Comparison of Heuristic Algorithms for Solving the Electric Vehicle Routing Problem

A dissertation submitted to The University of Manchester for the degree of

Master of Science in Data Science

in the Faculty of Social Sciences

Year of Submission

2022

Student ID

10854686

Candidate's School

School of Social Sciences

LIST OF CONTENTS

Word count: 13651

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Conventional Internal Combustion Engine Vehicles (IECVs) produce greenhouse gases and fine-grained particles when consuming fuel, significantly impacting the environment. Driven by environmental concerns and application considerations, electric vehicles (EVs) have received considerable attention. However, the endurance of EVs is relatively short compared to that of IECVs. Routes for EVs should be carefully planned based on existing charging stations to ensure that electric vehicles have sufficient power for transportation. That is the primary goal of the Electric Vehicle Routing Problem (EVRP). Heuristic algorithms, including Ant Colony Optimisation (ACO), Variable Neighbourhood Search (VNS), and Simulated Annealing (SA), are used to solve the EVRP. In the CEC-12 EVRP Competition, VNS received the first ranking, whose code was provided on Github. SA and ACO utilised and extended VNS's Insertion Heuristics (IH) and DBCA algorithms, combined with six Local Search operators, to improve their effectiveness. The running results of VNS were used as benchmarks to evaluate the performance of ACO and SA algorithms. These benchmarks were also used to measure the ability of three algorithms to solve the EVRP. Simulation studies show that VNS has the best performance regarding average running time, degree of algorithm convergence, and final results. SA performs well regarding running results and algorithm convergence, but its running time is the longest of the three heuristic algorithms. Due to several reasons, although the running time is short, the ACO algorithm performs poorly in the EVRP problem, with the worst final results among these three algorithms. Thus, compared to ACO and SA, the VNS algorithm may still be the most appropriate algorithm for solving the EVRP problem.

DECLARATION

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

INTELLECTUAL PROPERTY STATEMENT

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy, in any relevant Dissertation restriction declarations deposited in the University Library, The University Library's regulations and in The University's Guidance for the Presentation of Dissertations.

ACKNOWLEDGEMENTS

Foremost, I would like to pay my authentic thankfulness to my dissertation supervisor Francisco Lobo for his meticulous support. His generous assistance helped me to find hint errors and tackle complicated problems. His extensive knowledge and unparallel insight guided me in the direction of exploring and consistently helped me to avoid anxiety and hesitation. I am also grateful to him for taking time out of his busy schedule to follow my project's progress, answer my questions, and discover and discuss research deficiencies.

Furthermore, I would like to thank Mark Elliot and Elizabeth McCormick for their academic and technical support during my postgraduate studies. Because of them, I have successfully passed through every stage of my postgraduate learning.

Finally, I want to thank my family, friends, and anyone who comforted me and encouraged me when I felt down because of the obstacles in my project. Without their trust and help, I couldn't have had the strong motivation to urge me to work hard on this dissertation.

Thank you all!

1 INTRODUCTION

A series of vehicles that consume limited fossil fuels are called Conventional Internal Combustion Engine Vehicles (IECVs), including gasoline vehicles, diesel, and natural gas vehicles [1]. To some extent, the rapid development of IECVs has promoted people's life convenience and society's processes in recent decades. People can quickly get around with the widespread use of private cars, and diesel buses and vans enable the transport industry to thrive. However, the degradation of the environment in recent years has made people realise that IECVs have a negative impact on the environment. According to von Schneidemesser *et al.* [2], the unsaturated combustion of fossil fuels in IECV engines produces many nitrogen oxides and fine-grained particulate matter. These harmful substances dramatically affect the air quality. Moreover, the consumption of fossil fuels by IECVs produces many greenhouse gases, which are the leading cause of global warming situations and various extreme weather events in recent years [3].

Fossil fuel vehicles used for road transport account for a significant portion of environmental impact among the many types of IECVs. Recent research reveals that IECVs used for transportation are one of the significant contributors to greenhouse gas emissions. According to statistics from European environmental agencies, the transport sector alone accounted for 20-25% of energy consumption and carbon emissions in 2016, and Road transport accounts for three-quarters of the transport sector or nearly 16% of total energy consumption and carbon emissions (White Paper on Transport, 2011). By the end of 2020, the ratio of road transport to total energy consumption and carbon emissions had grown to almost 20% [4]. The problems of energy consumption, environmental pollution, and carbon emissions associated with fossil-based vehicles are already manifest. In order to resolve these dilemmas, governments, factories, and individuals from all walks of life must find appropriate solutions.

The introduction of alternative energy provides novel ideas for dealing with environmental problems brought about by IECV. Alternative energy replaces fossil fuels with clean energy sources such as natural gas, hydrogen, and bioenergy. Compared to fossil fuels, these clean energy sources are less polluting, emit fewer greenhouse gases, and are renewable; therefore, they have attracted the attention of many vehicle-related manufacturers. The progress of alternative energy technologies

also promotes the rapid development of AFVs (Alternative Fuel Vehicles). AFVs are a general term for a class of vehicles fuelled by clean and renewable energy [5]. They can be classified as natural gas vehicles, biofuels (ethanol), electric vehicles (EVs), hybrid vehicles (HEVs), and full cell vehicles (hydrogen). Among different kinds of alternative fuel vehicles, electric vehicles dominate the field by a wide margin for the following reasons. Pamucar [6] pointed out in his study that although natural gas AFVs technology is relatively mature and has relatively low carbon emissions as a fossil fuel, it still inevitably produces greenhouse gas. Since hydrogen vehicles produce pure water as their final product, they have an advantage in terms of cleanliness. However, Deluchi [7] pointed out that the existing technology for generating hydrogen is still in its infancy, and it is difficult to achieve mass production, which is why hydrogen fuel is expensive. Furthermore, the location, terrain, and weather conditions required for building hydrogen charging stations with existing technologies have limited the popularity of hydrogen vehicles. As a result, hydrogen is not used as the primary source of new energy for AFVs.

Unlike other types of AFVs, electric vehicles offer several unique advantages. Firstly, as a renewable and readily available energy source, electricity can easily be converted from other forms of sources, including nuclear, wind, or water energy. Utilising these energies reasonably will result in minimal environmental pollution or even no pollution. Moreover, establishing and maintaining EV charging stations are less costly than gas stations because the current urban circuit facilities are well-developed [8]. Finally, Schneidr *et al.* [9] pointed out that EVs produce less noise and are more stable during driving, costing less than IECVs traveling the same short distance.

With the gradual maturation of EV technologies, its shortcomings have also become apparent. First of all, electric vehicles still pose a significant safety risk. Tesla alone should be responsible for eight spontaneous combustion accidents in China in 2021, which has also raised concerns regarding their safety. In addition, the driving cycle and life cycle of electric vehicles are shorter than traditional cars. Gray and Morsi [10] found that since all functions of EVs are realized by electricity, the aging problem of their circuits will cause a shorter lifespan. Delucchi and Lipman [11] also investigated the battery of electric vehicles and found that electric vehicles' battery capacity will gradually decrease as the number of charging times increases. This leads to regular battery replacement becoming necessary to maintain EVs' performance, while this requirement does not apply to fossil-based vehicles. Moreover, the immature charging planning scheme is the most significant bar-

rier to the development of electric vehicles. As electric vehicles are not small toy cars, their charging methods depend on the placement of charging piles in the city. Owners of electric vehicles will experience considerable inconvenience due to too few charging stations or an uneven distribution of charging piles. Meanwhile, mao *et al.* [12] pointed out that charging times for electric vehicles are particularly long while refueling times for petrol vehicles are negligible. There is a high risk of blocking and queuing if the distribution of charging stations is uneven. As a result, once batteries run out, electric cars must wait to be charged at a nearby charging station until the battery is full-charged. Electric vehicles, therefore, are unsuitable for long-distance tasks when the planning of charging piles is not yet perfect.

Problems related to the battery and safety can only be addressed with development in the future. In contrast, several mature research plans have been made to design the routing of charging stations. The introduction of Electric Vehicle Routing Problem (EVRP) provides a theoretical basis for solving the problem of uneven distribution of charging piles or electric vehicle routing planning. The original path planning problem vehicle routing problem (VRP) was proposed by Dantzig and Ranser [13], which was used to design a reasonable path scheme for trucks transporting goods to ensure the shortest total travel distance. The VRP can be extended to EVRP by introducing specific constraints: Each electric vehicle has a fixed battery capacity and loading capacity. If the electric vehicle's battery is exhausted, it must go to the nearby charging station to replenish electricity; if the capacity is insufficient, it must return to the warehouse to load the goods. Lin *et al.* [14] clarified in their research that the EVRP could be studied in two aspects:

1. Knowing the target points and charging stations, set a reasonable path plan for the electric vehicles so that the total distance can be the shortest.
2. Knowing the target points, a range of places, and the number of charging stations, set the charging stations to reasonable sites (to make the total driving path of the electric vehicles shortest).

This project focuses only on the first case, which involves planning reasonable paths for electric vehicles in the event charging stations are placed.

Heuristics can be defined as previous experience that facilitates solving problems and determining probability [15]. In Burke *et al.*'s [16] definition, heuristic algorithms aim at finding the optimal

solutions around a feasible solution. Three heuristic algorithms were implemented in this dissertation: the Simulated Annealing (SA) algorithm, the Ant Colony Optimization (ACO) algorithm, and the Variable Neighbourhood Search (VNS) algorithm. Although they all fall within the category of heuristic algorithms, their internal logic differs significantly. Therefore, this project will implement these algorithms on the same data sets and compare their performance through the results.

Using heuristic algorithms to solve the EVRP is the primary aim of this project. However, in 2020 Mavrovountiotis designed a competition for this problem, calling for possible algorithms and providing a dataset and a basic framework for the code. Although the code details of the winning algorithms have not been released, their final results have been displayed on Marovountiotis's website. The implementation framework of three algorithms was based on Mavrovountiotis's code framework, and the datasets he provided were used as the testing dataset. After implementing these three algorithms, the testing results were compared with the winning algorithms' results provided on Mavrovountiotis's website to verify the effectiveness of the implemented algorithms. It is worth noting that the code for this project is not entirely original. One of the winning algorithms, VNS in the Mavrovountiotis competition, has been found on GitHub, and SA and ACO parts used some of the VNS's functions to extend their efficiency. The core of this project is understanding the EVRP and existing VNS code, innovation, extension, and integration of the code part. Also, it is necessary to understand the advantages and disadvantages of each algorithm and explore their performance on the EVRP problem.

The remainder of this report is organised as follows: Section 2 provides the literature review for this project. Section 3 presents the model for the Electric Vehicle Routing Problem and the representations and meanings of its solutions. Section 4 introduces the methodology part, including the implementation of ACO and SA, the details for the DBCA, and Insertion Heuristics algorithms learned from the existing VNS code. Section 5 reports the computational experiments and compares the performance of three algorithms. Finally, it discusses the conclusion for this project, the reflections on the experimental procedure, and the outlook for future tasks.

2 LITERATURE REVIEW

People often abstract real-life difficulties into problems with specific constraints. The ideas or methods to solve these problems are called algorithms. Algorithms vary in complexity, which is usually reflected in time complexity. Given the input data size of n and some non-negative integer k, if an algorithm can solve the problem in $O(n^k)$ time complexity, it is said to be solvable in polynomial time [17]. Meanwhile, algorithms can be classified as deterministic or non-deterministic based on their final results. Regardless of how many times an algorithm is run, a deterministic algorithm produces the same result every time. For non-deterministic algorithms, different executions of a particular input may result in different outputs. The property of non-deterministic is related to its inherent degree of randomness. NP-Hard problems can be described as a set of problems that are difficult to solve in polynomial time with non-deterministic algorithms. Numerous NP-Hard problems, including circuit satisfiability, set cover, vertex cover, and traveling salesman problems, are being explored by researchers. The Vehicle Routing Problem (VRP) and its variants are a series of problems related to road planning. They are all complex NP-Hard problems that are difficult to solve in polynomial time by simple linear algorithms.

According to Dantzig and Ramser [13], the VRP is a classic NP-Hard road planning challenge designed to solve the vehicle transportation problem in daily life. It abstracts the transportation problem into a simple vehicle-cargo transportation problem. The VRP consists of a fleet of vehicles, a warehouse, and multiple target points. It has three restrictions in its initial format:

1. A vehicle cannot visit the same target point.
2. A target point can only be accessed by one vehicle.
3. All target points must be visited.

The VRP aims to find the most reasonable path planning so that the entire fleet has the shortest driving path. However, the VRP cannot generalise all practical problems because practical situations have many restrictions on vehicles, traveling time, and traveling procedures. So, according to the report of Mor and Speranza [18], by adding specific constraints called variants, the VRP can be extended into problems with specific functions. For example, the CVRP, known as the Capacitated

Vehicle Routing Problem, sets the capacity of vehicles as a critical limit to satisfy the requirements of the transportation industry [19]. If a vehicle does not have enough cargo remaining to satisfy its customers, it must return to the warehouse for resupply and continue its transportation mission. Based on the VRP, the Vehicle Routing Problem with Time Window (VRP-TW) incorporates unloading and transportation time considerations [20]. It aims to meet the needs of its customers, deliver goods within the shortest timeframe, and travel the shortest distance.

The VRP is designed explicitly for fossil fuel vehicles, whose constraints are relatively simple. As electric vehicles have become more prevalent, road planning problems have become more complex and challenging. The Electric Vehicle Road Problem (EVRP) adds additional restrictions on vehicles based on Vehicle Road Planning (VRP) for electric vehicles [14]. It stipulates that each electric vehicle has a maximum load capacity and a maximum battery capacity, which limits the trajectory of the electric vehicle's operation. This means that vehicles cannot move as freely as they did in VRP. In the EVRP, if the remaining load capacity cannot meet the requirements of any customer, it can only return to the warehouse for replenishment; If the remaining capacity cannot support the travel to the next target point, it can only find the nearest charging pile for electricity charging. The most variant of the Electric Vehicle Routing Problem (EVRP) is Electric Vehicle Routing Problem with Time-Window (EVRP-TW). Li *et al.* [21] stated that EVRP-TW considers all time-related factors, including vehicle operating time, EV charging time, loading time and unloading time. Based on a constant electric vehicle speed, the EVRP-TW's sole objective is to complete its missions in the shortest time possible.

A VRP-like problem can take various forms, but the core idea is always the same. Dantzig and Ramser [13] introduced the basic process of solving the VRP in detail: Firstly, an initial solution should be generated using the greedy, random search or other algorithms. Then, existing constraints of VRP are used to judge whether the initial solution is reasonable. Finally, a new solution is generated based on the initial solution. These three steps are repeated until the desired result is obtained. A more complex variant, such as the EVRP, can be solved by changing the details of the problem generation and constraints check while the overall framework can remain stable. Due to the generality of the VRP and its variants, we can take inspiration from any algorithms that solve them, regardless of individual differences.

Local Search (Neibourhood Search) is a traditional approximate algorithm for solving problems [22]. It starts from an initial solution and then searches its neighbourhood. If a solution with better performance is found, Local Search updates the searching neighbourhood and performs searching around this better-performing solution. Generally, several restrictions, such as the maximum number of iterations, are added to the Local Search algorithm to control the depth of searching. If the constraints are violated, the algorithm will stop running, and the locally optimal solution is obtained [23]. Local Search can be used to solve the VRP, EVRP, and their variants. Shaw [24] proposed to use the Large Neighbourhood Search (LNS) method to solve the VRP with several constraints. In his design, target points in a solution are selectively removed when performing the LNS, and then a scheme is devised to reinsert these target points [24]. He proved that reasonable design of selection and insertion methods could enable the algorithm to traverse more excellent solutions and improve the possibility of finding the optimal solution [24]. It is also possible to use this method to adjust the access order for the charging piles and customers in the EVRP. Kilby *et al.* [25] proposed an improved version of the Local Search scheme, called Guided Local Search (GLS), to generate more possible solutions by using various crossover operators in local searches, such as 2-opt and 3-opt. At the same time, he also planned to divide the solution into multiple blocks and performed random combinations to realise the reconstruction of the solution.

Research has also led to significant changes in the Local Search algorithm due to the deepening of research. People begin to combine Local Search methods with some mature algorithms to adapt to more complex VRP variants. The Iterative Local Search (ILS) is the innovative product of the Local Search. Lourenço *et al.* [26] defined the Iterative Local Search as: a metaheuristic that integrates mature heuristics into local search iterations to produce new solutions to the problem. Silva *et al.* [27] applied the ILS to a variant of the VRP: Split Delivery Vehicle Routing Problem (SDVRP). SDVRP stipulates that the cargo requirements of each customer are not necessarily transported at one time and can be completed by the fleet in batches. Silva *et al.* [27] proposed a multi-start ILS scheme, which achieved remarkable results in the SDVRP problem. This scheme allows one recursion to generate incomplete solutions, and new solutions can continue to iterate based on these incomplete solutions until a complete solution is generated. Penna *et al.* [28] introduced an ILS-RVND algorithm, which utilised the Random Variable Neighbourhood Descend (RVND) in the descend procedure of iterated local search (ILS). RVND randomly shuffles existing solution elements

to generate new solutions. At the same time, in the iterative process, RVND irregularly perturbs the previously visited optimal solutions to obtain possible better solutions. Introducing a large amount of randomness allows the ILS-RVND algorithm to traverse more excellent solutions. However, there is also a certain probability that each round's solutions may differ. Because the ILS algorithm may perform many useless searches in similar solution regions each time, Brandão [29] proposed a more imaginative iterative search scheme. He introduced additional memory during the iterative gradient descent of ILS to record poor search results. After the following search yielded these results, the algorithm jumped from the current searching space to other neighbourhoods. Such an algorithm is called a memory-based iterated local search algorithm. The effect of this idea has been tested on the multi-start VRP problem by Brandão, but it is not unsuccessful. Brandão [28] reflected that the reason for the poor performance of the algorithm might be that the stored strategy was not smart enough.

In Local Search, the objective function value continuously changes until the optimal local value is reached by performing a series of local changes within a fixed range [30]. The area over which local search are performed is over referred to as neighbourhood. Hansen *et al.* [31] pointed out that many algorithms related to the Local Search have particular extensions to it in order to prevent getting trapped in the local optima. The Variable Neighbourhood Search (VNS) consists of multiple Local Search neighbourhoods to obtain possible solutions by systematically varying the neighbourhoods within the local area [30]. It does not follow any search trajectory but explores farther and farther neighbourhoods. It usually does a fine-grained search in a neighbourhood, continuing to search if the results are consistently good. However, if the search result is poor, or there is a possibility of falling into a local optimum, it will skip to other neighbourhoods to search. Kytöjoki *et al.* [32] successfully implemented the variable algorithm on a large-scale VRP problem and made significant progress. They found that VNS can be used to guide standard local searches. Moreover, in order to adapt the algorithm to the ultra-large-scale vehicle routing problem, Kytöjoki *et al.* [32] designed a VNS algorithm with high execution speed and low memory usage. Wassan *et al.* [32] studied a new VRP variant, which is called MT-VRPB (Multiple Trip Vehicle Routing Problem with Backhauls). They designed a two-layer VNS model for this problem, embedding sequential VND and a multi-layer Local Search approach in an iterative process. According to Wassan *et al.* [32], this two-layer model performed well in MT-VRPB, and the running time was not significantly differ-

ent from that of traditional VNS. Rezgui *et al.* [33] applied the VNS algorithm to the electric vehicle routing problem, and they compared the performance of VNS with other heuristic algorithms, such as Tabu search and Genetic algorithm. He found that when the algorithms were run on the same dataset, VNS ran longer than the other algorithms but performed better. The possible reason was that the VNS search was more detailed in a neighbourhood [33].

In addition to innovating the Local Search methods, scholars have tried to combine multiple heuristic algorithms with the Local Search to get better results. Gendreau *et al.* [34] combined Tabu Search with a simple Local Search operator to solve the EVRP. However, the final result of his experiment was not ideal. He reflected that the internal iteration of the algorithm was too simple, and the regions with excellent solutions were not effectively covered [34]. Taillard *et al.* [35] improved Tabu Search's internal logic and introduced the edge exchange concept in the Local Search part. The edge exchange logic randomly generates two new solutions by shuffling, exchanging, and recombining two solutions. According to the results of Taillard *et al.* [35], edge exchange can significantly improve the effect of tabu search. Sassi *et al.* [36] innovatively combined the concept of Iterative Local Search (ILS) with the Tabu Search and achieved good results. This newly generated Tabu Search algorithm called Iterated Tabu Search (ITS), is used to solve more complex electric vehicle problems with multi-electric fleets.

Besides tabu search, the Simulated Annealing algorithm can also be used as an efficient heuristic algorithm to solve VRP an EVRP problems. It controls the extent of the search in the solution's neighbourhood by simulating a temperature drop. Vincent *et al.* [37] used a simplified Simulated Annealing algorithm model with Random Local Search to solve the EVRP. In order to traverse more solutions, he proposed four random operators:

1. Swap: Randomly choose two solutions in the solution set, pick a customer, and then swap the customer's position in the two solutions.
2. Insertion: A customer is arbitrarily removed from a solution and then inserted into other positions.
3. Insert_AFS: In a solution containing only customers, insert a charging station at random.
4. Delete_AFS: Delete a charging station at random in a solution containing both customers and charging stations.

According to Vincent's experimental results [37], the Simulated Annealing algorithm with the Random Local Search performed relatively well, and the optimal solution can be found on data sets with low complexity. However, when it comes to large-scale datasets, its performance lagged far behind VNS. Rodríguez-Esparza *et al.* [38] introduced the concept of reinforcement learning in the Simulated Annealing algorithm. According to his theory, it is not necessary to pay attention to the details of each parameter but rather to its general range of values. The Simulated Annealing algorithm retains its best results at runtime and adaptively changes its parameters in the iterative process to search for solutions more efficiently [38]. Rodríguez-Esparza *et al.* [38] implemented these principles on a variant of the EVRP, Capacitated Electric vehicle routing problem (CVRP). They achieved much progress: The overall training time was effectively shortened, and the performance was significantly improved when working on complex data sets.

Algorithms such as Local Search are algorithms designed through mathematical logic, while algorithms such as Simulated Annealing and Tabu Search are algorithms abstracted by imitating physical principles. Apart from that, some algorithms are bionic algorithms abstracted by people by observing natural phenomena. These algorithms highly imitate certain natural features and are of great help to the results of the problem. Inspired by the ants foraging behaviours, the Ant Colony Optimisation significantly restores ants' ability to find food, release pheromones, and profit from group dynamics [39]. Bullnheimer *et al.* [40] tried to use the improved ant colony algorithm to solve the traditional vehicle routing problem (VRP). However, on fourteen benchmark datasets, the performance of the ant algorithm was significantly inferior to that of better iterative algorithms such as Variable neighbourhood search (VNS) and simulated annealing (SA). The author reflected that the process of generating new solutions was too simple with poor randomness. It leads to insufficient coverage of possible reasonable solutions. Because the ant algorithm is a swarm algorithm, such an impact will cause all subsequent solutions to be ineffective. Ding *et al.* [41] discussed the shortcomings of the ACO algorithm, such as premature convergence, slow search speed, and poor search depth. He then introduced a hybrid Ant Colony Optimisation (HACO) schema, which introduced a λ-based disaster operator to adjust the depth of searching and convergence time. Because in the actual transportation task, vehicle congestions may inevitably occur, reducing the transportation efficiency. Yin and Chuang [42] therefore designed an Adaptive Memory Artificial Bee Colony (AMABC) algorithm, which utilised the Tabu Search algorithm to manage

the conditions of each road capacity to reduce or even avoid congestion on transport vehicles. Lin *et al.* [43] improved the ACO algorithm by adding additional constraints on the current searching problem, allowing it to solve the Multi-Depot Green Vehicle Routing Problem (MDGVRP) by maximising revenue and minimising costs.

This literature review presents the background knowledge of VRP and its variants in detail. At the same time, the algorithms for solving the VRP problem can be extended and improved to solve more variants, such as VRP-TW, EVRP, and EVRP-TW. Moreover, previous researches on the Local Search, Tabu Search, Simulated Annealing algorithm, Variable Neighbourhood Search, and Ant Colony Optimisation are discussed. Their excellent perspectives will be incorporated and used to solve the existing EVRP in this project. Lastly, the ACO, VNS, and SA algorithms will be implemented in this project, giving them new meaning, and making them practical tools for solving the EVRP.

3 ELECTRIC VEHICLE ROUTING PROBLEM MODEL

The EVRP can be simplified to the model shown in Fig. 29. It consists of four essential elements: Electric Vehicles (EVs), the Depo, Customers (target points), and Charging Stations. An EV fleet is a collection of homogeneous electric vehicles with the same loading and battery capacity. They shuttle between charging stations and target points to perform road transport tasks. The Depo can be regarded as a warehouse for goods and a charging station. All EVs should depart from the Depo, fully loaded and fully charged, and finally return to Depo after all tasks have been performed. Each Customer (target point) has a certain amount of cargo requirements. Only when their requirements are fully satisfied will the problem be solved entirely. The following situations generally occur during the traveling of electric vehicles:

1. According to the algorithm, when the EV has sufficient power and carries enough cargo, it will deliver the cargo to the appropriate customer.

2. If the EV does not have enough power to reach the next customer, it finds a charging station that its battery can support and is relatively close to charging and then continues its tasks.

3. Suppose the remaining goods of an EV cannot meet the requirements of the target customer.

The algorithm will determine how the electric vehicle should proceed: whether it should deliver the goods to possible other customers or restock at the Repo.

The EVRP may have many variants, which are significantly different in the constraints for the problems. However, this project only considered the simplest case, with two constraints on the EVs. While Other factors, such as the time for recharging and loading and unloading, were excluded from the problem.

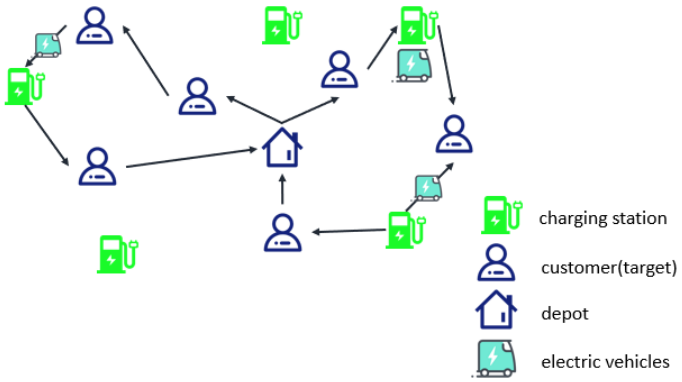


Fig. 1. The EVRP RoadMap

Suppose in the EVRP there are m electric vehicles, each of which has a maximum loading capacity Q and a maximum energy capacity C. Symbol V stands for n customers, which can be represented as $V = \{1, 2, ..., N\}$ or $V = \{v_1, v_2, ..., v_n\}$. $F = \{n+1, n+2, ..., n+s\}$ or $F = \{f_1, f_2, ..., f_s\}$ is a set of s charging stations. As electric vehicles always start from the Depo and return the Depo, let 0 denote the departure from depo and $N+1$ denote return depo, which is abbreviated to $Depo = \{0, N+1\}$. The collection of vertices of customers and charging stations can be represented as $V' = V \cup F$. Thus, a set of vertices, which contains all the customers and the depo, can be represented as $V_0' = V' \cup \{0\}$ and $V_{N+1}' = V' \cup \{N+1\}$. The Depo, customers and charging stations are different nodes existing on a graph, which are connected by a few edges or arcs. The arcs are represented as $L = \{(i, j) \,|\, i, j \in V_{0,N+1}', i \neq j\}$. After knowing the basic elements, the EVRP instance, containing customers, charging stations and the Depo, can be abstracted as a fully connected weighted graph $G = (V_{0,N+1}', L)$.

Arcs of the EVRP have practical significance, representing the connection between two vertexes. In the actual problem, they have a certain length, represented by the symbol D. The distance between vertex i and j can be defined as $D = \{d_{ij} \,|\, i, j \in V_{0,N+1}', i \neq j\}$. The average speed of an

electric vehicle on the arc L_{ij} is v_{ij}, and the time for driving on the arc is t_{ij}. Under normal circumstances, the speed of vehicles on the road is irregular. Their acceleration and speed may vary from time to time. However, in order to simplify the operating mode of the vehicle, the EVRP assumes that the electric vehicle is in the form of a constant speed, which can be represented by v. In this case, the consumption rate of battery power per unit distance can also be considered constant, which can be set to r. The following two formulas can be derived from the above assumptions and settings:

$$t_{ij} = d_{ij}/v \tag{1}$$

$$Con_{ij} = r \times d_{ij} \tag{2}$$

where t_{ij} represents the time it takes for an electric car to travel on the $arc(i,j) \in L$ and Con_{ij} represents the total energy consumption.

Each customer $i \in V$ has a corresponding demand for goods, which can be expressed as δ_i. When an electric vehicle numbered k is moving on $arc(i,\ j)$, its current remaining loading can be represented as q_{ij}^k. The reasonable range for the current remaining loading should be $0 \leq q_{ij}^k \leq Q$. Meanwhile, the current remaining battery volume can be represented as c_{ij}^k. The reasonable range for the current remaining battery energy left should be $0 \leq c_{ij}^k \leq C$. It is worth noting that because this EVRP model does not take into ac- count the charging time and the time to unload the loaded cargo, the charging and the transfer of the cargo are completed instantly. After an electric car reaches the i-th vertex and completes the task at this location (unloading cargo or recharging), the remaining cargo volume is μ_i and the remaining electricity is σ_i. In addition, there are two binary variables that determine whether the vehicles reach customers or charging stations. x_{ij}^k is a binary decision variable, which decides whether an electric vehicle k visited customer j immediately after i. The value for this variable is as follows:

$$x_{ij}^k = \begin{cases} 1, \text{ if EV } k \text{ visited the customer } j \text{ immediately after } i, \\ 0, \text{ otherwise}, \end{cases} \tag{3}$$

y_{ij}^k is another binary decision variable, which decides whether an electric vehicle k visited charging

station i. The result for this variable is shown in Eq. (4):

$$y_i^k = \begin{cases} 1, & \text{if EV } k \text{ visited the charging station } i, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

Table 1 summarised all the previously mentioned variables and notations related to the Electric Vehicle Routing Problem.

The objective of the EVRP is to find the minimum route for EVs to meet the customers' demand under the constraints of load capacity and battery power capacity. A feasible solution is a routing chain where the Depo is at both ends. Customers must be included once in the middle part of the solution, while the number of the Depo and charging stations will vary depending on the situation. A wide range of solutions can be generated by permutations of customers and charging stations to achieve the desired result. The solution for the EVRP can be evaluated by the formulation as below:

$$\min \phi(\pi) = \sum_{i \in V'} x_{0i} d_{0i} + \sum_{i,j \in V'} x_{ij} d_{ij} + \sum_{j \in V'} x_{j0} d_{j0} \tag{5}$$

subject to

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} = 1, \forall i \in V \tag{6}$$

$$\sum_{i \in V'_{N+1}, k \in \{0,\ldots,m\}} {x_{i0}}^k = m \tag{7}$$

$$\sum_{i \in V', l \in N \cup F} {x_{il}}^k = \sum_{i \in V'_{N+1}, l \in N \cup F} {x_{li}}^k \tag{8}$$

$$\sum_{i \in V, j \in V_{N+1}, k \in M} \delta_i x_{ij}^k \leq Q \tag{9}$$

$$e_i^k \geq \min\{\mu_{i0}^k, (\mu_{ij}^k + \mu_{j0}^k)\} \forall i \in N, \forall j \in F, \forall k \in M \tag{10}$$

$$x_{ij}^k \in \{0, 1\} \forall i \in N, \forall j \in F, \forall k \in M \tag{11}$$

Table 1. Notations in the Model Formulation

Graph Sets:

n	Number of customers;
m	Number of electric vehicles;
s	Number of charging stations;
V	Set of customers;
F	Set of charging stations;
G	The fully connected weighted graph;
0	The starting depo;
$N+1$	The ending depo;
V'	Set of charging stations and customers;
V'_{N+1}	Set of charging stations, customers and ending depo;
V'_0	Set of charging stations, customers and starting depo;
$V'_{0,\,N+1}$	Set of all vertices, including starting / ending depo, customers and charging stations;

Parameters:

L	Set of arcs / edges;
L_{ij}	The arc / edge between vertex i and j;
D	Set of distances between vertexes;
d_{ij}	The distance between vertex i and j;
v_{ij}	The traveling velocity of the electric vehicle between i and j;
v	The average velocity of the electric vehicle in the whole EVRP;
t_{ij}	The travel time between vertex i and j;
Q	Maximum battery capacity of electric vehicles;
C	Maximum load capacity of electric vehicles;
r	The average battery consume rate;
δ_i	The demand of customer i;
μ_i	The remaining cargo volume of an electric vehicle when reaching the vertex i;
σ_i	The remaining battery electricity of an electric vehicle when reaching the vertex i;

Decision variables:

q_{ij}^k	The battery level of electric vehicle k when it departures from vertex i to j;
c_{ij}^k	The cargo level of electric vehicle k when it departures from vertex i to j;
x_{ij}^k	if vehicle k immediately visited vertex j after visiting vertex i, $x_{ij}^k = 1$, other wise, 0;
y_i^k	if vehicle k visited charging station i, $y_i^k = 1$, other wise, 0.

$$y_i^k \in \{0,1\} \forall i \in N, \forall k \in M \tag{12}$$

Eq. (5) defines the objective function of the EVRP, and the unit of the output value of this function is distance. Eq. (6) guarantees that a customer can only be visited once during the entire problem resolution process. According to Eq. (7), an EV must start from the Depo and eventually return to the Depo, regardless of how it performs its mission. The specific behavior of electric vehicles is limited by Eq. (8). EVs are not permitted to remain at the location of customers or charging stations; they must set off immediately after completing their current task or charging requirements. Eq. (9) puts a limit on the capacity of vehicles, which means that the current EV's load capacity cannot exceed the total load capacity under any circumstances. EVs cannot always travel and must plan for subsequent trips; Eq. (10) ensures that EVs have enough energy to return to the Depo. Eq. (11) and Eq. (12) provide a more detailed description of the binary variables mentioned in Eq. (16) and Eq. (4).

3.1 Illustration of Solution Representation

The mathematical representation of the EVRP in the previous section is described in detail. The result obtained by utilising a series of algorithms solving the EVRP is called the solution to the EVRP. Since the EVRP problem is complex, numerical coding is used to represent different objects, such as 0 for the Depo and other natural numbers for charging stations and customers. Fig. 2 illustrates a possible route for the Electric Vehicle Routing Problem with ten customers and four charging stations. Objects with a number between 1-10 represent customers, whereas objects with a number greater than 10 represent charging stations, and 0 represents the Depot. The three primary elements of the EVRP are attached with different shapes to facilitate better identification. The Depo is symboliseed by a square, the customers by a circle, and the charging stations by a diamond.

A solution represents the order in which the electric vehicle accesses each EVRP element, which can be seen in the upper table in Fig. 2. Solutions may consist of multiple routing paths; each can be described as starting from 0 and ending at 0. For example, the first route path in the Fig. 2 can be represented as $0 \rightarrow 4 \rightarrow 5 \rightarrow 11 \rightarrow 1 \rightarrow 3 \rightarrow 12 \rightarrow 0$, which passes four customers 4, 5, 1, 3, and two charging stations 11 and 12 in turn. The solution shown in Fig. 2 contains three vehicle

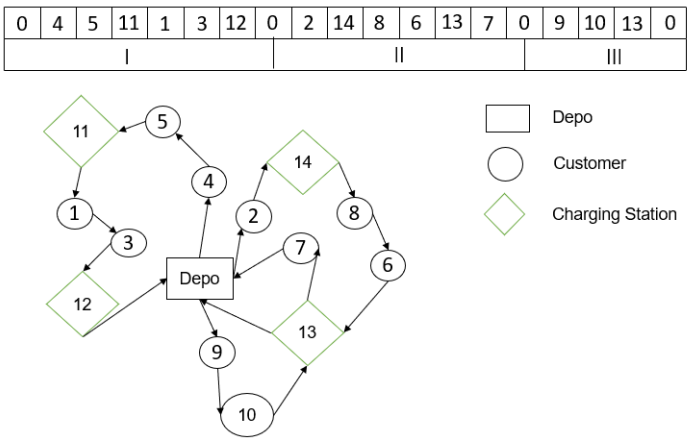


Fig. 2. A Feasible EVRP Exemplary Solution

routes. In addition, combined with the practical situation, it has two practical significance:

- It will take three round trips to complete all the tasks if only one electric vehicle exists.
- Since the electric vehicles fleet can perform tasks synchronously, this solution requires at least three vehicles to accomplish the task within the minimum amount of time.

4 METHODOLOGY

Understanding the problem's essence is only the project's first step. The following subsections will explain how algorithms solve the EVRP and the principles behind it. The Variable Neighborhood Search (VNS) algorithm has been implemented in the (github). Based on the advanced algorithm ideas in the VNS, this project makes improvements and applies them to the implementation of Ant Colony Optimisation algorithm and Simulated Annealing algorithm.

4.1 Insertion Heuristics

The optimisation of the road planning problem begins with the solution of the Traveling Salesman Problem (TSP). In the traditional TSP, the objective is to determine in what order all customers should be visited so that the distance traveled is minimised [44]. Since non-repetition access is the only limitation, the solving algorithms are relatively diverse. A standard method is placing all

customers in an array and shuffling or permutating them using multiple Local Search methods to find better solutions. However, compared with the TSP, the EVRP is much more complicated, with restrictions on the battery capacity and loading capacity of the EVs. As a result of these restrictions, the EVs cannot move forward continuously, and they are required to return to the Depot frequently for replenishment. Therefore, it is not feasible to directly imitate the TSP's method to initialise EVRP's solution for two reasons:

1. Not all charging stations will appear in the solution set. If all the charging stations are added to the solution at the beginning, it will lead to frequent deletion operations, which are highly inefficient.

2. If a reasonable strategy is not adopted to generate a solution, the probability that the solution meets the EVRP constraints will be low, making it difficult for the algorithm to converge.

Considering these deficiencies, a more efficient method for assisting the generation of solutions is devised, which is called Insertion Heuristics (IH). The details of the Insertion Heuristics algorithm are shown in Algorithm 1. It is worth noting that we assume that labels 1-8 are customers, and those with labels above 8 are charging stations.

Before using the IH algorithm, a solution containing only all customers needs to be generated, called the Like TSP (L-TSP) solution. The initial Like TSP solution can be represented by π^{ltsp}. Its form is very similar to that of the traditional TSP solution, with only one difference: The traditional TSP solutions contain all customers and do not permit repetitions; while an L-TSP solution requires the EV to start from the Depo and finally back to the Depo, which causes repetition at the starting and ending points.

The first step of the IH algorithm is to consider the load capacity limitation of electric vehicles and convert the L-TSP solution into a solution without charging stations. The newly generated solution can be called a non-recharging (NR) solution and is represented by π^{nr}. The Effect of this step is shown in Fig. 3. The IH algorithm sequentially checks whether the remaining capacity of the EV can satisfy the next customer on the L-TSP solution. If the remaining goods of an EV can satisfy the next customer, the checking process continues; otherwise, 0 is inserted into the position where the next customer will not be satisfied. For example, when the EV in Fig. 3 passes the route $0 \rightarrow 1 \rightarrow 3$

and its remaining goods cannot satisfy the next customer 7, 0 is added between customer 3 and customer 7. Adding 0 is just a numerical representation of the solution operation. Essentially, it means that one EV can only return to the Depo due to insufficient remaining goods, and another takes over the transportation duties. Fig. 3 shows the process of converting an L-TSP solution to an NR solution by the IH algorithm.

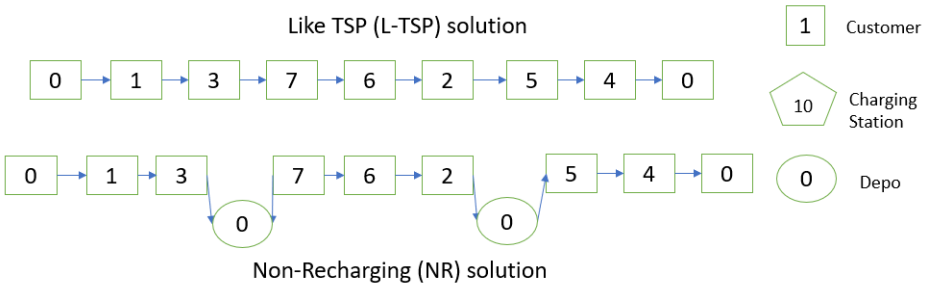


Fig. 3. The IH Algorithm Converts an L-TSP Solution to an NR Solution

The NR solution π^{nr} is not yet feasible since it does not consider the battery capacity limitations of the EVs. In this step, the IH algorithm will take into account the limitations of EVs' battery and convert the solution π^{nr} into a feasible solution π^{fea}. Traversing each point in the NR solution determines the furthest customer to which the electric vehicle has enough power to provide service. The furthest customer that can be visited in each route path can be marked as $P_{furthest}$. Charging station's selection strategies for each EV are based on the location of the customer $P_{furthest}$. The proper charging stations' selection will generate two situations, which will be explained in conjunction with Fig. 4.

1. The IH algorithm will find the closest charging station $F_{closest}$ to point $P_{furthest}$. $F_{closest}$ will be arranged after $P_{furthest}$ in the solution, if it is confired that the electric vehicle can reach $F_{closest}$. In Fig. 4, customer number 7 is $P_{furthest}$ and charging station numbered 10 represents $F_{closest}$. After determining that the remaining power will allow the EV to drive from customer 7 to charging station 10, $F_{closest}10$ is placed directly behind the $P_{furthest}7$.

2. Suppose that an EV cannot reach the nearest charging station $F_{closest}$. In this case, the IH algorithm will roll back in the solution: the current $P_{furthest}$ will be changed to the previous point of $P_{furthest}$ in the path and perform the first Scenario. In Fig. 4, we suppose that customer 6 is $P_{furthest}$ set at the beginning. However, it cannot reach the nearest charging point

because of the battery capacity limit. Therefore, customer 7 is selected as $P_{furthest}$. By using the idea in the first situation, a feasible solution will be generated.

Lastly, the IH algorithm will repeat the above operations until all the route paths of the original NR solution have been traversed, and all the load and battery capacity constraints have been satisfied. After the operation of the IH algorithm, the final solution π^{fea} is a feasible EVRP solution.

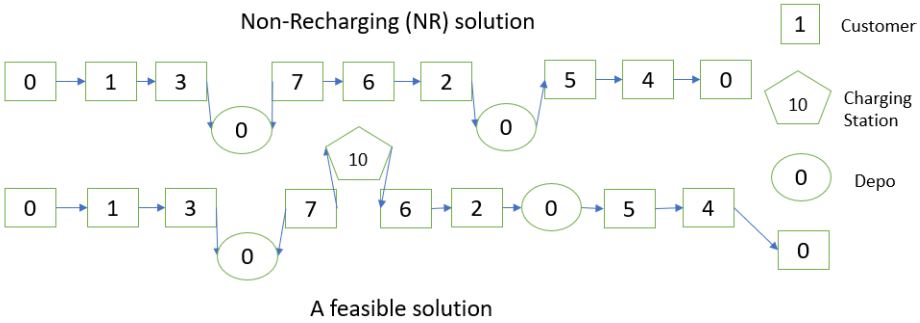


Fig. 4. The IH Algorithm Converts an NR Solution to a Feasible EVRP Solution

4.2 Density Based Clustering Algorithm

The Variable Neighbourhood Searching (VNS), Ant Colony Optimisation (ACO), and Simulated Annealing (SA) begin by generating an initial solution. The quality of the initial solution will directly affect the final result, so an intelligent solution generation method should be developed. As a linking algorithm, the Insertion Heuristics (IH) can be used to generate a feasible solution. However, it cannot create a solution independently since it requires an existing L-TSP solution as input. This section combines Density-Based Clustering with the IH algorithm to produce an excellent initial solution.

The Density-Based Clustering algorithm (DBCA) performs well on multidimensional clustering and it is insensitive to noise [45]. It uses distance as the primary basis to divide multiple points in space into specific clusters. The DBCA can also cluster customers and charging stations since they can be considered scattered points in a two-dimensional space. Each charging station can be regarded as the cluster's center, and customers are distributed to the nearest cluster based on their distance to the charging stations. Customers in a cluster will preferentially choose the charging station in the cluster for charging. The complete algorithm for the DBCA is shown in Algorithm 2.

Algorithm 1 Insertion Heuristics

Input: An infeasible L-TSP solution π^{ltsp}
Output: A feasible solution π^{fea}

1: %Convert the L-TSP solution to the NR-solution
2: Clear(π^{nr})
3: **for** Every node P in π^{ltsp} **do**
4: **if** P is accessible considering the load capacity **then**
5: π^{nr}.append(P)
6: **else**
7: π^{nr}.append(0)
8: goto step 4:
9: **end if**
10: **end for**
11: %Convert the NR solution to a feasible solution
12: Clear(π^{fea})
13: Clear(π^{cur})
14: **for** Every node P in π^{nr} **do**
15: **if** P is accessible considering the battery capacity **then**
16: π^{cur}.add(P)
17: $P_{furthest} \leftarrow P$
18: **else**
19: Find the cloest station $F_{nearest}$ to $P_{furthest}$
20: **if** $P_{furthest}$ can reach $F_{nearest}$ **then**
21: π^{cur}.add($F_{nearest}$)
22: **else**
23: $P_{furthest} \leftarrow$ previous of $P_{furthest}$
24: goto step 20
25: **end if**
26: **end if**
27: **end for**
$\pi^{fea} \leftarrow \pi^{cur}$

Output: π^{fea}

First, multiple clusters are established centered on each charging station (Step 1 to 4). Then customers are allocated to different clusters according to their distance from each cluster center (charging station) (Step 5 to 8). The third step involves connecting all customers based on the distance from each cluster center to the Depo. The DBCA algorithm arranges vehicles start from Depo, preferentially accesses the customers of the nearest cluster. These customers in the same cluster are accessed based on their distance to the Depo. When all cluster customers are connected, the EV will travel to the next cluster and perform a similar operation (Step 9 to 15). The steps above generate an unfeasible solution, which is the L-TSP solution. The DBCA then uses the logic of Insertion Heuristics to convert it into a feasible solution for the EVRP (Step 16). The DBCA makes some changes to the principles of Insertion Heuristics: It eliminates the need to iterate through all charg-

Algorithm 2 Density Based Cluster Algorithm

for Each charging station $F \in$ Charging Stations **do**
  Set F as Cluster Center
  Build Clusters F
end for
for Each customer $P \in$ Customers **do**
  Calculate distance between P and each Cluster F
  Distribute the P to the cloest Charging Stations F
end for
for Each cluster $C \in$ Clusters **do**
  for Each Customers $P \in$ cluster C **do**
    Calaculate distance from P to the Depo
  end for
  $\pi_{mid} \leftarrow$ Connect cluster C based on the distance
  π_0.append(π_{mid})
end for
Improved_Insertion_Heuristics(π^0)

Output: π^0

ing stations to identify the closest one to the current customer, saving much time. The procedures of the DBCA can be seen in Fig. 5.

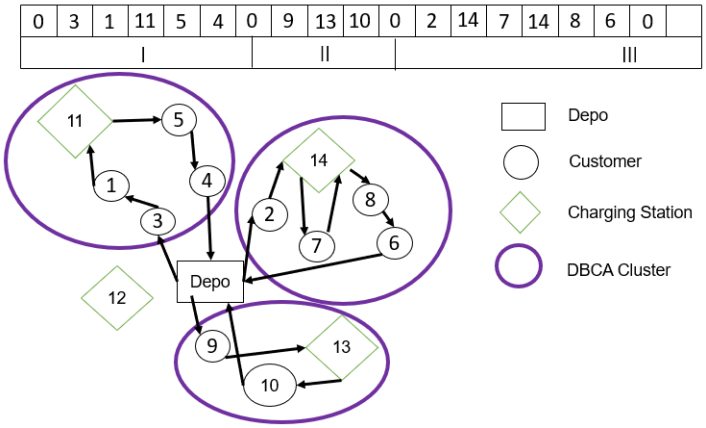


Fig. 5. The DBCA Initial Solution Generation

Purple ovals indicate that the DBCA includes customers and charging stations within different clusters. It is worth noting that not all charging stations will form a cluster, such as charging station 12 in Fig. 5. The algorithm concatenates all customers into an L-TSP solution, which is $0 \rightarrow 3 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 9 \rightarrow 10 \rightarrow 2 \rightarrow 7 \rightarrow 8 \rightarrow 6 \rightarrow 0$. After that, the DBCA performs an improved version of Insertion Heuristics to insert the Depo and charging stations into the solution. As a result of the improvement of the Insertion Heuristics, customers can only choose to charging stations within the same cluster but cannot select charging stations within other clusters.

4.3 Local Search

Whether it is VNS, ACO, or SA algorithm, it is inevitable to get trapped into local optimum in the process of searching. In this case, the Local Search algorithms are proposed to expand the search space and improve the quality of the solutions. It is a general term for a class of operators that aims to reorganize the original routing orders by perturbing, scrambling, and generating a new arrangement. This project uses six basic Local Search operators to improve the solution's performance: 2-opt, D-2-opt, T-2-opt, 3-opt, AFS-Realoc-one, and AFC-Realoc-Multiple. The Local Search operators 2-opt, D-2-opt, T-2-opt, and 3-opt cannot be directly implemented on a EVRP solution. It is necessary to obtain an L-TSP form of the EVRP solution, perform a local search, and then use Insertion Heuristics to construct a new EVRP solution. However, the AFS-Realoc-one and AFC-Realoc-Multiple are rearrangements of the charging stations in the EVRP solution. They do not affect the order of the permutations of customers so that they can be operated directly on the EVRP solution. The followings are detailed descriptions of each operator used in conjunction with the diagrams.

4.3.1 The 2-opt Perturbation

The 2-opt algorithm is one of the simplest Local Search operators. A new solution is generated by arbitrarily exchanging two positions in an L-TSP solution (neither of these two positions may be the start or end position of the solution). As shown in Fig. 6, the locations of customer 5 and customer 7 are swapped.

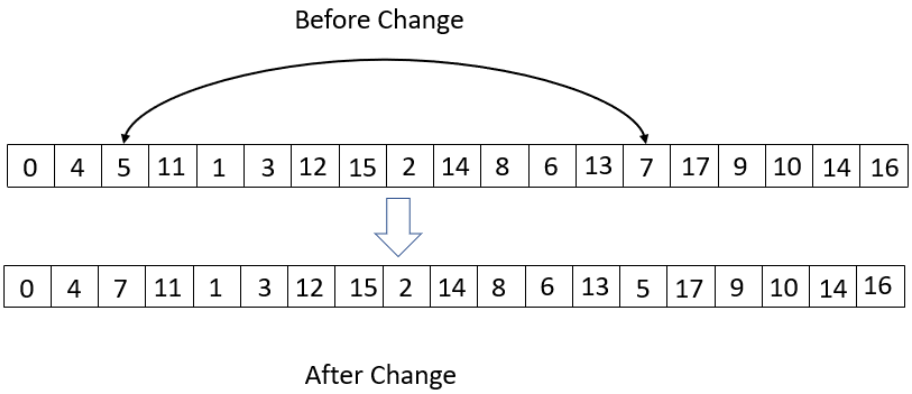


Fig. 6. The 2-opt Local Search Operator

4.3.2 The D-2-opt Perturbation

The D-2-opt extends the function of the 2-opt operator and makes some changes. It changes the customers' order in the solution by arbitrarily identifying four positions in a solution and swapping them in pairs. Additionally, these four positions cannot be the first and last positions of the L-TSP solution.

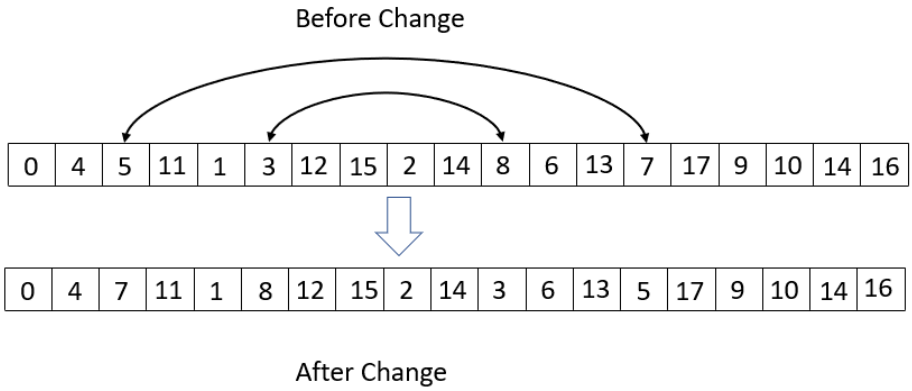


Fig. 7. The D-2-opt Local Search Operator

4.3.3 The T-2-opt Perturbation

The T-2-opt follows the same principles as the D-2-opt and 2-opt, except that it must randomly determine six positions and exchange them in pairs. Fig. 8 shows that the positions for customer 3 and 8, 5 and 7 and 17 and 16 are swapped.

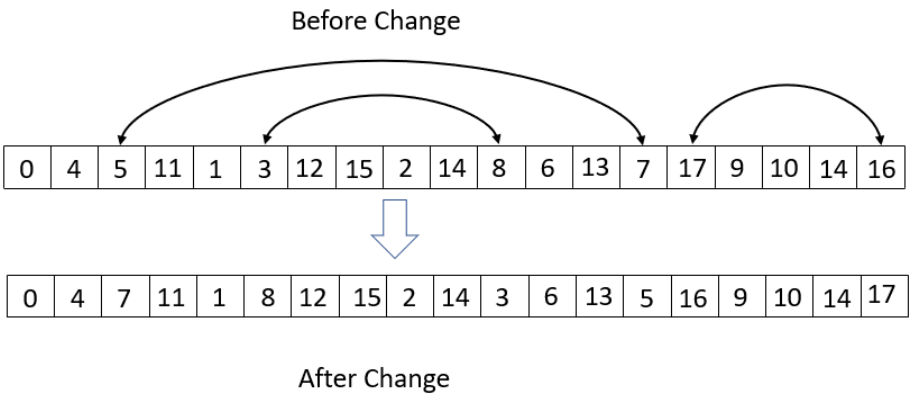


Fig. 8. The T-2-opt Local Search Operator

4.3.4 The 3-opt Perturbation

The 3-opt operator is more like an enhanced version of the D-2-opt operator. Three positions are randomly determined in the L-TSP solution, and their positions are exchanged at random. It changes three elements simultaneously, adding more randomness compared to D-2-opt. The locations of Customer 5, 15, 9 are randomly swapped as shown in the Fig. 9.

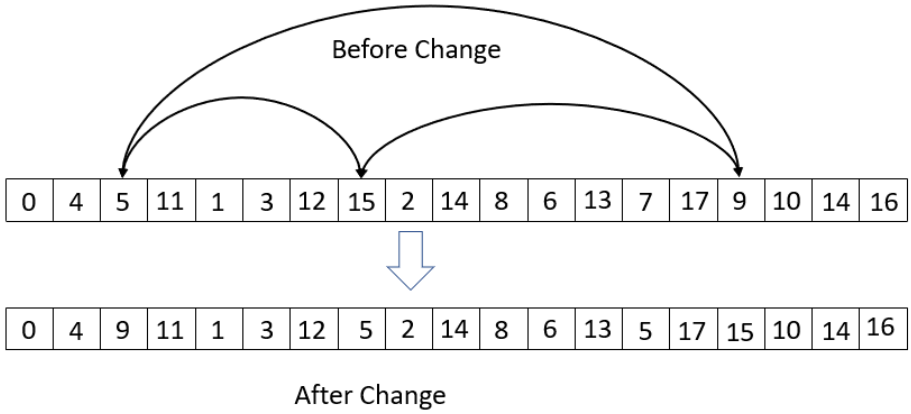


Fig. 9. The T-3-opt Local Search Operator

4.3.5 The AFS-Realoc-One Perturbation

The AFS-Realoc-One operates on a complete EVRP solution. It randomly replaces one charging station in the solution with another charging station, as shown in the Fig. 10.

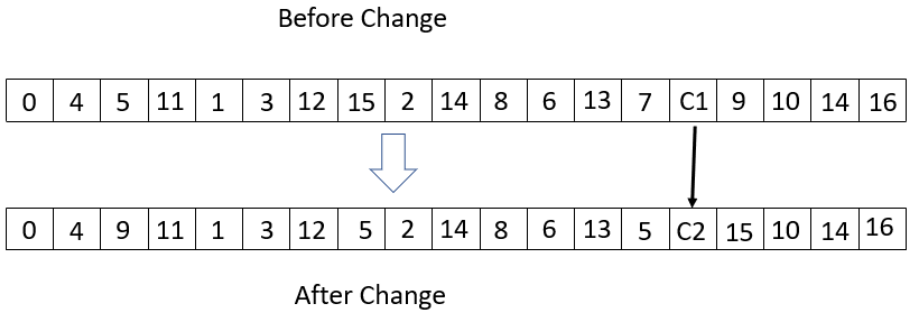


Fig. 10. The AFS-Realoc-One Local Search Operator

4.3.6 The AFC-Realoc-Multiple Perturbation

AFS-Realoc-Multiple also operates on a complete EVRP solution. In contrast to the AFS-Realoc-One, AFS-Realoc-Multiple will randomly select multiple charging stations and swap their positions to

generate new solutions. The AFC-Real-One affects the sub-route containing the specific charging station (a sub-segment of the solution, starting at 0 and ending at 0). At the same time, the AFS-Realoc-Multiple expands the influence to several sub-routes, so the change to the solution is more significant. The following figure shows the effect of the AFC-Realoc-Multiple operator.

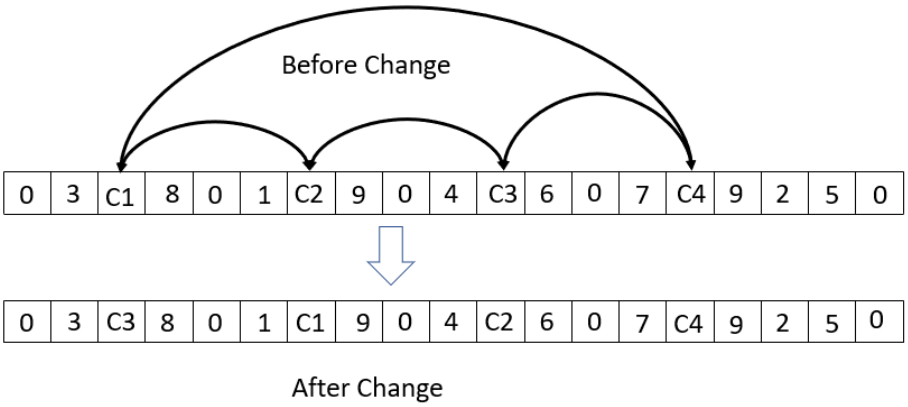


Fig. 11. The AFC-Realoc-Multiple Local Search Operator

Even though the Local Search can produce many solutions, not all are feasible solutions to the EVRP. Therefore, after the Local Search, it is necessary to check the validation of the solution additionally. The project has also designed a strategic way to increase the randomness of operator selections and search for more possible solutions: In the Local Search process, the previously mentioned six operators are labeled with a number. During the optimisation, each iteration will generate a random number between 1 and 6, representing the operator selection. The Local Search algorithm will then execute the operator corresponding to the number generated by the current iteration.

4.4 Implementation of Ant Colony Optimisation

Ants regulate their group behaviours through a tight positive feedback system [39]. In other words, the behaviour of an ant will affect the entire colony's action, and the colony's activity will affect the behaviour of each individual. Activities of the ant colony are controlled by the pheromone, a chemical released by ants on the roads they have passed [46]. Ants perform two tasks when they are traveling: The first task is to identify the road with high pheromone concentration and select it as the direction of travel. Then it releases pheromones on the route it passed to increase its concen-

tration, providing references for subsequent ants. Such a positive feedback mechanism allows the ant colony to find a suitable path to move forward naturally.

ACO is an efficient path search algorithm proposed by Dorigo *et al.* [47] in 1996, inspired by the phenomena of the ant colony. This algorithm derives an independent system for generating and updating solutions based on how ants find paths and release pheromones that guide the colony. Due to its inherent heuristic strategies, the ant colony algorithm can effectively solve complex path-routing problems with multiple constraints, such as multi-depot routing for green vehicles [43]. This project combines the ACO algorithm with the Insertion Heuristics (IH), the DBCA and various Local Search operators to strategically solve the EVRP. The procedure for the ACO algorithm is shown in Algorithm 3, which used parts of the algorithm model of Mavrovouniotis *et al.* [48].

Algorithm 3 Ant Colony Optimisation

$t \leftarrow 0$
$\tau_0 \leftarrow$Initialise_Pheromone
while termination state is not satisfied **do**
  for Every ant π in Ant Colony A **do**
    $\pi \leftarrow$ DBCA_solution_generation
    $\pi^{ltsp} \leftarrow$ get L-TSP solution(π)
    Construct_solutions(π^{ltsp})
    Local_Search(π^{ltsp})
    $\pi \leftarrow$Insertion_Heuristics(π^{ltsp})
    if Validation_Check(π) not feasible **then**
      goto step 8
    end if
  end for
  $\pi^{ib} \leftarrow$ Find_Best_Iterations(A)
  if $(\pi^{ib} < \pi^{bs})$ **then**
    $\pi^{bs} \leftarrow \pi^{ib}$
    if iterative threshold is detected **then**
      Migration(π^{ib})
    end if
  end if
  Update_Pheromone
  $t \leftarrow t + 1$
end while
Output: π^{bs}

4.4.1 Initial Colony Construction

The ACO algorithm abstracts each EVRP solution as an individual ant and a set of solutions as an ant colony. Changes to a solution may affect the entire solution set. An L-TSP solution is generated when constructing an ACO solution, and then the IH algorithm is used to construct a complete solution. In constructing the L-TSP solution, an ant will start from the Depo, traverse all customers, and finally return to the Depo. When reaching a customer, an ant may have multiple options for its next target. The ACO algorithm converts the concentration of pheromones into a probability density distribution, which helps ants to decide which direction to choose. Ants will always choose the path with the highest probability. The formula for the road probability distribution is as follows:

$$P_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{a}(h_{ij})^{\beta}}{\sum_{z \in J_k(i)} (\tau_{iz})^{a}(h_{iz})^{\beta}}, & \text{if } j \in J_k(i), \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

P_{ij}^{k} represents the probability that the ant k chooses to move from customer i to customer j. $J_k(i)$ stands for the set of currently accessible customers which can be chosen by the ant k. It is worth noting that if a customer was visited by the ant k, it should be removed from $J_k(i)$. τ_{ij} and h_{ij} represent the current pheromone concentration and heuristics information factor between the customer i and j respectively. The heuristics information factor h_{ij} can be calculated based on the distance of arc L_{ij}, which is shown in Eq. (14).

$$h_{ij} = \frac{1}{d_{ij}} \tag{14}$$

The variables α and β determine the relative influence of pheromone density τ_{ij} and heutistics information factor h_{ij} on the probability respectively. These two values directly impact the probability of pheromones on each road, impacting the choice of roads for the ants.

In Steps 5 and 6, the ACO uses the DBCA to generate an initial solution for the EVRP in the beginning, and its L-TSP version solution is extracted. Based on the probability distribution shown in Eq. (13), the ACO algorithm establishes a new L-TSP solution in Step 7. Steps 8 and 9 perform Local Search on the L-TSP solution and use Insertion Heuristics to convert it to a complete EVRP solution.

Since Local Search may not produce feasible solutions, Steps 8 and 9 must be repeated until a valid solution is created (Steps 10 to 12).

4.4.2 Updating Pheromone

pdating the pheromone is an essential part of the ACO algorithm, which is shown in Step 19 of the pseudo-code Algorithm 3. The changes in pheromone on each road can be classified into two categories:

- Pheromones will evaporate at a specific rate between customers or between customers and Depo. The significance of this step is to avoid some roads being too dominant and affecting the choice of other roads by the ants.
- The pheromone of all roads will be updated after the ants construct an L-TSP road and feasibly perform a Local Search.

ACO uses the following formula to dynamically update the pheromone [49]:

$$\tau_{ij}^{new} = (1-\varphi)\tau_{ij}^{old} + \Delta\tau_{ij}^{ib} \quad i,j \in (V \cup \{0\} \cup \{N+1\}) \tag{15}$$

where τ_{ij}^{new} represents the pheromone between vertex i and j after performing pheromone update operations. τ_{ij}^{old} represents the original pheromone concentration between vertex i and j. φ controls the rate of pheromone evaperation between vertex i and j.

$$\Delta\tau_{ij}^{ib} = \begin{cases} \dfrac{Q}{\text{Cost}^{ib}}, & \text{if the vertex } i \text{ and } j \text{ are on the L-TSP route} \\ 0, & \text{otherwise,} \end{cases} \tag{16}$$

where Q is a constant, and Cost^{ib} represents the cost for the best-so-far solution in the current iteration.

4.4.3 Migration

The migration determines how the ACO algorithm transfer an ant's solution to the ant colony. Mavrovouniotists *et al.* [48] proposed to consider two key aspects when performing migration. i) When to propagate an ant's solution to the entire population. and ii) How migration should happen. WConcerning the first aspect, an iterative threshold was designed in this ACO algorithm. An ant's solution is propagated to the ant population when the number of iterations reaches this threshold. For the second aspect, a gradient propagation method is designed to pass the solution to the colony. If the performance of the current best-so-far solution is worse than the overall best-so-far solution, then randomly replace the solution of one-fifth of the population with this solution. If the performance of the current best-so-far is better than the overall best-so-far solution, then Randomly pick half of the ants from the population and replace their solution with the current best solution. Steps 14 to 20 in the pseudo-code Algorithm 3 shows the process of ACO obtaining the current best-so-far solution in the iterations and perform the migration. It first determines whether the current optimal solution is the global best-so-far solution. After that, It performs a migration operation based on a certain conditional iteration threshold.

4.5 Implementation of Simulated Annealing

The Simulated Annealing (SA) is a trajectory-based random algorithm that escapes from the local optima and obtains the global optimum [37]. According to Yang [50], metals have the lowest energy, the largest crystal size, and fewer structural defects when rapidly cooled to a crystalline state. By adjusting the cooling rate (aka Annealing schedule), Simulated Annealing (SA) simulates the metal cooling process to a crystalline state. This project combines the annealing algorithm with some local search operators to obtain better solutions.

4.5.1 Simulated Annealing and Boltzmann Function

Traditional gradient-based algorithms and deterministic algorithms tend to fall into local optima. This result is because their small search span limits the search area. In other words, these algo-

rithms are prone to stagnating in the search after finding a local optimal solution. This kind of problem, however, has been effectively addressed by the introduction of the SA algorithm. Van Laarhoven and Aarts [51] have proved that the Simulated Annealing Algorithm can converge to global optimality when combined with sufficient randomness and a slow cooling rate.

Markov Chains and Random Search are the core concept of the SA algorithm. The iterative process accepts solutions that improve the objective function and solutions that are not satisfactory from time to time. The SA determines when not ideal solutions should be accepted using the Boltzmann function, whose main idea is shown in the Fig. 12.

The process of generating a new solution from an old solution using an algorithm such as local search can be described as searching in the neighborhood of the old solution. Obtaining the optimal solution can be compared to climbing a mountain, and the process of reaching the mountain's peak can be compared to climbing a higher peak. Each curve represents a search for a solution. The path $AC \rightarrow CD \rightarrow DE$ in Fig. 12 represents the traditional search algorithms. As a result of its small search range, it is easy to fall into a local optimum with A as the optimal solution. This means that despite many searching steps, it cannot find a better value than A. AB stands for the idea of using the Boltzmann function, increasing the search span according to the strategy, and accepting a not ideal solution. As shown in Fig. 12, although the result B produced by searching AB may not be as good as result A, the subsequent search BH will yield a better solution than A.

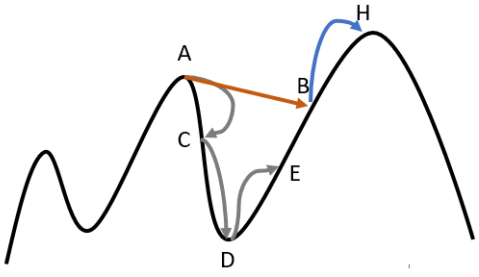


Fig. 12. Hill Climbing Model

The parameter settings of the Boltzmann function are shown below. Through the definition of section 1.1, the objective function is defined as $\phi(\pi)$, and π is a feasible EVRP solution. Suppose that the current optimal solution is π_{old}. Through the local search on π_{old}, a new feasible solution π_{new} is generated. The SA will accept the solution π_{new} if it performs better than the solution π_{old}. Otherwise, the Boltzmann function will randomly take a value in the Gaussian distribution interval $[0, 1]$

as the receptive threshold, represented by the symbol ε. Then, it will use the following formula to determine whether the solution is accepted or not:

$$p = \exp\left[-\frac{\Delta\phi}{T}\right] > \varepsilon \tag{17}$$

where T is the current temperature when running the Simulated Annealing algorithm. $\Delta\phi$ represents the change in energy level, which can be calculated by:

$$\Delta\phi = \phi(\pi_{new}) - \phi(\pi_{old}) \tag{18}$$

The probability p, also called *transition probability*, represents the probability that the non-ideal solution is accepted. The solution can only be accepted if its value is greater than the random value ε. In addition, several conclusions can be derived from Eq. (17):

- During the early stages of the annealing algorithm, the value of T is high, and the non-ideal solutions are more likely to be accepted; as the iteration progresses, the value of T becomes lower, and the probability of the solutions being accepted decreases.
- If the performance of π_{new} is better than that of π_{old}, the value of $\Delta\phi$ is negative. So propability p is greater than 1. Considering that r is drawn from a uniform distribution in $[0, 1]$, the solution must be accepted by the algorithm.

4.5.2 Simulated Annealing Optimisation

The implementation process of the SA algorithm is relatively simple, which is shown in Algorithm 4. Firstly, the initial solution of the algorithm, the starting and ending temperature, and the max iteration times are Initialised (Steps 1 to 5). In step 6, SA regulates the rate at which the temperature drops. Processes 7 to 15 represent a normal local search process in SA. The temperature drop threshold and the maximum number of iterations limit the depth of the final execution of SA. Steps 16 to 21 show that if the current best-so-far solution works better than the global-so-far solution,

Algorithm 4 Simulated Annealing

1: $t \leftarrow 0$
2: $\pi^0 \leftarrow$ DBCA_solution_generation
3: $T_0 \leftarrow$ Set_start_temperature
4: $T_f \leftarrow$Set_final_temperature
5: $N \leftarrow$ Set_max_iterations
6: Define_cooling_schedule $T \leftarrow \alpha T$, $(0 < \alpha < 1)$;
7: **while** $T > T_f$ and $t < N$ **do**
8: $\varepsilon \leftarrow$ Get_from_Gaussian_Distribution
9: $\pi^{ltsp} \leftarrow$ get L-TSP solution(π)
10: Construct_solutions(π^{ltsp})
11: Local_Search(π^{ltsp})
12: $\pi \leftarrow$Insertion_Heuristics(π^{ltsp})
13: **if** Validation_Check(π) not feasible **then**
14: goto step 11
15: **end if**
16: $\pi^{ib} \leftarrow$ Find_Best_Itertions
17: **if** $(\pi^{ib} < \pi^{bs})$ **then**
18: $\pi^{bs} \leftarrow \pi^{ib}$
19: $\pi \leftarrow \pi^{ib}$
20: **end if**
21: calaculate probability p
22: **if** not improved and $p > \varepsilon$ **then**
23: $\pi^{bs} \leftarrow \pi^{ib}$
24: $\pi \leftarrow \pi^{ib}$
25: **end if**
26: $t \leftarrow t + 1$
27: **end while**
Output: π^{bs}

record it as the global solution. The Boltzmann function is used to aperiodically accept not ideal solutions to widen the search, as shown in Steps 22 to 25.

Table 2. Details of the EVRP Instances

Instance	Customer	Charging Stations	Load Capacity	Energy Capacity	Energy Consumption
E-n22-k4	22	8	6000	94	1.2
E-n23-k3	23	9	4500	190	1.2
E-n30-k3	30	6	4500	178	1.2
E-n33-k4	33	6	8000	209	1.2
E-n51-k5	51	9	160	105	1.2
E-n76-k7	76	9	220	98	1.2
E-n101-k8	101	9	200	103	1.2
X-n143-k7	143	4	1190	2243	1
X-n214-k11	214	9	944	987	1
X-n351-k40	351	35	436	649	1
X-n459-k26	459	20	1106	929	1
X-n573-k30	573	6	210	1691	1
X-n685-k75	685	25	408	911	1
X-n749-k98	749	30	396	790	1
X-n819-k171	819	25	358	926	1
X-n916-k207	916	9	33	1591	1
X-n1001-k43	1001	9	131	1684	1

5 SIMULATION STUDIES

The previous sections primarily discussed the process and principles of implementing two heuristic algorithms: ACO and SA, and how they were combined with the DBCA and IH algorithms extended from the VNS code. This section uses EVRP instances to test the performance of these two algorithms. Additionally, since the code for VNS won the first prize in the CEC-12 EVRP Competition, the results of the VNS algorithm were used as the benchmark to measure the gap between the implemented algorithms and the winning algorithm. C++ language was used to implement the ACO, SA, and VNS algorithms in VScode Code Editor. All the programs were executed on the Apple Macbook-14, with Apple M1 Pro Chip and 16GB RAM.

5.1 Data Set Illustration

Mavrovouniotis provided the seventeen benchmark instances used in this project on his homepage. They were designed for the CEC-12 EVRP Competition and specifically tested the performance of algorithms on the EVRP. Three algorithms were selected as winners in the CEC-12 EVRP Competition, and their final performance on each instance can be viewed on Mavrovouniotis's

homepage. This project only used the results of VNS as benchmarks for evaluation and comparison. The details for seventeen instances are shown in Table 2.

These instances can be divided into two categories: E-type and X-type. The power consumption of EVs is 1.2/km for the E-type instances and 1/km for the X-type instances. The X-type instances are much more complicated than E-type, with more customers and charging stations but less load capacity for EVs. Moreover, since the distance between each element (the Depo, customers, and charging stations) of an X-type problem is relatively large, which makes it more difficult for an algorithm to plan routes than an E-type problem.

5.2 Algorithmic Parameter Settings

CEC-12 proposed a control method for controlling a program's maximum iteration depth and running times. The process of checking the feasibility of a solution, or solution replacement, is considered an evaluation. A program's maximum iteration depth should not exceed 25000 times. In addition, it stipulated that a program should be executed 20 times, of which the smallest result was the final result for the algorithm. This project complied with these CEC-12 standards for the program and used the generated results for evaluation and comparison.

Parameters tunning plays the most crucial role in determining the performance of algorithms on the instances. The implementation and parameter adjustment of VNS were completed before the CEC-12 Competition. Therefore, this project accurately tuned the parameters of algorithms ACO and SA.

Table 3. Details of the SA Parameters

Start Temperature: T_{start}	End Temperature: T_{end}	Alpha: α	Evaluations: $evals$
100	1	0.0004	25000

5.2.1 SA Parameter Settings

The final parameter settings of the SA are shown in Table 3. Settings of temperature T_{start} and T_{end} took into account Vincent *et al.*'s [37] parameter settings to solve the Hyprid EVRP. The setting of

Table 4. The Performance of SA Algorithm with Different α Values on Seventeen Instances.

Instances	$\alpha = 0.00004$		$\alpha = 0.0004$		$\alpha = 0.04$		$\alpha = 0.4$	
	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)
E-n22-k4	384.68	3.84	384.68	0.99	384.68	0.13	393.61	0.02
E-n23-k3	571.95	4.03	571.95	1.06	571.95	0.14	573.13	0.03
E-n30-k3	509.47	4.83	509.47	1.45	509.47	0.19	509.47	0.04
E-n33-k4	840.15	7.2	840.15	2.72	840.57	0.33	845.46	0.06
E-n51-k5	529.9	12.68	529.9	4.4	543.41	0.55	563.16	0.1
E-n76-k7	696.27	25.24	696.27	9.78	698.81	1.36	721.78	0.25
E-n101-k8	837.25	42.95	844.65	17.65	855.68	2.21	877.66	0.41
X-n143-k7	16371	94.9	16371	49.78	16419	7.2	16784	1.35
X-n214-k11	11510	225.41	11510	119.64	11695	19.36	11853	3.19
X-n351-k40	27401	980.66	27401	522.3	27491	84.52	28106	16.25
X-n459-k26	25481	1161.4	25481	632.46	25784	122.91	26961	27.62
X-n573-k30	52458	2197.5	52458	1291.6	52451	404.52	53393	74.44
X-n685-k75	71586	4343.9	71586	2425.6	72333	572.93	73008	153.16
X-n749-k98	81322	7931.9	81322	4468.5	81697	1123.1	82122	258.19
X-n819-k171	164932	16900	164932	9369.6	165491	2106.1	165995	620.33
X-n916-k207	342012	36745	342012	21299	342921	6282.4	344422	2048.7
X-n1001-k43	78027	6643.5	78027	3883.1	78091	1181.7	78669	197.43

the α parameter was determined by careful evaluation. α controls the rate of the temperature drop through the following formula:

$$T_{n+1} = \alpha T_n \tag{19}$$

If the temperature drops too rapidly, the SA algorithm will miss many possible excellent solutions, while the solution will do many useless searches if the temperature drops too slowly. In order to enhance the solution performance, it is therefore imperative to set the α value reasonably.

The value of α was systematically adjusted within the range $\{0.00004, 0.0004, 0.004, 0.4\}$ in 10-fold increments. The performance for each α value can be seen in Table 4. When the value of α is small, the SA search range is small, and more solutions with better performance may be found, while the large α leads to the miss of performance. It can be seen that performance of the model with $\alpha = 0.0004$ and $\alpha = 0.004$ is much better than that of the model with $\alpha = 0.04$ and $\alpha = 0.4$. However, the time the program takes to run is inversely proportional to the depth of the search. Detailed searches can also consume much time. Table 4 shows that the greater the value of α, the faster it requires to complete the instances.

Ultimately, the project chose 0.00004 as the best value for α. The value of α with values 0.4 and 0.04 are discarded because they lead to the poor performance of the SA algorithm. Compared with

the model with $\alpha = 0.00004$, the model with $\alpha = 0.0004$ performs the same on most instances, and its running time is less. However, it does not perform as well as the model with $\alpha = 0.0004$ on the E-n101-k8 instance. Since the EVRP problem has no constraints on the running time, the model takes the parameter $\alpha = 0.0004$ with the best performance as the final choice.

5.2.2 ACO Parameter Settings

The parameter settings for the ACO are shown in Table 5 In ACO, α and β determine the influence degree of pheromone concentration and heuristics information on the road probability distribution, respectively. q and ρ affect the process of pheromone updating. These four parameters were tuned from Mavrovouniotis's study *et al.* [48], which utilised the ACO to work out the EVRP-TW problem. Therefore, this project adopted these four parameters of Mavrovouniotis and meticulously tuned the number of ants participating in the ACO algorithm.

Table 5. Details of the ACO Parameters

Number of ants: N	Alpha: α	Beta: β	Q: q	Rho: ρ	Evaluations: $evals$
6	1	2	80	0.1	25000

The number of ants N was tuned from values $N \in \{1, 2, 6, 10, 15\}$. During parameter tuning, this experiment also examined the effect of increasing the number of ants on algorithm performance. These results can be seen in Table 6. The influence of changes in the number of ants on the performance of the ACO algorithm is not intuitive. The most optimal number of ants may be different for each instace. For example, on the intance E-n30-k3, the model with $N = 20$ has the best performance where in X-n916-k207, the model with $N = 1$ perofrms the best. However, after careful observation, it can be found that when the number of ants is small ($N < 10$), the effct of the model is better than that with a large ant number. Among these models, $N = 6$ is chosen as the best parameter because it performs best on the most instances, although it is slightly inferior to the ant model with a number of 1 on large-scale X-type data.

Table 6. The Performance of ACO Algorithm with Different N Values on Seventeen Instances.

Instances	$N=1$		$N=2$		$N=6$		$N=10$		$N=15$		$N=20$		$N=25$	
	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)	Result(km)	Time(s)
E-n22-k4	384.68	1.14	384.68	2.85	384.68	5.33	384.68	8.57	384.68	11.63	384.68	17.56	384.68	21.49
E-n23-k3	571.95	1.04	571.95	2.49	571.95	3.83	571.95	6.72	573.49	11.03	573.49	16.89	573.49	20.97
E-n30-k3	515.62	1.32	512.6	2.83	547.31	6.61	554.33	11.29	517.91	15.8	512.35	23.28	518.28	28.33
E-n33-k4	844.25	1.6	840.57	3.94	840.82	5.96	840.57	8.94	840.82	13.46	840.57	22.8	840.57	28.89
E-n51-k5	575.02	3.51	554.32	9.07	541.35	15.01	564.05	23.57	580.38	32.41	561.58	51.76	550.18	64.5
E-n76-k7	709.58	6.85	708.62	15.45	706.58	26.87	703.36	47.51	737.12	71.38	765.13	129.11	723.54	156.08
E-n101-k8	891.81	11.25	898.77	26.56	886.91	49.2	926.82	86.49	944.75	129.91	948.32	212.63	988.49	261.25
X-n143-k7	17099.04	19.24	17485.53	41.07	17248.9	72.7	17248.9	124.94	18543.11	212.33	18322.78	565.2	19068.89	651.16
X-n214-k11	14467.71	46.8	13886.1	132.9	16423.17	267.43	16482.78	463.75	16182.8	643.85	16359.19	1093.46	16676.06	1316.78
X-n351-k40	37313.86	180.4	38535.93	561.39	37313.86	930.34	43167.54	1545.69	43167.54	1716.55	43855.18	2969	43498.52	3597.5
X-n459-k26	37302.82	242.66	38553.52	815.57	37302.82	1668.76	38950.47	2643.14	40397.02	3528.14	36955.94	5626.98	41891.41	6688.24
X-n573-k30	60615.44	428.07	60029.31	1348.92	60481.07	2337.58	63890.73	3812.27	60797.01	5190.35	66978.27	8432.32	66978.27	10102.08
X-n685-k75	118145.5	889.07	120371.5	2551.32	118145.5	4590.34	124017.6	6565.47	124045.4	9065.56	126674.8	14747.09	126962.7	17546.44
X-n749-k98	115862.6	892.48	120133.1	2118.37	115862.6	3784.82	128225.5	5793.05	130637.1	6701.57	129284	11055.8	130637.1	13417.18
X-n819-k171	201547.5	1756	203001.2	4543.08	204626	7868.65	199881.6	10868.81	205386.8	14350.38	204759.6	21865.16	205386.8	25774.14
X-n916-k207	402473.1	2162.92	405723.9	5495.74	406904.8	9021.69	409771.2	12924.31	406393.5	16855.74	409756.4	25703.2	409756.4	30377.58
X-n1001-k43	115028.9	1386.81	115896.1	3905.43	115028.9	7319.34	129876.5	11721.38	122169.6	16592.86	132183.3	26220.98	130842.4	32097.37

Instances	VNS	SA	ACO
E-n22-k4	2.89	3.84	5.33
E-n23-k3	3.02	4.03	3.83
E-n30-k3	3.52	4.83	6.61
E-n33-k4	4.44	7.2	5.96
E-n51-k5	8.19	12.68	15.01
E-n76-k7	15.19	25.24	26.87
E-n101-k8	24.74	42.95	49.2
X-n143-k7	43.52	94.9	72.7
X-n214-k11	102.11	225.41	267.43
X-n351-k40	449.69	980.66	930.34
X-n459-k26	510.89	1161.35	1668.76
X-n573-k30	883.84	2197.49	2337.58
X-n685-k75	1897.31	4343.88	4590.34
X-n749-k98	3402.71	7931.85	3784.82
X-n819-k171	7323.42	16900.05	7868.65
X-n916-k207	14670.93	36744.68	9021.69
X-n1001-k43	2702.85	6643.54	7319.34

Table 7. The Running Time (unit: s) for Three Algorithms on Seventeen Instances.

5.3 Results and Analysis

The previous subsections described the process of parameter tuning of SA and ACO. This subsection will compare the performance of ACO, VNS, and ACO on the instances, combined with their running results.

5.3.1 Time Analysis

The running time of the three algorithms can be seen in Table 7. As the size of an instance increases, the time required for algorithms to solve the problem also increases significantly, which is a normal

situation. The gap between the running time of the three algorithms also becomes more significant with the rise of the instance's complexity. For example, the running time of SA on the E-n51-k4 is 1.5 times that of VNS, and when the scale increases to X-n819-k171, the running time gap widens to more than 2 times. Due to the increasing size of the data set, the number of customers and charging stations increases, and the algorithm must spend considerable time generating new solutions and determining their feasibility. It is the critical point of the dramatic increase in runtime.

The VNS algorithm performs best from a time perspective, despite its slightly longer runtime on X-n916-k207 than the ACO. In nearly half of the datasets, ACO's runtime advantage is slightly greater than SA's. Therefore, descending order of the running time advantage can be obtained: $VNS > ACO > SA$.

5.3.2 Result Analysis

The results of the three algorithms can be seen in Table 8. Each algorithm was run 20 times, taking the maximum, and minimum and calculating the mean and standard variance values. An algorithm's performance can be measured by its minimum and maximum values and average values. In the CEC-12 competition, from E-n22-k4 to X-749-k98 instances, the solutions obtained by VNS mostly represent the current best solution. Also, SA performed well, with a small gap between its results and VNSs, ranging from 1% to 5%. On E-n101-k8, the minimum result of SA is even better than that of VNS. However, from the average and maximum values, in the results of 20 runs, the SA algorithm is generally not as good as the VNS results. The performance of ACO is the worst. The gap between its results and VNS is more than 10% on nearly half of the small-scale instances. On X-type instances, such as X-n916-k207, the gap even expands to more than 30%.

The standard deviation reflects the difference between the results of each run. At the same time, it also reflects whether the algorithm can obtain results stably. VNS and SA appear to have similar standard deviations, indicating that the convergence situation is good and the final algorithm is likely to be stable. ACO has an extremely high standard deviation, nearly ten times higher than SA or VNS on E-type instances and nearly 30-40 times higher on X-type instances. It reflects that the randomness of ACO is too large, the results obtained by the algorithm are volatile, and it cannot achieve a convergence state. Therefore, descending order of the arrangement of the results can be

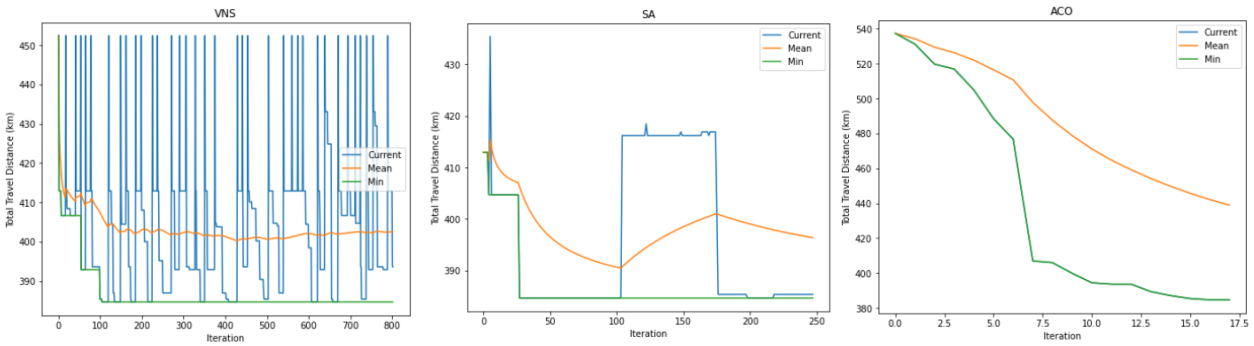
Table 8. The Final Results for Three Algorithms on Seventeen Instances.

Instances	VNS				SA				ACO			
	Min(km)	Max(km)	Mean(km)	Stdev	Min(km)	Max(km)	Mean(km)	Stdev(km)	Min(km)	Max(km)	Mean(km)	Stdev(km)
E-n22-k4	384.67	384.67	384.67	0.00	384.67	384.67	384.67	0.00	384.68	390.30	386.42	1.71
E-n23-k3	571.94	571.94	571.94	0.00	571.94	571.94	571.94	0.00	571.95	580.82	576.90	3.91
E-n30-k3	509.47	509.47	509.47	0.00	509.47	509.47	509.47	0.00	547.31	564.66	559.42	4.95
E-n33-k4	840.14	840.46	840.43	1.18	840.15	840.57	840.18	0.11	840.82	875.59	851.60	12.44
E-n51-k5	529.90	548.98	543.26	3.52	529.90	558.74	544.09	7.31	541.35	688.99	595.31	50.89
E-n76-k7	692.64	707.49	697.89	3.09	696.27	714.52	700.25	4.75	706.58	821.10	742.90	37.52
E-n101-k8	839.29	853.34	853.34	4.73	837.25	862.39	849.92	5.83	886.91	1090.31	969.28	57.08
X-n143-k7	16028.05	16883.38	16459.31	242.59	16371.24	16990.74	16651.51	219.37	17538.57	22190.91	19613.31	1498.49
X-n214-k11	11323.56	11660.70	11482.20	76.14	11510.00	11749.48	11627.80	63.94	16423.17	17091.40	16796.90	219.14
X-n351-k40	27064.88	27418.38	27217.77	86.20	27400.98	27793.42	27571.87	115.48	38420.86	44554.86	42127.79	1958.34
X-n459-k26	25370.80	25774.62	25582.27	106.89	25480.60	26277.43	25790.09	181.12	38479.31	43842.92	42049.33	1818.04
X-n573-k30	52181.51	51929.24	52548.09	278.85	52457.57	53210.88	52720.25	200.48	60481.07	68491.82	65028.93	3076.57
X-n685-k75	71345.40	72187.75	71770.57	197.08	71586.42	72452.24	72135.05	222.16	119893.35	129138.70	126226.11	3342.77
X-n749-k98	81002.01	81634.06	81327.39	176.19	81322.39	82202.47	81858.28	223.22	119453.66	134336.90	129513.49	5870.20
X-n819-k171	164289.95	165571.48	164926.41	318.62	164931.77	166101.04	165340.43	303.33	204625.95	211219.37	208468.36	2227.46
X-n916-k207	341649.91	343338.01	342460.70	510.66	342011.67	343828.88	342835.39	500.90	406904.79	414484.00	410823.78	2625.89
X-n1001-k43	77476.36	78464.68	77920.52	234.73	78027.08	78878.80	78442.38	241.36	117095.34	135248.44	127971.96	6971.24

obtained: $VNS > SA > ACO$.

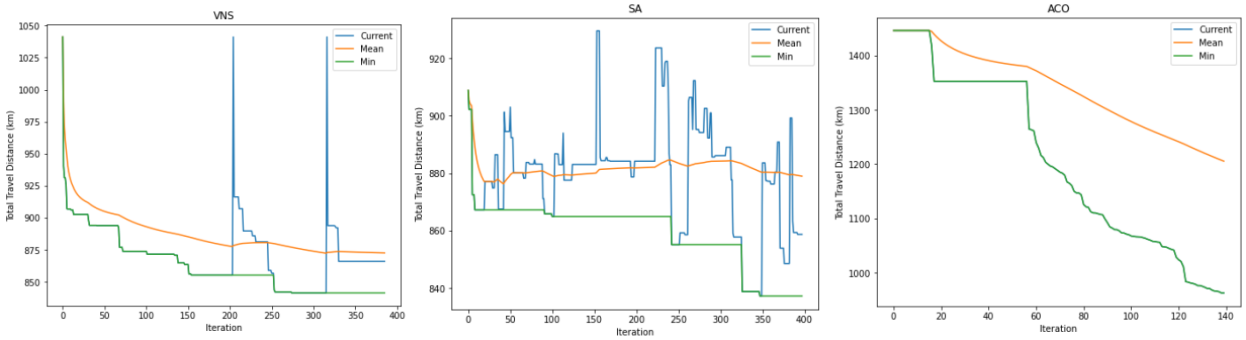
5.3.3 Running Process Analysis

This part focuses on the intermediate process of the algorithm runtime. During the optimisation, the performance of the global best solution, current solution's performance, and mean performance were recorded. These information was plotted into distance-iteration graphs, which are used to explore the changes during optimisation. Instances can be divided into three categories based on their optimising patterns, which are represented by the instances "E-n22-k4", "E-n101-k8", and "X-n1001-k43" respectively. The optimising legends for these three instances are analysed in this section, while images of the other instances will be placed in the Appendix. E-n22-k4 represents a small-scale instance, E-n101-k48 represents a medium-scale instance, and X-n001-k43 represents a large-scale one.



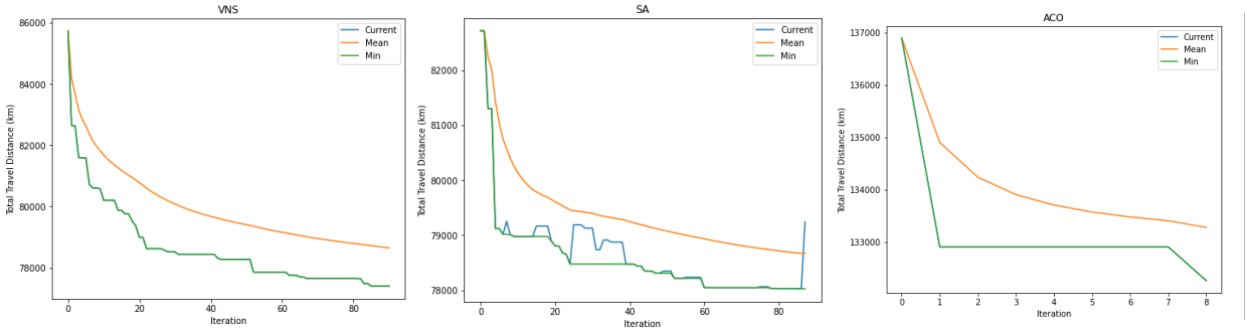
VNS SA ACO

Fig. 13. Visual Comparison of Three Algorithms on Instance E-n22-k4



VNS SA ACO

Fig. 14. Visual Comparison of Three Algorithms on Instance E-n101-k48



VNS SA ACO

Fig. 15. Visual Comparison of Three Algorithms on Instance X-n1001-k43

The blue line in the graph represents the performance of the current solution, the green line represents the performance of the best solution, and the orange line represents the average performance. The increase in scale affects the search depth of VNS. In Fig. 13, VNS can perform multiple neighourhood searches (the blue line has multiple fluctuations), but in Fig. 14, it only perform three neighourhood searches, with two fluctuations in the blue line. At the same time, the number of iterations of VNS also decreases with the increase of data size. However, these patterns do not apply to the SA and ACO. Changes in their searches may be related to the location of charging stations and customers.

The trend of the average reflects the convergence degree of the algorithm. The average curves of the algorithms of VNS and SA in Fig. 14 and Fig. 15 tend to be stable after decreasing, indicating that they have successfully reached the state of convergence. The average value curve of the ACO algorithm keeps decreasing, indicating that it has not converged in the finite iteration at all. On the large-scale dataset X-1001-k43, none of the three algorithms converge. One possible reason is that the complexity of the problem is relatively high, and it is difficult to obtain better performance in the neighbourhood of the excellent solution.

6 CONCLUSION

This project implements and compares the three heuristic algorithms for solving the electric vehicle routing problem (EVRP). This problem is considered a variant of the vehicle routing problem (VRP), which considers the EV load capacity and battery capacity restrictions. However, it ignores time-related information, such as charging and unloading time. The objective of the EVRP is to design the shortest possible driving route for EVs, given the location of charging stations and customers, to satisfy all customer requirements while meeting load and battery capacity constraints. Three heuristic algorithms, VNS, ACO, and SA, have been used in this project to solve this problem. The IH and DBCA algorithms, learned from the VNS, were used in the implementation process of SA and the ACO to improve their performance.

The experimental results show that both SA and VNS can be used as perfect algorithms for solving the EVRP. The VNS is slightly better than the SA regarding running time and final results. The ACO algorithm performs exceptionally poorly. Although its running time is shorter than SA, its final performance is much worse than SA. Moreover, the ACO algorithm does not converge during the running process, indicating that it may fall into local optimal or that the ability to find an excellent solution is poor. The reason may be that there is a problem with parameter adjustment, the migration strategy is not smart enough, or the entire search strategy is too simple.

In future work, the parameters of SA and ACO will be adjusted more strategically to see if they can continue to improve the performance when solving the EVRP. Moreover, the ACO will be combined with more excellent algorithms and try more novel migration methods to improve its performance on the EVRP instances. Finally, I expect to find inspiration from more algorithms to explore more efficient ways to solve EVRP problems. Some other heuristic algorithms, such as the Genetic Algorithm, will also be introduced and explored.

7 APPENDIX

Fig. 16. Visual Comparison of Three Algorithms on Instance E-n22-k4

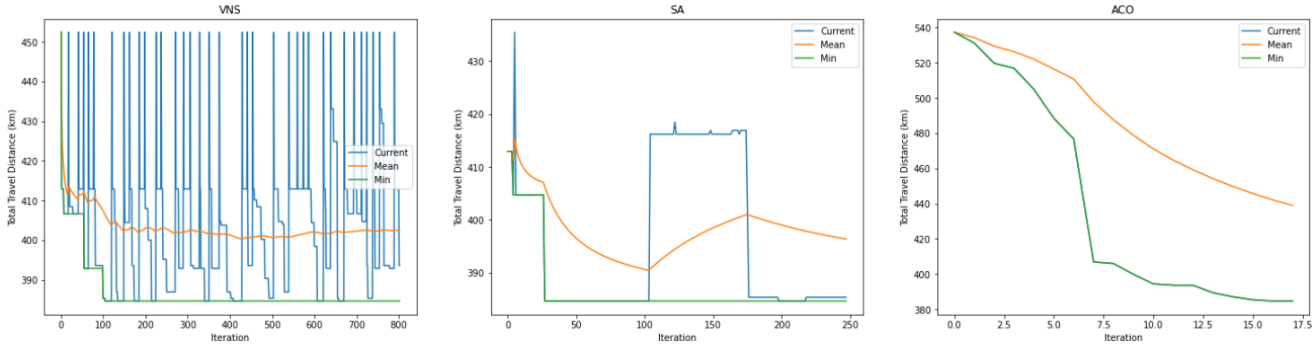


Fig. 17. Visual Comparison of Three Algorithms on Instance E-n23-k3

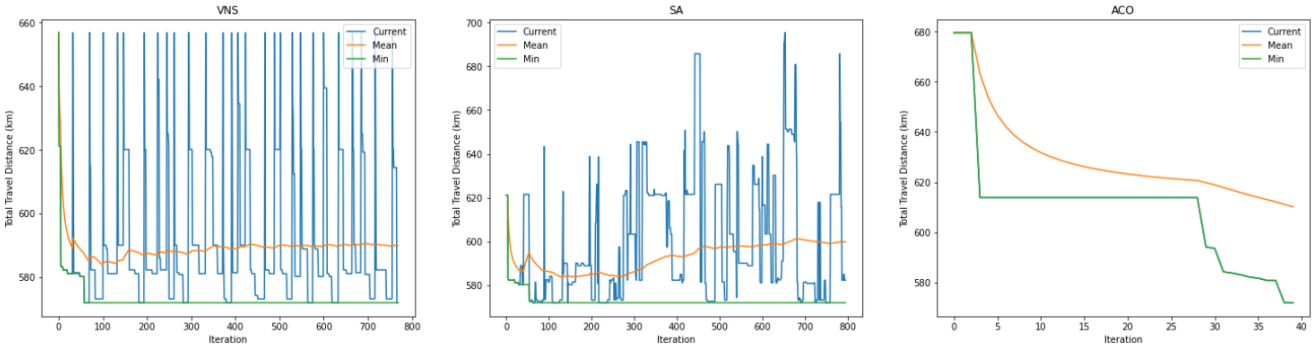


Fig. 18. Visual Comparison of Three Algorithms on Instance E-n30-k3

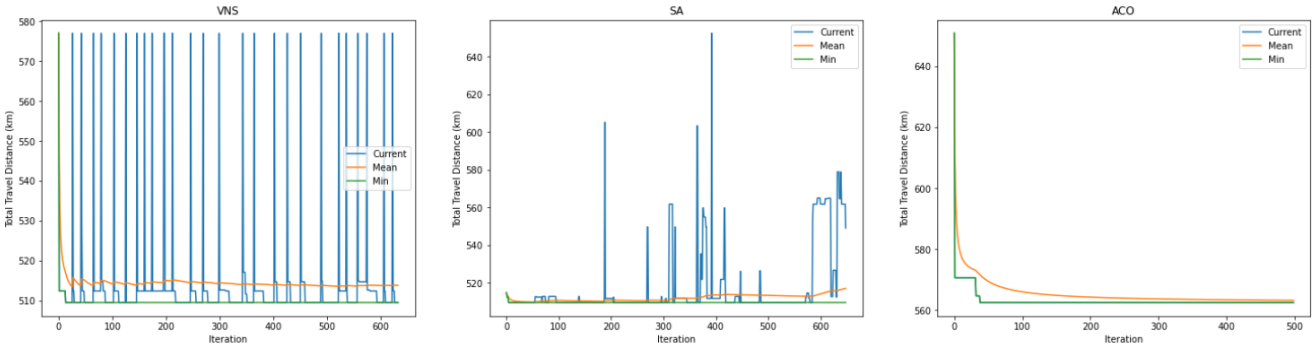


Fig. 19. Visual Comparison of Three Algorithms on Instance E-n33-k4

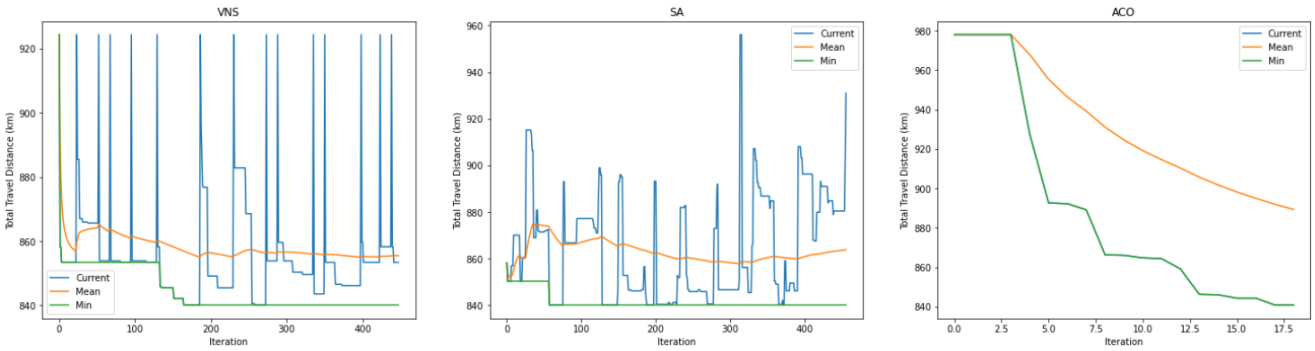


Fig. 20. Visual Comparison of Three Algorithms on Instance E-n51-k5

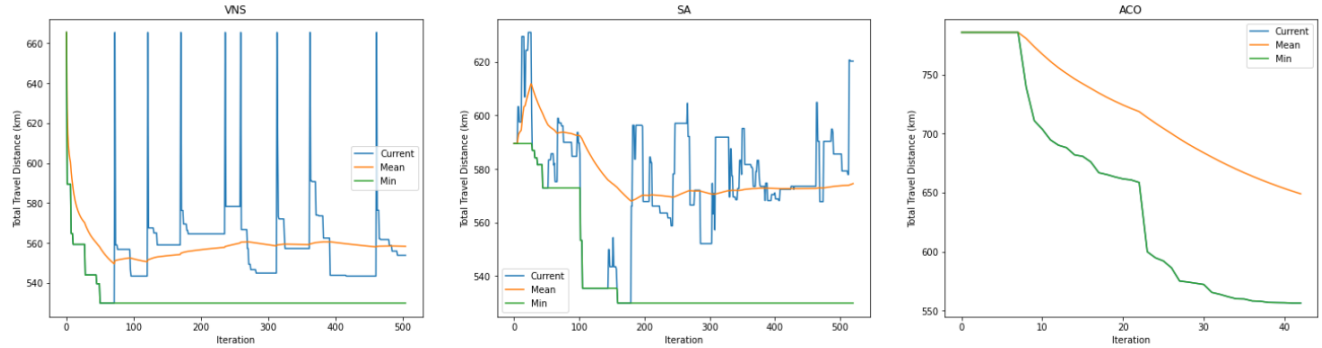


Fig. 21. Visual Comparison of Three Algorithms on Instance E-n101-k8

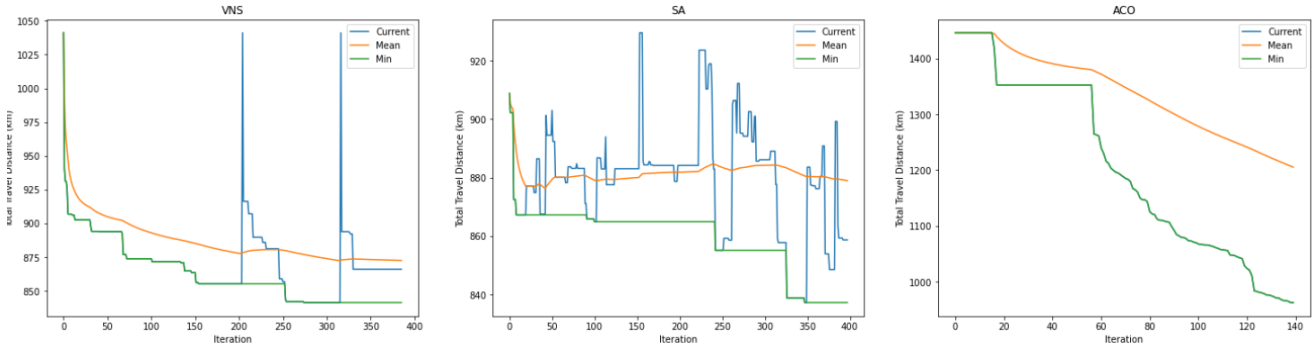


Fig. 22. Visual Comparison of Three Algorithms on Instance X-n143-k7

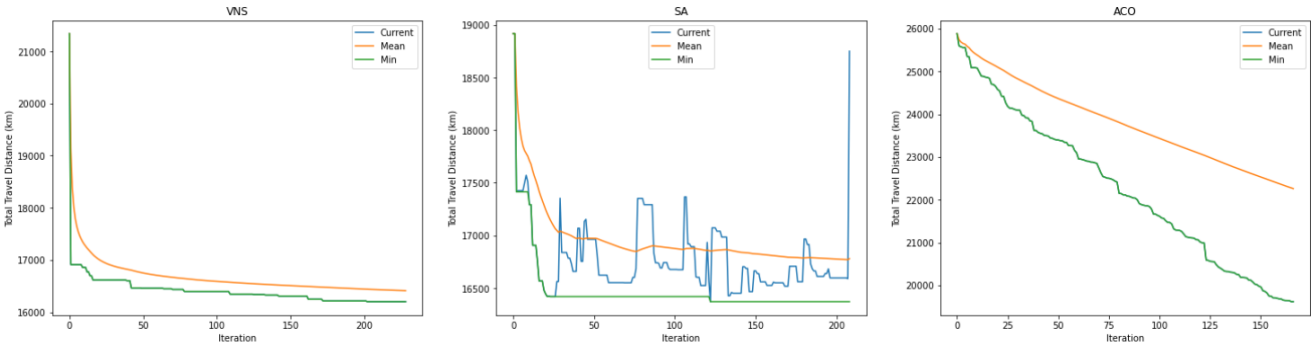


Fig. 23. Visual Comparison of Three Algorithms on Instance X-n214-k11

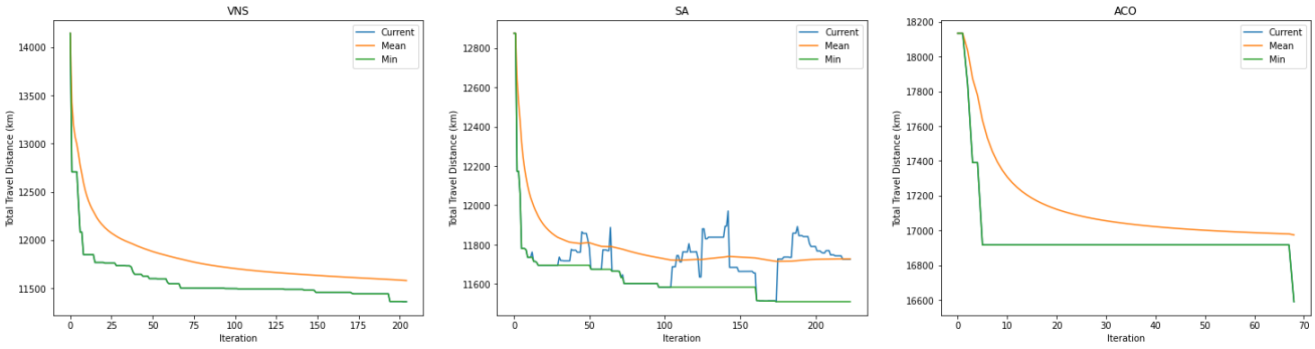


Fig. 24. Visual Comparison of Three Algorithms on Instance X-n459-k26

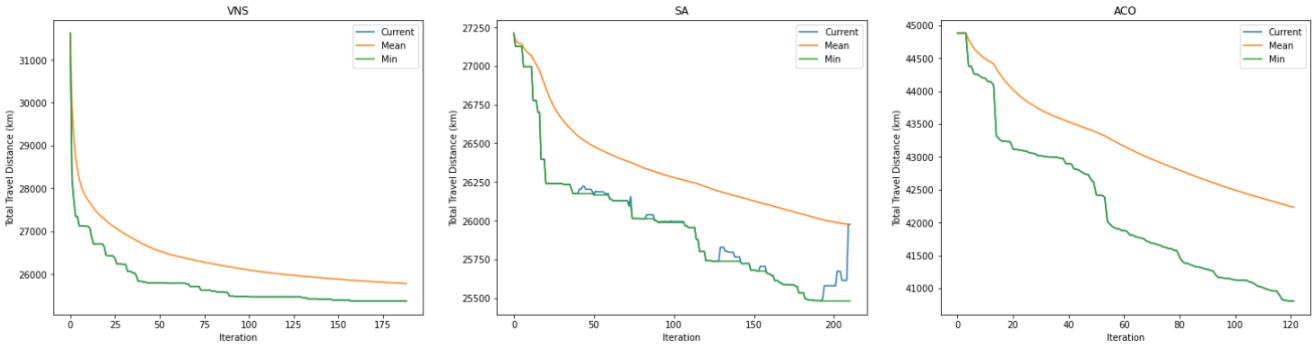


Fig. 25. Visual Comparison of Three Algorithms on Instance X-n573-k30

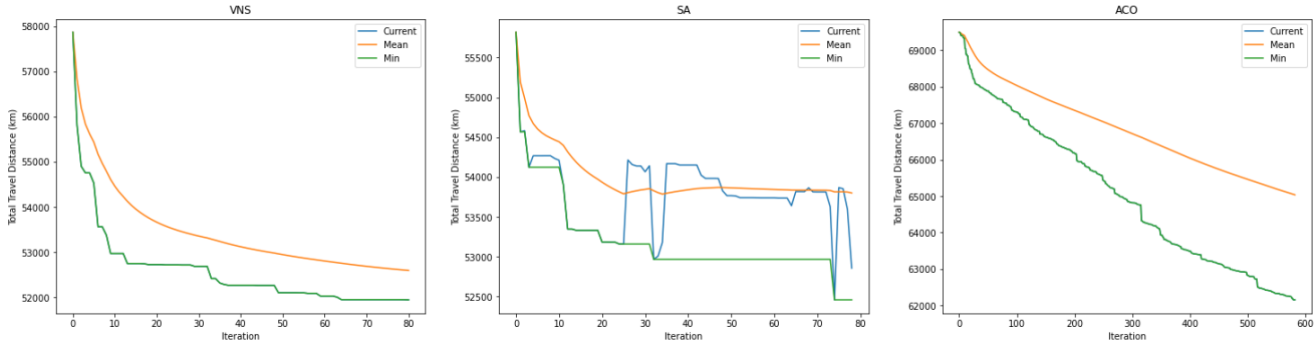


Fig. 26. Visual Comparison of Three Algorithms on Instance X-n685-k75

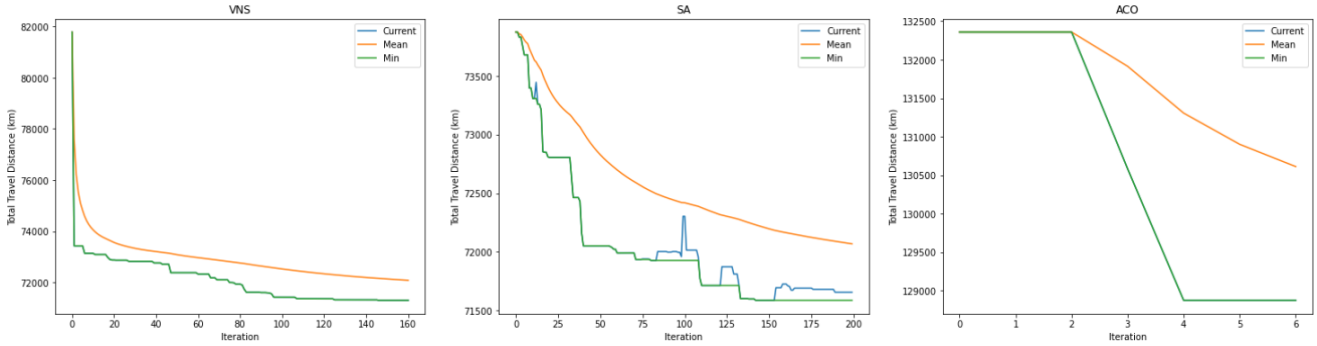


Fig. 27. Visual Comparison of Three Algorithms on Instance X-n819-k171

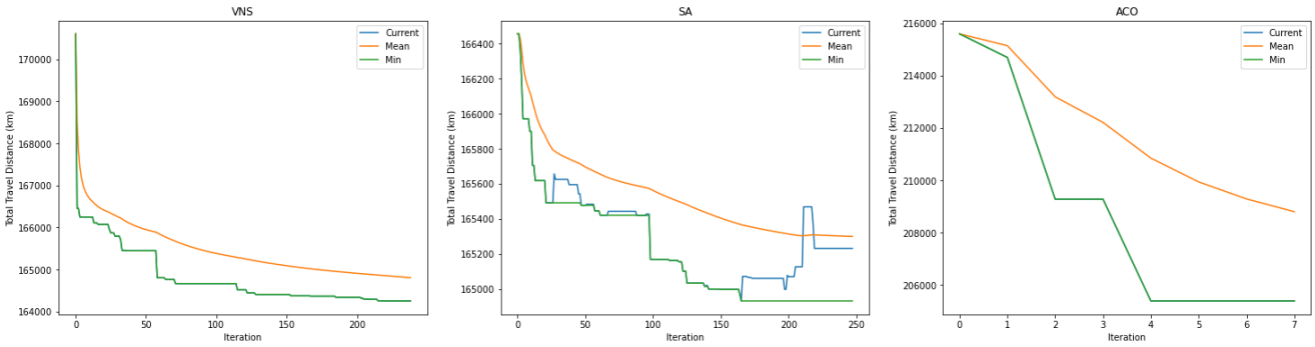


Fig. 28. Visual Comparison of Three Algorithms on Instance X-n916-k207

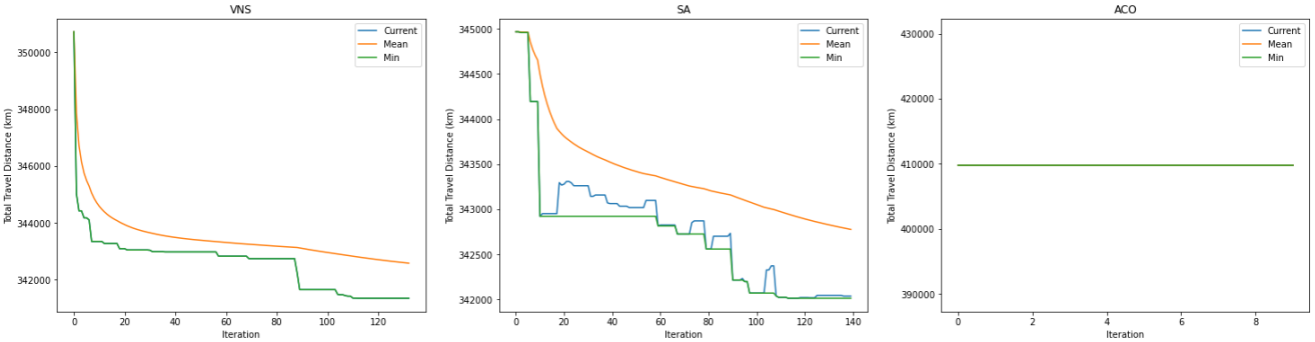
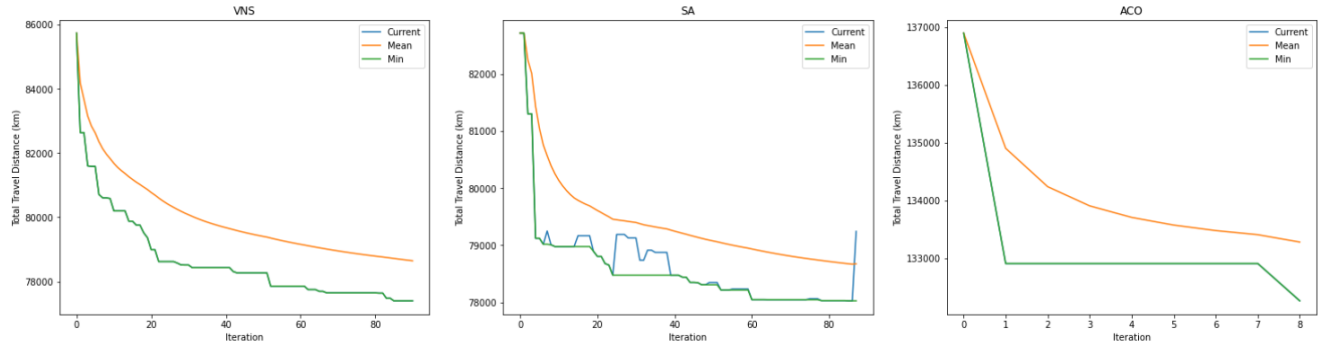


Fig. 29. Visual Comparison of Three Algorithms on Instance X-n1001-k43



REFERENCES

[1] A. Albatayneh, M. Assaf, D. Alterman, and M. Jaradat, "Comparison of the overall energy efficiency for internal combustion engine vehicles and electric vehicles," *Environmental and Climate Technologies*, vol. 24, pp. 669–680, Oct. 2020. DOI: 10.2478/rtuect-2020-00.

[2] E. von Schneidemesser, K. Steinmar, E. C. Weatherhead, B. Bonn, H. Gerwig, and J. Quedenau, "Air pollution at human scales in an urban environment: Impact of local environment and vehicles on particle number concentrations," *Science of The Total Environment*, vol. 688, pp. 691–700, 2019, ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2019.06.309. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969719328827.

[3] E. A. Rosa and T. Dietz, "Human drivers of national greenhouse-gas emissions," *Nature Climate Change*, vol. 2, no. 8, pp. 581–586, Aug. 2012, ISSN: 1758-6798. DOI: 10.1038/nclimate1506. [Online]. Available: https://doi.org/10.1038/nclimate1506.

[4] H. Shiraki, K. Matsumoto, Y. Shigetomi, T. Ehara, Y. Ochi, and Y. Ogawa, "Factors affecting co2 emissions from private automobiles in japan: The impact of vehicle occupancy," *Applied Energy*, vol. 259, p. 114 196, 2020, ISSN: 0306-

2619. DOI: https://doi.org/10.1016/j.apenergy.2019.114196. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261919318835.

[5] J. K. Dagsvik, T. Wennemo, D. G. Wetterwald, and R. Aaberge, "Potential demand for alternative fuel vehicles," *Transportation Research Part B: Methodological*, vol. 36, no. 4, pp. 361–384, 2002, ISSN: 0191-2615. DOI: https://doi.org/10.1016/S0965-8564(01)00013-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856401000131.

[6] D. Pamucar, F. Ecer, and M. Deveci, "Assessment of alternative fuel vehicles for sustainable road transportation of united states using integrated fuzzy fucom and neutrosophic fuzzy marcos methodology," *Science of The Total Environment*, vol. 788, p. 147 763, 2021, ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2021.147763. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969721028345.

[7] M. DeLuchi, "Hydrogen vehicles: An evaluation of fuel storage, performance, safety, environmental impacts, and cost," *International Journal of Hydrogen Energy*, vol. 14, no. 2, pp. 81–130, 1989, ISSN: 0360-3199. DOI: https://

doi.org/10.1016/0360-3199(89)90001-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0360319989900013.

[8] A. Y. S. Lam, Y.-W. Leung, and X. Chu, "Electric vehicle charging station placement: Formulation, complexity, and solutions," *IEEE Transactions on Smart Grid*, vol. 5, no. 6, pp. 2846–2856, 2014. DOI: 10.1109/TSG.2014.2344684.

[9] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle-routing problem with time windows and recharging stations," *Transportation science*, vol. 48, no. 4, pp. 500–520, 2014.

[10] M. Gray and W. Morsi, "On the impact of single-phase plug-in electric vehicles charging and rooftop solar photovoltaic on distribution transformer aging," *Electric Power Systems Research*, vol. 148, pp. 202–209, 2017, ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2017.03.022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779617301220.

[11] M. A. Delucchi and T. E. Lipman, "An analysis of the retail and lifecycle cost of battery-powered electric vehicles," *Transportation Research Part D: Transport and Environment*, vol. 6, no. 6, pp. 371–404, 2001, ISSN: 1361-9209. DOI: https://doi.org/10.1016/S1361-9209(00)00031-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361920900000316.

[12] J. Martínez-Lao, F. G. Montoya, M. G. Montoya, and F. Manzano-Agugliaro, "Electric vehicles in spain: An overview of charging systems," *Renewable and Sustainable Energy Reviews*, vol. 77, pp. 970–983, 2017, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2016.11.239. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032116310152.

[13] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959. DOI: 10.1287/mnsc.6.1.80. eprint: https://doi.org/10.1287/mnsc.6.1.80. [Online]. Available: https://doi.org/10.1287/mnsc.6.1.80.

[14] J. Lin, W. Zhou, and O. Wolfson, "Electric vehicle routing problem," *Transportation Research Procedia*, vol. 12, pp. 508–521, 2016, Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain, ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2016.02.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146516000089.

[15] D. B. Lenat, "The nature of heuristics," *Artificial Intelligence*, vol. 19, no. 2, pp. 189–249, 1982, ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(82)90036-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370282900364.

[16] E. K. Burke, M. Gendreau, M. Hyde, *et al.*, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013, ISSN: 1476-9360. DOI: 10.1057/jors.2013.71. [Online]. Available: https://doi.org/10.1057/jors.2013.71.

[17] L. J. Stockmeyer, "The polynomial-time hierarchy," *Theoretical Computer Science*, vol. 3, no. 1, pp. 1–22, 1976, ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(76)90061-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/030439757690061X.

[18] A. Mor and M. G. Speranza, "Vehicle routing problems over time: A survey," *Annals of Operations Research*, pp. 1–21, 2022.

[19] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter, "On the capacitated vehicle routing problem," *Mathematical programming*, vol. 94, no. 2, pp. 343–359, 2003.

[20] J. Schulze and T. Fahle, "A parallel algorithm for the vehicle routing problem with time window constraints," *annals of operations research*, vol. 86, pp. 585–607, 1999.

[21] H. Li, Z. Li, L. Cao, R. Wang, and M. Ren, "Research on optimization of electric vehicle routing problem with time window," *IEEE Access*, vol. 8, pp. 146 707–146 718, 2020.

[22] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, “How easy is local search?” *Journal of computer and system sciences*, vol. 37, no. 1, pp. 79–100, 1988.

[23] E. Aarts, E. H. Aarts, and J. K. Lenstra, *Local search in combinatorial optimization*. Princeton University Press, 2003.

[24] P. Shaw, “Using constraint programming and local search methods to solve vehicle routing problems,” in *International conference on principles and practice of constraint programming*, Springer, 1998, pp. 417–431.

[25] P. Kilby, P. Prosser, and P. Shaw, “Guided local search for the vehicle routing problem with time windows,” in *Meta-heuristics*, Springer, 1999, pp. 473–486.

[26] H. R. Lourenço, O. C. Martin, and T. Stützle, “Iterated local search: Framework and applications,” in *Handbook of metaheuristics*, Springer, 2019, pp. 129–168.

[27] M. M. Silva, A. Subramanian, and L. S. Ochi, “An iterated local search heuristic for the split delivery vehicle routing problem,” *Computers & Operations Research*, vol. 53, pp. 234–249, 2015.

[28] P. H. V. Penna, A. Subramanian, and L. S. Ochi, “An iterated local search heuristic for the heterogeneous fleet vehicle routing problem,” *Journal of Heuristics*, vol. 19, no. 2, pp. 201–232, 2013.

[29] J. Brandão, “A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem,” *European Journal of Operational Research*, vol. 284, no. 2, pp. 559–571, 2020.

[30] N. Mladenović and P. Hansen, “Variable neighborhood search,” *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[31] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, “Variable neighborhood search,” in *Handbook of metaheuristics*, Springer, 2019, pp. 57–97.

[32] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau, “An efficient variable neighborhood search heuristic for very large scale vehicle routing problems,” *Computers & operations research*, vol. 34, no. 9, pp. 2743–2757, 2007.

[33] D. Rezgui, J. C. Siala, W. Aggoune-Mtalaa, and H. Bouziri, “Application of a variable neighborhood search algorithm to a fleet size and mix vehicle routing problem with electric modular vehicles,” *Computers & Industrial Engineering*, vol. 130, pp. 537–550, 2019.

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

[35] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, “A tabu search heuristic for the vehicle routing problem with soft time windows,” *Transportation science*, vol. 31, no. 2, pp. 170–186, 1997.

[36] O. Sassi, W. R. Cherif-Khettaf, and A. Oulamara, “Iterated tabu search for the mix fleet vehicle routing problem with heterogenous electric vehicles,” in *Modelling, computation and optimization in information systems and management sciences*, Springer, 2015, pp. 57–68.

[37] F. Y. Vincent, A. P. Redi, Y. A. Hidayat, and O. J. Wibowo, “A simulated annealing heuristic for the hybrid vehicle routing problem,” *Applied Soft Computing*, vol. 53, pp. 119–132, 2017.

[38] E. Rodŕiguez-Esparza, A. D. Masegosa, D. Oliva, and E. Onieva, “A new hyper-heuristic based on adaptive simulated annealing and reinforcement learning for the capacitated electric vehicle routing problem,” *arXiv preprint arXiv:2206.03185*, 2022.

[39] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[40] B. Bullnheimer, R. F. Hartl, and C. Strauss, “An improved ant system algorithm for thevehicle routing problem,” *Annals of operations research*, vol. 89, pp. 319–328, 1999.

[41] Q. Ding, X. Hu, L. Sun, and Y. Wang, “An improved ant colony optimization and its application to vehicle routing problem with time windows,” *Neurocomputing*, vol. 98, pp. 101–107, 2012.

[42] P.-Y. Yin and Y.-L. Chuang, "Adaptive memory artificial bee colony algorithm for green vehicle routing with cross-docking," *Applied Mathematical Modelling*, vol. 40, no. 21, pp. 9302–9315, 2016, ISSN: 0307-904X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X16303249.

[43] Y. Li, H. Soleimani, and M. Zohal, "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives," *Journal of cleaner production*, vol. 227, pp. 1161–1172, 2019.

[44] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in operations research and management science*, vol. 7, pp. 225–330, 1995.

[45] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "A local-density based spatial clustering algorithm with noise," *Information systems*, vol. 32, no. 7, pp. 978–986, 2007.

[46] E. Cao, R. Gao, and M. Lai, "Research on the vehicle routing problem with interval demands," *Applied Mathematical Modelling*, vol. 54, pp. 332–346, 2018.

[47] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[48] M. Mavrovouniotis, C. Li, G. Ellinas, and M. Polycarpou, "Parallel ant colony optimization for the electric vehicle routing problem," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2019, pp. 1660–1667.

[49] H. Mao, J. Shi, Y. Zhou, and G. Zhang, "The electric vehicle routing problem with time windows and multiple recharging options," *IEEE Access*, vol. 8, pp. 114 864–114 875, 2020.

[50] X.-S. Yang, "Chapter 5 - simulated annealing," in *Nature-Inspired Optimization Algorithms (Second Edition)*, X.-S. Yang, Ed., Second Edition, Academic Press, 2021, pp. 83–90, ISBN: 978-0-12-821986-7. DOI: https://doi.org/10.1016/B978-0-12-821986-7.00012-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128219867000123.

[51] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*, Springer, 1987, pp. 7–15.