessed: 2023-05-29.

[40] Alaa Khamis and Yinan Wang. *Tabu Search — AI Search Algorithms for Smart Mobility*. http:https://smartmobilityalgorithms.github.io/book/content/TrajectoryAlgorithms/TabuSearch.html. Accessed: 2023-05-16.

[53] Kyle Robinson and Dr. Nancy L. Hutchinson. *Math Heuristics*. http:https://www.ldatschool.ca/math-heuristics/. Accessed: 2023-05-02. (Visited on 07/30/2014).

[57] Nunna Tejaswi. *Performance analysis on hybrid and exactmethods for solving clustered vrp: A comparative study on vrp algorithms*. 2017.

[62] *Vehicle Routing Problem*. http:https://developers.google.com/optimization/routing/vrp. Accessed: 2023-05-03.

[63] Winston Wagner. *Examples of Heuristics in Computer Science*. http:https://blog.boot.dev/computer-science/examples-of-heuristics-in-computer-science/. Accessed: 2023-05-03. (Visited on 11/30/2020).

[66] *What is Google Earth?* http:https://serc.carleton.edu/29016.1306. Accessed: 2023-05-16. (Visited on 05/28/2020).