METAHEURISTIC APPLICATONS ON ELECTRIC VEHICLE TRAVELING SALESMAN PROBLEM

Jury:
Promoter:
Sabine LIMBOURG
Readers:
Maud BAY
Bart SMEULDERS

Dissertation by
Gökberk YAMAK
For a Master of Science
in Business Engineering
Academic year 2018/2019

Abstract

Metaheuristic Applications on Electric Vehicle Traveling Salesman Problem

Gökberk Yamak

HEC Liège
Master of Science in Business Engineering
Focus in Supply Chain Management and Business Analytics

Promoter: Pr. Sabine Limbourg

August, 2019

65 Pages

The energy consumption behaviors of the vehicles with electric motors are different compared to traditional internal combustion engines. As a result of regenerative braking systems, electric vehicles also have the possibility to recover energy during the journey. This feature causes a considerable variation in consumption functions, especially on negative slopes. This study focuses on finding the optimal or near-optimal TSP tours for electric vehicles on real-time fed data with the consideration of the road grades, transported loads, the speed of the vehicle, and acceleration-deceleration. These conditions mean that much more complexity in a traveling salesman problem, whose exact methods are already requiring a significant amount of computation time. The ultimate aim was obtaining high-quality solutions using efficient steepest ascent and simulated annealing metaheuristics while reducing the computation times.

Key words: electric vehicles, traveling salesman problem, energy minimization, battery management, metaheuristic, simulated annealing, steepest ascent

Acknowledgements

Throughout the writing of this dissertation, I have received comprehensive guidance and support from my supervisors. First, I would like to thank Professor Sabine Limbourg, who is the promoter of this dissertation, and Professor Maud Bay for their invaluable supervision and expertise during the whole study.

I would like to extend my gratitude and thanks to all of my professors and academic staff of the Center for Quantitative Methods and Operations Management (QuantOM), for their contributions to my professional development.

I would specially thank my colleague and partner Begüm Şekercan, for her priceless and extraordinary support during not only this study, for the last four years.

Then, I would like to thank my parents, my mother Eda Yamak and my father Gürol Yamak, for their invaluable support to me, regardless of distances and beyond the borders. Moreover, I would like to extend my thanks for my aunt Hülya Yamak, my grandparents Nermin Yamak and Halil İbrahim Yamak, and my brother Göktürk Keleş, whose contributions to my whole education life could not be ignored.

Last but not least, I would also like to thank Professor Hande Küçükaydın and Professor Yasemin Arda, for their extensive contributions to my professional skills, and their all supports during my higher education.

Table of Contents

List of Figures

List of Tables

1 Introduction

The development of modern automobiles with internal combustion engines is a super important keystone in the history of the human being. This key technology allows us to transport people, goods, and services to far distances in a relatively short amount of time. In the last few decades, the automobile industry has developed quite rapidly, and as a consequence of decreasing car-owning costs, the number of vehicles in use has increased dramatically. In the Europe including European Free Trade Association (EFTA) countries, which is a geopolitically important area for several modes of transportation, the total number of vehicles in use increased by approximately 16 million between the years 2012 and 2016, according to a recent report published by European Automobile Manufacturers Association (ACEA, 2018). Moreover, this growing number of vehicles running with petroleum-based fuels have caused and are causing severe health and environmental problems such as greenhouse gas (CO_2, CH_4, N_2O) emissions, local air quality, noise pollution, and oil dependency.

In the Transport White Paper published by the European Commission, the European Union (EU) provides a roadmap for a more sustainable European transport (European Commission, 2011). One of the most significant aims of the EU is to reach CO_2 free city logistics in major cities by 2030. It is intended to reach this goal by developing and using new and sustainable fuels and propulsion systems. The slow but steady phasing out of traditional vehicles from the urban neighborhood enables to decrease oil dependence, greenhouse gas emissions, and the other issues mentioned above. In order to satisfy European air quality requirements, some important European cities have created low-emission zones where access to metropolitan regions is barred to trucks that fulfill specific emissions. Today, the market share of electric vehicles is increasing due to the discontinuation of the use of diesel and then gasoline vehicles in the next few years.

This leads to a change in the dynamics of vehicle routing, one of the major research areas of computer science and combinatorial optimization. The main reason for this is that the fuel consumption functions of conventional vehicles with an internal combustion engine often converge to average fuel consumption values in mixed-use. Also, the extensive network of gas stations and fast fueling result in an almost uninterrupted ride. This allows time-sensitive work to be carried out without problems. On the other hand, the fuel consumption values of the currently developed electric vehicles are not so correlated with the distance taken. Thanks to the regenerative braking systems applied to the brake discs of electric vehicles, these vehicles also have the opportunity to gain energy, unlike traditional vehicles.

Due to this fact, in classical routing optimization studies conducted to date, travel costs are mostly accepted as proportional to the distance taken. However, now, this approach needs to be updated for electric vehicles. In this study, it is aimed to present a solution approach on the optimization of routing of electric vehicles which have become a popular research topic recently. In doing so, some of the parameters affecting the result, such as road slope, vehicle speed, acceleration, and deceleration factors, were taken into consideration. Besides, in the case examined, it was assumed that the vehicles were used for load distribution and the routes to return to the warehouse with the maximum battery capacity possible without charging during the journey were investigated.

This study consists of seven main sections, including this introduction. In the second section, a broad literature review has been done on the topics covered. In section three, the problem has been introduced in detail, within a mathematical framework. In the fourth section, the implementation process has been widely described from the development phase to the algorithms used to solve. In section five, the results of the study and analyzes were shared. In section six, the used project management approach, and its subprocesses have been told. So finally, in the last section, the whole study is concluded.

2 Literature Review

The question of finding the shortest possible tour, which starts and ends at a given origin point, and visits each location in a given set once, where a set of locations and distances or traveling costs between them are known leads to the traveling salesman problem (TSP). The origin of this key combinatorial optimization problem and the research field of operations research and theoretical computer science is unclear. However, the problem was primarily promoted by Hassler Whitney in 1934, and the first mathematical consideration of TSP is done by Merill M. Flood in 1937 to find optimal routes for school buses (Flood, 1956).

Traveling salesman problem has been studied extensively for more than seventy years, has many applications, subproblems, and extensions due to specific cases emerged from different applications' requirements. Transportation and logistics purposes do not limit TSP's application area. It is used for optimizing wiring scheme of computer boards by determining the shortest possible path that wire connects each circuit elements and closes the circuit, which allows computers faster processing capabilities (Lenstra & Rinnooy Kan, 1975). Other well-known applications of TSP are wallpaper cutting (Garfinkel, 1977), hole punching (Reinelt, 1992), crystallography (Bland & Shallcross, 1989), and dartboard design (Blazewicz, Eiselt, Gerd, Laporte, & Weglarz, 1991). TSP applications and extensions are widely surveyed many times (Eiselt & Sandblom, 2000; Laporte, 1992; Reinelt, 1994).

This problem is usually defined on a complete graph with a set of nodes that indicates locations, a set of arcs that indicates connections between locations and traveling costs associated with arcs. TSP can be split into two. The classical TSP model, which is also known as symmetric traveling salesman problem (STSP), assumes that each distance or traveling cost between two location points are same in each opposite direction and it can be described in a complete undirected graph. On the other hand, the asymmetric traveling salesman problem (ATSP) can be described in a complete digraph and has to consider two different distance values between each pair of locations. Since most of the proposed algorithms to solve TSP applications are developed for STSPs so far, in 1983 Jonker and Volgenant has introduced a way to transform ATSP to STSP, and showed that an asymmetric TSP with n nodes is actually equivalent to symmetric TSP with $2n$ nodes, in the worst case (Jonker & Volgenant, 1983). The algorithms to solve ATSP are also broadly surveyed (Roberti & Toth, 2012).

TSP can be modeled as an integer linear programming (ILP) program. Besides many approaches, two of them are quite notable, which are known as Miller-Tucker-Zemlin (MTZ) formulation (C. E. Miller, Tucker, & Zemlin, 1960) and Dantzig-Fulkerson-Johnson (DFJ) formulation (Dantzig, Fulkerson, & Johnson, 1954). For both ILPs, during a solver execution, linear programming (LP) relaxations are done on the constraint set regarding integrality, in most of the cases; and DFJ formulation produces a stronger LP relaxation compared to MTZ formulation that leads to a higher objective function value on relaxation (Wong, 1980). Because of that fact, for larger instances of asymmetric TSP, DFJ formulation is more in use (Velednistsky, 2017).

Traveling salesman problem's $\mathcal{NP}$-hardness is proven (Karp, 1972), even for the distances on a plane that satisfies triangular inequality, which leads to Euclidian TSP (Garey, Graham, & Johnson, 1976). In some specific cases of TSP, couple of time-efficient algorithms can be used as an exact method to solve problem (Burkard, Deineko, Dal, Veen, & Woeginger, 1998), however for the most prominent cases, exact methods come with an exponential time complexity, and computation time versus solution quality tradeoff emerges.

The brute-force method to solve TSPs can be described as examining all possible solutions for a given graph, and it is very impractical because, for symmetric TSPs, number of different tours will be $(n-1)!/2$, where the number of locations has to be visited denoted by n. This number will reach to $(n-1)!$ if the problem instance constructs an ATSP. A well-known dynamic programming algorithm reduces the worst-case running time to $O(n^2 2^n)$, but it is still too high to solve mid-size and large-size instances (Held & Karp, 1962).

In addition to them, many researchers have introduced branching algorithms, which are widely used as an exact method to solve. The branch-and-bound (B&B) algorithms' logic lay on problem relaxation, and relaxing the constraint set regarding sub tour elimination produces an assignment problem (AP), which can be solved in polynomial time (Carpaneto, Martello, & Toth, 1988). On this basis, Carpaneto and Toth proposed their well-known B&B algorithm, and they succeeded by solving a TSP with randomly generated 240 vertices, under a minute time (Carpaneto & Toth, 1980). Besides, they stated that their algorithm is CPU time-efficient; however, comes up with a memory leak problem. Other B&B approaches are also available in the literature (Balas & Christofides, 1981; D. L. Miller & Pekny, 1991; Smith, Srinivasan, & Thompson, 1977). Also, an overview of branch-and-cut (B&C) algorithms can be seen (Naddef, 2007).

Because of the TSP is an $\mathcal{NP}$-hard problem, it is quite natural that approximation algorithms, in other words, heuristic approaches, are broadly investigated. In the literature, TSP heuristics mainly classified under two streams: construction heuristics and metaheuristics (Gutin, 2009). The logic lays under construction heuristics is building a complete feasible solution from scratch, while metaheuristics are seeking to improve solution quality by starting from an already constructed tour, and improving it iteratively.

According to the needs of different specific problems, various heuristics has developed and used as construction heuristics. However, some well-known heuristics used in many pieces of research and their computational performances are experienced. One of the most known and most apparent TSP heuristics is called nearest neighbor (NN) algorithm. In NN, the algorithm starts with a random selection of a vertex on the graph as the origin point if it is not stated, then adds the nearest unvisited location into the tour in each step, until all vertices become visited, and at the end returns to the chosen origin point. NN also has a variant called repetitive nearest neighbor (RNN) algorithm, which returns the tour with best objective function value, after repeating the NN algorithm starting from each vertex on the graph.

Another most known approximation algorithm is greedy heuristic (GH). On a directed graph, all arcs in the set are sorted according to their lengths or costs, and they are added into solution considering feasibility conditions of TSP, accordingly. NN, RNN, and GH algorithms are widely examined in the literature; however, their exact origins are unclear. Excellent performance analysis can be found for those algorithms (Gutin, Yeo, & Zverovich, 2002). Other famous approximation guaranteed heuristic algorithm is known as Christodes algorithm, which is dealing with Euclidian TSP and approximates the solution to the minimum spanning tree (MST) of the graph with a guarantee of an objective function value at most 0.5 times higher than the optimal value (Christofides, 1976). More extensive overviews of construction heuristics can be referred (Johnson et al., 2002; Johnson & McGeoch, 2002).

On the other hand, metaheuristics or improvement heuristics, which are mainly related to this study, in general, are known to be typically faster than exact methods, and they usually produce pretty good solutions that can be evaluated as near-optimal, by doing local-search, global-search or following a hybrid strategy. Local search algorithms that use edge exchange are the most studied TSP improvement algorithms in which a tour is improved by replacing k with edges that are not in the solution. One of the earliest metaheuristic applications on the traveling salesman problem was made using the well-known simulated annealing (SA) algorithm (Kirkpatrick, Gelatt, & Vecchi, 1983). The same paper introduced the simulated annealing algorithm as an optimization method by observing the analogy with the annealing process. Other most common metaheuristic applications in the literature related to traveling salesman problem are a genetic algorithm (GA) and iterated tabu search (ITS) methods. One of the very first introductions to the genetic algorithm can be found in the refereed literature

(Mitchell, 1996). Also, in the literature, there are many applications on TSP, and its extensions can be found (Sze & Tiong, 2007).

Contrarily, ITS is a comparably newer approach whose one of the first TSP applications is done after the millennium (Misevičius, 2004). Also, new generation approaches are becoming increasingly popular (Antosiewicz, Koloch, & Kamiński, 2013). Two of these noticed approaches' details, which are harmony search and swarm optimization, can be found in the literature (Geem, Kim, & Loganathan, 2001; Zhang, Sun, Wang, & Yang, 2007).

Another related problem to TSP is vehicle routing problem (VRP). It has been a well-studied problem where the objective is to create efficient vehicle routes between the depot and a set of customers for deliveries while satisfying particular constraints. Different optimization methods that are developed for the VRPs has immensely contributed to the lower costs also to the reduction of congestion environment and noise pollution. However, aside from the benefits of VRP optimization, today, one of the most important actors on the road to the -free city logistics is the electric engines. It is found that electric vehicles (EV) are suitable for stop-and-go movements due to its regenerative braking system (Pelletier, Jabali, & Laporte, 2014).

When considering a fleet that consists of EV's for urban freight transport, the biggest issue is that the fact that classical VRP and its variants consider that internal combustion engines perform the routes. However, EV's short driving ranges, long battery recharging times, and the limited charging infrastructure; cannot be presented by the classical VRP and its variants. The least energy consumption route is needed rather than the shortest one determined by the classical VRPs. This is why new routing algorithms have to be developed for EV. In order to develop these algorithms, the main factors in energy consumption which are the mass of the vehicle and payload, engine efficiency, vehicle speed, drive pattern, road grade, and vehicle rechargeability while driving has to be investigated (Baum, Dibbelt, Pajor, & Wagner, 2013; Bektaş & Laporte, 2011; Kara, Kara, & Yetiş, 2007; Touati-Moungla & Jost, 2012).

Even though EVs are developed recently, there are few studies where finding the most energy-efficient route was addressed. In another study, the routing of electric vehicles from a graph-theory point of view is studied (Artmeier, Hasselmayr, Leucker, & Sachenbacher, 2010). The energy-optimal routing problem is modeled as a shortest path problem where there are constraints on the vehicle's charge level so that it cannot be negative and cannot exceed the maximum capacity of the battery. Weights of the edges are allowed to take negative values to portray the recapturing of energy from the regenerative brake. Simple algorithms are given as solution methods. Their results are improved by using Dijkstra's algorithm to enable creating routes for EVs on large networks (Eisner, Funke, & Storandt, 2011). In 2011, an idea was introduced to bound different forms of energy, which results in a consistent heuristic function (Sachenbacher, Leucker, Artmeier, & Haselmayr, 2011). This heuristic improves the results of related literature by order of magnitude with their $O(n^2)$ algorithm (Artmeier et al., 2010).

Up to this point, none of these studies takes into account the recharging decisions at nodes. Considering that the vehicle must recharge on its route, the problem of finding the minimum cost of the path for this EV is modeled as a dynamic program (Sweda & Klabjan, 2012). The Green VRP (GVRP) where vehicle routes and recharging of vehicles at alternative fuel stations are determined simultaneously is introduced. The GVRP is modeled as a mixed-integer linear program. The objective is to find a set of vehicle tours with minimum distance. Each vehicle in the problem, starts from the depot and visit a set of customers and returns to the depot without exceeding the vehicle's driving range. Tour can contain a stop at one or more AFSs so that the vehicle can recharge. As a solution method, a modified Clarke and Wright savings heuristic and a density-based clustering algorithm to create a set of feasible tours are used. Afterward, a post-optimization phase takes place (Erdoğan & Miller-Hooks, 2012). Different from the previous study, a MIP formulation for GVRP where multi depots and visit to more than one AFS are allowed, is introduced (Taha, Fors, & Shoukry, 2014). In Alejandro Montoya's paper, they solved the GVRP with a modified multi-space sampling heuristic which includes a set of three randomized heuristics, a tour partitioning procedure, and a set partitioning formulation. A repair mechanism optimally inserts visits to refueling stations to restore the feasibility of routes violating the vehicle's fuel constraint (Montoya, Guéret, Mendoza, & Villegas, 2016).

Another extension is studied on an electric VRP with time windows (EVRPTW) (Schneider, Stenger, & Geoke, 2014). A linear charging rate and charging the batteries entirely are the two assumptions made. They developed one of the earliest metaheuristic approaches that combine variable neighborhood search (VNS) and iterative tabu search and have proposed benchmark instances for EVRPTW. Building on the work of Michael Schneider's, Keskin and Çatay worked on EVRP that allows partial charging at charging stations. Their results on Schneider's benchmark instances showed that when the partial recharge option is modeled as a continuous decision variable, it yielded better results (Keskin & Çatay, 2016).

Thus far, the charging functions were assumed to be linear, whereas they are nonlinear in reality. A nonlinear charging function for EVRP is introduced, and a hybrid metaheuristic which combines an iterated local search (ILS) and a heuristic concentration (HC) that tries to build global optimum using parts of the local optima found during a heuristic search procedure is developed (Montoya, Guéret, Mendoza, & Villegas, 2017). Recently, the aspect of capacitated charging stations is studied with a nonlinear charging function. The proposed solution method is route-first, assemble-second metaheuristic (Froger, Mendoza, Jabali, & Laporte, 2017). One most recent work is by Keskin, where EVRP with time windows including queuing at charging stations is studied. They proposed a metaheuristic that combines adaptive large neighborhood search and the solution of a MILP (Keskin, Laporte, & Çatay, 2019).

In addition to all of the literature mentioned above, in the recent working paper of Bay and Limbourg, they focused on ETSP, where cities are given the aim is finding the optimal directed cycle with the introduced mixed-integer non-linear programming (MINLP) model, which maximizes the level of available energy at the end of the tour (Bay & Limbourg, 2017). Unlike most of the previous studies, the recharging operation was not taken into account in their study; also the introduced objective function considers some of the critical factors of energy consumption, which are the speed of the vehicle, road grades, transported load, and regeneration ability, in addition to the distance traveled.

This study is based mainly on Bay and Limbourg's study. Thereon, in addition to considered factors of energy consumption or regeneration functions used, the calculation is developed by also considering the effect of acceleration and deceleration factors as proposed in the relevant literature (Ehsani, Gao, Gay, & Emadi, 2005). Besides, by taking into consideration the dramatically increasing CPU times of exact method depending on the size of the graphs, this problem has been aimed to be solved by well-known local search methods, which are the steepest ascent and simulated annealing metaheuristics. Also, considerations on a more accurate collection of road slopes in real-life applications were discussed.

3 Problem Description

Traditional passenger cars, motorcycles, buses, trucks take power they need from internal combustion engines (ICE). Vehicles that have been using gasoline, diesel, or liquid petrol gas (LPG) are still in use with a very high ratio, and they are still dominating the vehicle market. Depending on this fact so far, most of the researches in the field of vehicle routing optimization and its sub-problems have been done considering the nature of traditional vehicles. However, the fuel (energy) consumption performance of vehicles with ICEs is relatively less affected by real-time changing environmental factors than vehicles with electric engines (EE) currently being developed. Consequently, conventional parametric cost matrices, which are used in previous studies, depending on distance, time, or composite factors, are not sufficient for routing of electric vehicles.

The energy consumption functions, of electric vehicles depend on many environmental factors and are significantly affected by them. Against the continuous fuel consumption values of the ICEs which do not deviate significantly from the average, there is the potential to gain energy as a result of the braking used at stop-and-go moments or on the negative slopes of the electric vehicles in addition to their energy consumptions. During these braking moments, a large amount of heat is released on the brake discs, which tend to damp the vehicle's current momentum while trying to slow it, due to the loss of kinetic energy. This energy loss can be recovered partially by an energy recovery mechanism called regenerative braking that converts released heat to electric energy. Where applicable, this mechanism allows electric motors to recover and store energy, limited by the vehicle's battery capacity.

The factors and related parameters, which are the inputs of energy consumption or energy regeneration functions can be investigated under two main categories. The first category is environmental factors, which depend on constant values assumed to be same in everywhere on the planet Earth, such as gravitational acceleration (g) and the density of air (ρ), and experimental coefficients such as rolling resistance on car tire (c_r) and drag resistance (c_d). On the other hand, a second category can be mentioned called vehicle parameters, which heavily depend on the vehicle's itself, such as the mass of the vehicle (M_v), frontal surface area (A) that interacts with air resistance, regeneration coefficient (α), overall gear reduction ratio (ε_0), mass factor (δ), auxiliary energy consumption rate (p^{aux}) and battery capacity (C_{max}). In addition to them, to calculate energy consumptions or regenerations in a dedicated time interval, a constant acceleration (a^+) and deceleration (a^-) rates, and constant speed (v) can be given into the function as input parameters.

Within the context of this study, a TSP application for electric vehicles is examined with a vehicle that has to distribute orders to customer nodes with sufficient loading capacity, departing from a specified depot location and closing its tour at the same point. Since the energy consumption or regeneration depends on the field profile and ascending-descending have an essential effect on it, the problem becomes an asymmetric TSP. The electric vehicle traveling salesman problem (ETSP) can be represented in a complete undirected graph $G = (N, A)$, where N is the set of nodes indexed by $i, j \in \{0, \ldots, |N| - 1\}$, and A is the set of arcs defined as $A = \{(i,j) : i,j \in N, \ i \neq j\}$. A graph representation example with four nodes is as follows.

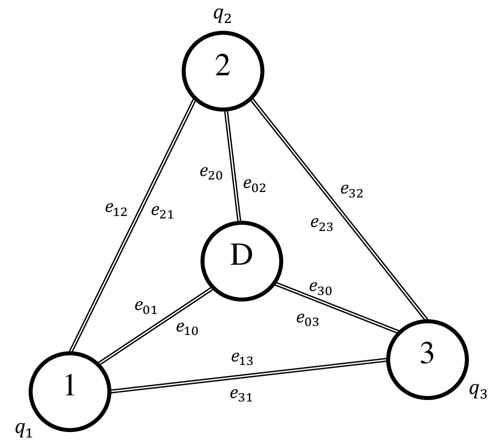

Figure 1: Graph Representation of the Problem

As can be seen in the above graph, on an arc between node i and node j, energy consumption or regeneration denoted as e_{ij} in both directions, and each node except depot location has a non-negative demand denoted q_i. Considering this is a TSP application, and the vehicle is obliged to visit all nodes in a single tour, the total demand of customers has to be load on the vehicle at the beginning, and be distributed accordingly. Where the total load carried between node i and node j is represented as m_{ij}, on a non-negative slope $\theta_{ij}^{\geq 0}$ or during acceleration with constant a^+ at a particular time t, the energy consumption can be calculated as follows (Ehsani et al., 2005).

$$p_{ij}^{out} = v\left(\left(M_v + m_{ij}\right)gsin\theta_{ij}^{\geq 0} + 0.5c_d\rho A v^2 + c_r\left(M_v + m_{ij}\right)gcos\theta_{ij}^{\geq 0} + \left(M_v + m_{ij}\right)\delta a^+\right)$$

Conversely, on negative slopes θ_{ij}^{-} or while decelerating with constant a^- at a particular time t, the energy regeneration (gain) will be as below.

$$p_{ij}^{in} = \alpha v \left(\left(M_v + m_{ij}\right) g sin\theta_{ij}^{-} + 0.5 c_d \rho A v^2 + c_r \left(M_v + m_{ij}\right) g cos\theta_{ij}^{-} + \left(M_v + m_{ij}\right) \delta a^{-} \right)$$

Regardless of the slope or acceleration-deceleration, each vehicle has a constant energy consumption due to its auxiliary units, such as climate control, lights, radio, if applicable cooler, and other similar parts. If the momentary energy consumption of the auxiliary units of the vehicle (p^{aux}) is added to the equation, the total energy consumption between node i and node j can be calculated with the following equation.

$$e_{ij} = \int_{out} p_{ij}^{out} dt + \int_{in} p_{ij}^{in} \, dt + \int p^{aux} \, dt$$

In 2015, Bay and Limbourg proposed a mixed-integer non-linear programming model (MINLP) to solve ETSP, which seeks to maximize the level of available energy at the end of the tour, without any intermediate charging stations (CS) and considering variable speed, loading and road grade parameters (Bay & Limbourg, 2015). They denoted the level of available energy at each node $j \in N$ with L_j, and assumed the available energy level at the beginning of the tour (L_0) is known. As a feasibility condition, they limited the upper bound of L_j with the maximum capacity of the battery (C_{max}), even the energy recovery mechanism captures more, and the lower bound with zero. Under these circumstances, where x_{ij} is a binary decision variable that takes the value of 1 if arc $(i, j) \in A$ is traversed and 0 otherwise, the level of energy can be calculated as follows.

$$L_j = \begin{cases} 0, & if \sum_{i=0}^{|N|-2} \left(L_i - e_{ij}\right) x_{ij} \leq 0 \\ C_{max}, & if \sum_{i=0}^{|N|-2} \left(L_i - e_{ij}\right) x_{ij} \geq C_{max} \\ \sum_{i=0}^{|N|-2} \left(L_i - e_{ij}\right) x_{ij}, & otherwise \end{cases}$$

They also limited the total travel time not to exceed T. Bay and Limbourg updated their MINLP approach in their working paper from 2017 by instead of using a variable speed, they preferred to fix speed parameter to an optimal speed (v^*) that minimizes energy consumption (Bay & Limbourg, 2017). From this point of origin, their model can be developed to take into account acceleration-deceleration also, while calculating energy consumption or regeneration, and the developed model can be stated as follows.

$$Maximize \quad L_{|N|-1} \tag{1}$$

subject to:

$$\sum_{j=1}^{|N|-1} x_{ij} = 1 \quad \forall i \in N \setminus \{|N|-1\} \tag{2}$$

$$\sum_{i=0}^{|N|-2} x_{ij} = 1 \quad \forall j \in N \setminus \{0\} \tag{3}$$

$$u_0 = 1 \tag{4}$$

$$u_{|N|-1} = |N| \tag{5}$$

$$2 \leq u_i \leq |N|-1 \quad \forall i \in N \setminus \{0, |N|-1\} \tag{6}$$

$$u_i - u_j + 1 \leq (|N|-1)(1-x_{ij}) \quad \forall i \in N, j \in N \setminus \{0\} \tag{7}$$

$$m_{0j} = \sum_{j=1}^{|N|-1} q_j \quad \forall j \in N \setminus \{0\} \tag{8}$$

$$m_{i,|N|-1} = 0 \quad \forall i \in N \setminus \{|N|-1\} \tag{9}$$

$$m_{jk} = m_{ij} - x_{ij} q_j \quad \begin{array}{l} \forall i \in N \setminus \{|N|-1\} \\ j, k \in N \setminus \{0\}, k \neq j \end{array} \tag{10}$$

$$M_{ij} = M_v + m_{ij} \quad \forall i \in N, j \in N \setminus \{0\} \tag{11}$$

$$\begin{aligned} p_{ij}^{out} &= M_{ij} g v \sin \theta_{ij}^{\geq 0} + 0.5 c_d \rho A v^3 \\ &+ c_r M_{ij} g v cos\theta_{ij}^{\geq 0} + M_{ij} v \delta a^{+} \end{aligned} \quad \forall (i,j) \in A \tag{12}$$

$$\begin{aligned} p_{ij}^{in} &= -\alpha (M_{ij} g v sin\theta_{ij}^{-} + 0.5 c_d \rho A v^3 \\ &+ c_r M_{ij} g v cos\theta_{ij}^{-} + M_{ij} v \delta a^{-}) \end{aligned} \quad \forall (i,j) \in A \tag{13}$$

$$e_{ij} = \left(\int_{out} p_{ij}^{out} dt - \int_{in} p_{ij}^{in}\, dt + \int p^{aux} dt \right) \quad \forall (i,j) \in A \tag{14}$$

$$L_i \leq C_{max} \quad \forall i \in N \tag{15}$$

$$L_j = \sum_{i=0}^{|N|-2} \left(L_i - e_{ij}\right) x_{ij} - s_j \quad \forall j \in N \tag{16}$$

$$x_{ij} \in \{0,1\}, \quad e_{ij} \in \mathbb{R}, \quad m_{ij}, p_{ij}^{out}, p_{ij}^{in} \in \mathbb{R}_{\geq 0} \quad \forall (i,j) \in A \tag{17}$$

$$L_i \in \mathbb{R}_{\geq 0}, \quad s_j \in \mathbb{R}_{\geq 0}, \quad u_i \in \mathbb{N} \quad \forall i \in N, \forall j \in N \tag{18}$$

In the above mathematical formulation, the objective function (1) aims to maximize the level of available energy at the end of the tour. Constraint set (2) states that the vehicle leaves each node i strictly once, while the constraint set (3) ensures that the vehicle enters each node j exactly once. Constraints (4) - (5) and constraint sets (6) - (7) are sub tour elimination constraints proposed in MTZ formulation that uses integer variables u_i, where $i \in N$, to determine the sequence number in each node i is visited (C. E. Miller et al., 1960). Constraints (4) and (5) indicate that starting node ($i = 0$) and ending node ($i = |N| - 1$) are the depot location. Constraint set (6) ensures that for each node i except the starting and ending nodes, the decision variable u_i should be in a closed interval between 2 and $|N| - 1$. Constraint set (7) defines the order of each node by satisfying $u_j \geq u_i + 1$, where $x_{ij} = 1$ or by satisfying $u_i - u_j \leq |N| - 3$ in the case of $x_{ij} = 0$. Constraint set (8) indicates that the load carried between the starting node 0 and each node j excluding the starting node should be equal to the total demand of all nodes in the graph, while constraint set (9) is stating that the load carried between any node i and the ending node $|N| - 1$ should be equal to zero. Constraint set (10) controls the load carried between all customer nodes by equalizing load carried between node j and node k (m_{jk}) to load carried between node i and node j (m_{ij}) minus the demand of node j (q_j) if the arc between node i and node j is traversed. Constraint set (11) defines the total mass moved on arc (i, j) by adding the mass of vehicle and load carried on the arc. Constraint set (12) defines the momentary energy consumption on each arc (i, j), where $\forall (i, j) \in A$, if the road grade ($\theta_{ij}^{\geq 0}$) is non-negative, or if the vehicle is accelerating with constant a^+. Similarly, constraint set (13) controls the momentary energy regeneration (gain) on each arc (i, j), where regeneration coefficient (α) is greater than zero, if the road grade (θ_{ij}^{-}) is negative, or if the vehicle is decelerating with constant a^-. Constraint set (14) calculates the total energy consumption or regeneration on each arc (i, j) by integrating the battery outputs and battery inputs over the associated time interval and summing them up. As mentioned earlier, the constraint set (15) bounds the available energy amount at each node i with the maximum battery capacity of the vehicle. In constraint set (16), the level of energy at node j is equalized to the level of energy at node i minus energy consumption or regeneration occurred on arc (i, j), if arc (i, j) is traversed ($x_{ij} = 1$). A non-negative slack variable s_j holds the excess energy if L_j tries to exceed the maximum allowed battery capacity. Lastly, constraint sets (17) - (18) states the binary, integer, non-negative, and unrestricted conditions on decision variables. No constraint is introduced to limit the total travel time of the vehicle, which is not considered within this study.

Since the mathematical formulation of ETSP that considers road grade, weight, speed and acceleration, leads to an MINLP model, which is $\mathcal{NP}$-hard and contains many computational difficulties as a consequence of combining mixed integer programming (MIP) that comes with a combinatorial difficulty of doing optimization over discrete decision variables, and non-linear programming (NLP), which possibly comes with discontiguous feasible regions that may contain the optimal solution anywhere inside it, exact methods' computation times are quite high (Bay & Limbourg, 2017). Considering this fact, this study has aimed to solve the defined problem with two metaheuristics' applications to reduce computation times while obtaining optimal or near-optimal solution sets. Implementation details and obtained results can be found below under the following sections.

4 Implementation

Since this study is based on algorithm design and testing procedures, the backbone of this study is the implementation process. This section is divided into seven subsections. In subsection 4.1, the hardware and software, which were used during implementation, were specified. Under 4.2, details of required inputs for implementation, and their collection processes were discussed. In subsection 4.3, the way to compute the objective function value of a given solution set was introduced. In subsections 4.4 and 4.5, the upper bound calculation method and initial solution generating methods were told, respectively. In 4.6, the neighborhood structure and neighbor selection processes, which are mandatory to execute local search methods, were proposed. Finally, under 4.7, the metaheuristics, which are the main subject of this study, were described in detail.

4.1 Implementation Environment

The complete development related to this study has been done using Java™ programming language with Java™ SE Development Kit (JDK) 8, update 102 on Java™ SE Runtime Environment (JRE) 8, update 121 (Oracle Corporation, 2014). There are several reasons for this decision. First of all, Java™ programming language is an object-oriented (OO) programming language which allows users to define classes of objects more precisely. This OO nature also allows developing specific algorithms for creating classes and objects which are derived from those classes. The second reason is, it is a well-documented computer language and easy to manage. Last but not least, the third reason is, it comes with excellent memory management, and it is working significantly faster than other interpreted languages in use.

For data storing and reading purposes, MongoDB Community Server 4.0.4 is used (MongoDB Inc., 2018a). MongoDB is a NoSQL, documented-oriented database program that allows storing files and mixed arrays, besides standard data types. Finally, as an integrated development environment (IDE), IntelliJ IDEA CE 2018.3 is preferred (JetBrains s.r.o, 2018). All developments and computations are done on a personal laptop computer running macOS Mojave (10.14.3) with 2.7 GHz Intel Core i5 processor and 8 GB DDR3 memory.

4.1.1 External Libraries

During the implementation phase, in addition to JDK 8u121's internal libraries, three external libraries are mainly used in the delivery of essential jobs. The first external library is MongoDB Java Driver 3.9.1 to establish the synchronous interaction between the Java program and the MongoDB server (MongoDB Inc., 2018b). The other external library is the callable library of Gurobi Optimizer 7.5.2 for Java™ programming language (Gurobi Optimization, 2017). Gurobi Optimizer is used to calculate upper bounds for objective functions, which will be discussed in detail later in section 4.4. The last external library that used for this study is JFreeChart 1.0.19, which is used to generate XY charts for objective function's value changes according to the number of iterations performed during metaheuristic algorithm runs (Object Refinery Ltd., 2017).

4.2 Gathering Inputs

Under this subsection, the ways of obtaining the required components to solve the previously described problem are discussed. In subsection 4.2.1 the parameters related to the vehicle, in 4.2.2 the parameters related to the physical environment, in 4.2.3 used test instances, in 4.2.4 the method of calculating road grades and distances, and finally in 4.2.5, the way of collecting nodes' elevation information was presented.

4.2.1 Vehicle Parameters

For this problem, the specifications of the vehicle affect the result dramatically. The most affecting specifications can be parametrized and be introduced into the mathematical formulation. According to the energy consumption or regeneration functions used in this study the most prominent specifications are mass of the vehicle (M_v), maximum battery capacity (C_{max}), momentary energy consumption of auxiliary units (p^{aux}), frontal surface area (A), regeneration coefficient (α) and mass factor (δ), which comes from Newton's second law (Ehsani et al., 2005). In addition to them the acceleration (a^+) or deceleration (a^-), and the speed of the vehicle (v) can be parametrized to simplify computations.

Most of the vehicle parameters considered in this study are collected from the previous studies (Bay & Limbourg, 2017; Ehsani et al., 2005). The used regeneration coefficient (α) value is the given value for the European Driving Cycle type ECE-1. The mass factor is calculated with the below formula, where ε_0 represents the overall gear reduction ratio, δ_1 represents the effect of the angular moment of wheels, and δ_2 represents the effect of the power plant-associated rotating parts.

$$\delta = 1 + \delta_1 + \delta_2 * \varepsilon_0^2$$

According to the studies mentioned above, δ_1 and δ_2 values can be estimated as 0.04 and 0.0025, respectively. ε_0 value is heavily depending on the considered vehicle's gear system, but for this study, it has been assumed as 3.8, which is reasonable. Acceleration (a^+), deceleration (a^-), and speed (v) parameters are supposed to be constant. a^+ and a^- values are user-defined constant parameters. On the other hand, v value can be calculated to minimize the energy consumption on any arc (i, j) by doing the following differentiation.

$$\left(\frac{\partial}{\partial v}\right)\left(\left(p_{ij}^{out} - p_{ij}^{in} + p^{aux}\right) * \frac{d_{ij}}{v}\right)$$

$$= \left(\frac{\partial}{\partial v}\right)\Bigg(\Big(v\big(M_{ij}gsin\theta_{ij}^{\geq 0} + 0.5c_d\rho Av^2 + c_rM_{ij}gcos\theta_{ij}^{\geq 0} + M_{ij}\delta a^+\big)$$

$$- \alpha v\big(M_{ij}gsin\theta_{ij}^{-} + 0.5c_d\rho Av^2 + c_rM_{ij}gcos\theta_{ij}^{-} + M_{ij}\delta a^-\big) + p^{aux}\Big)\left(\frac{d_{ij}}{v}\right)\Bigg)$$

Differentiating above equations to zero leads to the solution below, where the optimal value of speed denoted as v^*.

$$v^* = \sqrt[3]{\frac{p^{aux}}{Ac_d\rho * (1 - \alpha)}}$$

As can be seen above, the optimal speed is not depending on any property of any arc (i, j), which means that the speed parameter can be assumed constant over the complete journey. All parameters related to the vehicle can be found in the following table.

$\boldsymbol{M_v}$	$\mathbf{2200\ kg}$
$\boldsymbol{C_{\max}}$	$24000\ Wh$
$\boldsymbol{p^{aux}}$	$1000\ W$
$\boldsymbol{A}$	$3.2\ m^2$
$\boldsymbol{\alpha}$	0.2587

$\boldsymbol{\delta}$	1.0761
$\boldsymbol{a}^+$	$2\ m/s^2$
$\boldsymbol{a}^-$	$-2\ m/s^2$
$\boldsymbol{v}$	$9.510864\ m/s$

Table 1: Vehicle Parameters

4.2.2 Physical Environment

Environmental factors are assumed to be constant at everywhere in the Earth, such as the gravitational acceleration (g), the density of air (ρ), the rolling resistance (c_r) and the drag resistance (c_d). c_r and c_d values are overall values used in literature, and they emerge from the interaction between vehicle's tires and the surface of roads. The environmental parameters used in this study can be found below.

g	$9.81\ m/s^2$
ρ	$1.225\ kg/m^3$
c_r	0.015
c_d	0.4

Table 2: Environmental Parameters

4.2.3 Test Instances

During the conceptual design and following the testing process, four instance files have been used. All locations in test instances are randomly selected locations around Liège, Belgium. Two of these four instances contain 6 customer nodes plus the depot location. The third file contains 10 customer nodes, and the fourth file contains 14 customers. The second and the third instances are common instances with the study of Bay and Limbourg (2017), which makes evaluation easier. In addition to coordinates of locations, instance files include a Boolean field to indicate the location is depot or customer, and a field stores the demands of locations. Test instances can be seen in Appendix A.

4.2.4 Directions Calculation

In order to calculate distances and directions between two nodes, three different approaches can be discussed. As mentioned earlier in the problem description, while calculating energy consumption or regeneration of electric vehicles, distance information is not enough alone. The slope of the road, in other words, the field profile, has vital importance to do more precise calculations. Since the distance between node i and node j is a property of the arc (i, j), in fact, the arc definitions can be done in these three different natures. The very first type of arcs are called simple arcs, which only contain the beginning node and end nodes on it, and a single slope value can be calculated among the arc with the elevation information of beginning and end nodes. The visual representation of the simple arcs is following.

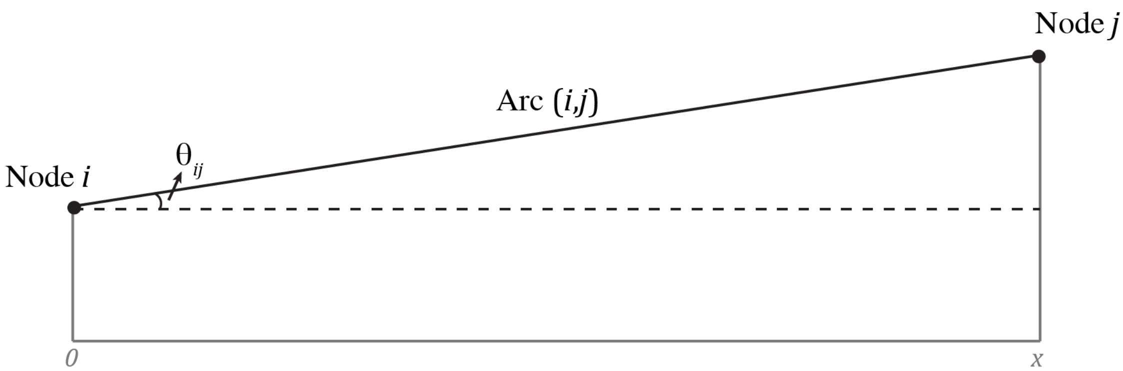

Figure 2: Visual Representation of Simple Arcs

The second proposed arc type is high-resolution arcs, which mean that by increasing the number of nodes on an arc, the road grade and the distance information collected between node i and node j can be increased as well. With this kind of an approach, the intermediate nodes on a straight line come with multiple slopes and coordinate information can represent the field profile in more detail, compared to a simple arc. The resolution (D), in other words, the number of nodes on the arc, is an input of the user. Where resolution value is set to 2, the arc becomes a simple arc. A visual representation of a high-resolution arc, with 11 nodes on it, can be found in Figure 3, which is below.

Due to the feature of the used API method, the intermediate nodes chosen by the method are equal distance apart. These equal distances correspond to the distance that calculated by dividing the length of vector projection of the straight line between node i and node j to $D - 1$.

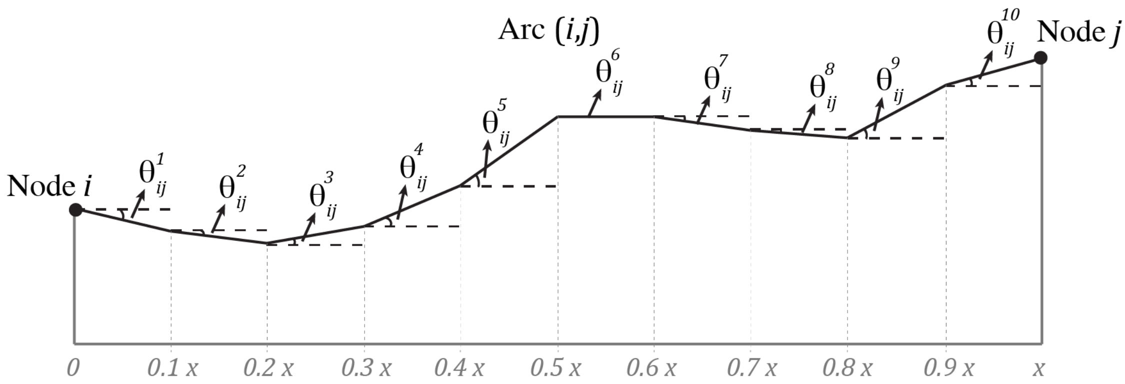

Figure 3: Visual Representation of High-Resolution Arcs (D = 11)

For the above arc types, the actual distances (d_{ij}), which is occurred because of the shape of the Earth, between two nodes are directly calculated from longitudes and latitudes thanks to the Haversine formula, where latitudes are φ_i and φ_j, longitudes are λ_i and λ_j, and the Earth's radius R, below.

$$\Delta\varphi = |\varphi_j - \varphi_i|$$

$$\Delta\lambda = |\lambda_j - \lambda_i|$$

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_i) * \cos(\varphi_j) * \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 * \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d_{ij} = R * c$$

The third and the last arc type is real-time arcs that are constructed by querying real-time directions considering road closures and traffic, between beginning and end nodes by using Google Maps Directions API (Google Inc., 2019a). Any request sent to Google Cloud returns an answer with a JSON file that describes every action should have taken while traveling between two nodes. This type of arcs has been programmed to include intermediate nodes that created on each action point, which indicates the actions like turning left or right, exiting from roundabouts and so on. Also, the real lengths of trajectories are directly calculated by parsing JSON files, so which means the distances between demand nods and the depot, are as close as possible to the real-life. The aim of using these arc type getting real field profiles on arcs and making more consistent decisions. A visual representation for the real-time arcs is as follows.

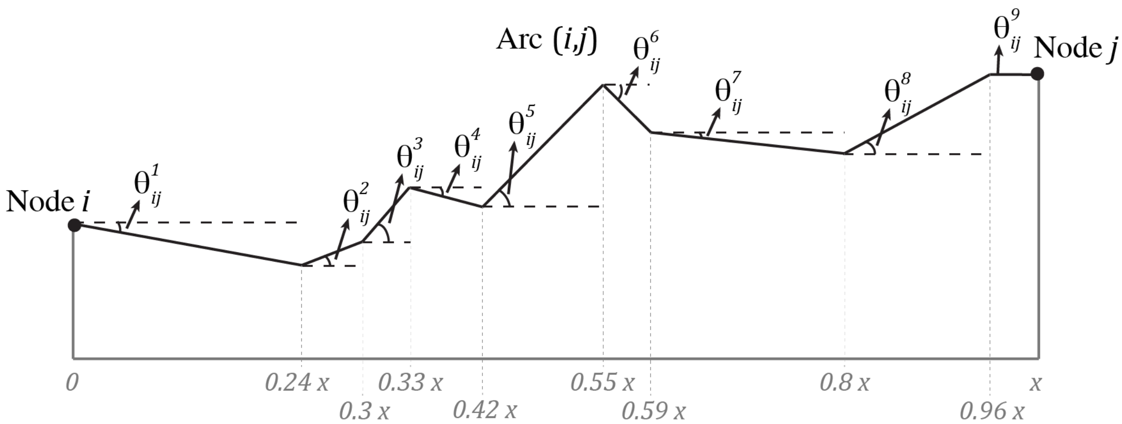

Figure 4: Visual Representation of Real-Time Arcs

Among these three proposed arc types, the arc construction process time goes up, respectively, because of the number of requests and answers processed. Besides, the real-time arcs are not symmetric between two nodes; it results in doubled arc creation. Creation times are quite reasonable for small instances, for example, the arc creation on a node-set with 10 customers, takes 1.5 minutes; however, for a larger set that includes around 20 nodes, the construction time can increase up to 5 minutes. To avoid the repetition of this time-consuming task, all nodes and constructed arcs are stored in a MongoDB database works on localhost, following their initial creation.

4.2.5 Elevation Data

With the purpose of collecting elevation data on a profile, Google Maps Platform's Elevation API service has been used (Google Inc., 2019b). Built-in API methods, provide users the altitudes of given locations pointed by their latitudes and longitudes. After having the directions information, the pre-determined breaking points' coordinates are queried to collect altitude data, which is used to calculate road grades between all specified nodes on an arc. Where between any node i and any node j, the road grade is denoted as θ_{ij}, the distance denoted as d_{ij}, and the altitudes are γ_i and γ_j, respectively, the road grade can be calculated in angular units as follows.

$$\theta_{ij} = \arctan\left(\frac{\gamma_j - \gamma_i}{d_{ij}}\right)$$

As mentioned above, the calculated road grade (θ_{ij}) is an essential input for energy consumption or energy regeneration calculations.

4.3 Fast Computation of the Objective Function

The value (z) of the objective function, which is introduced in section 3, of a given solution set can be calculated with the formula below.

$$z = \begin{cases} \max\left(C_{max} - 2.778 * 10^{-4} \sum_{i=0}^{|N|-2} \sum_{j=1}^{|N|-1} e_{ij} x_{ij}, 0\right), & if\ L_i > 0, \quad \forall i \in N \setminus \{|N|-1\} \\ 0, & if\ L_i \leq 0, \quad \exists i \in N \setminus \{|N|-1\} \end{cases}$$

The objective function value calculation can be shown as an equation using cases. The first case states that, if the available energy at any node i, except the end node, is above 0, can be calculated with the function above. Since all calculations are based on Système International (SI) unit system, the second term of the formula is multiplied by a conversion factor, where the battery capacity is given as Watt-hour (Wh), to convert Joule (J) to Watt-hour (Wh), a conversion factor has been applied. Because of the objective function value, which has to represent the level of energy at the endpoint ($L_{|N|-1}$), cannot fall below 0, a maximum of function has been used. In the second case, where there is at least one node visited with an empty battery, the final available energy level is automatically forced to be 0, since the vehicle cannot accelerate again. The developed software is programmed to exit and give infeasibility if it is the case, instead of returning 0.

4.4 Upper Bound Calculation

Bound calculations are important steps to assess the quality of the outcome. They are used for estimating the distance between the found solutions' objective function value to the optimal solution set's objective function value. Since this problem defined as a maximization problem, an upper bound calculation is necessary in this case. Where the objective function value of found solution set denoted as z, objective function value of optimal solution set denoted as z^*, and the upper bound value denoted as $\bar{z}$, the relation between these values can be depicted as follows.

$$z \leq z^* \leq \bar{z}$$

Typically, bound calculations are done by relaxing a constraint or constraint set from the complete model, and resulting objective function value is assumed as an upper bound or lower bound for the problem. For this problem, an upper bound calculation is done by relaxing the constraints related to the load carried on arcs. With this logic, each customer's demand is assumed as 0, and the only load carried on arcs is assumed as equal to the mass of the vehicle (M_v), which is constant during the journey. Effect of total mass is included by both energy in and energy out situations, and because of the regeneration coefficient (α) is in a closed interval $[0,1]$, and a TSP solution set constructs a closed-loop tour, any additional load will lead to a negative effect on objective function value, which means that the proposed upper bound satisfies the above rule. When the carried additional load between node i and node j assumed to be negligible, the energy consumption or regeneration functions will be like the followings.

$$p_{ij}^{out'} = v(M_v g sin\theta_{ij}^{\geq 0} + 0.5 c_d \rho A v^2 + c_r M_v g cos\theta_{ij}^{\geq 0} + M_v \delta a^+)$$

$$p_{ij}^{in'} = \alpha v(M_v g sin\theta_{ij}^{-} + 0.5 c_d \rho A v^2 + c_r M_v g cos_{ij}^{-} + M_v \delta a^-)$$

$$e_{ij}' = \int_{out} p_{ij}^{out'} dt + \int_{in} p_{ij}^{in'} dt + \int p^{aux} dt$$

Then, a basic TSP model is optimally solved with a cost matrix contains possible energy consumptions or regenerations instead of the distances or traveling costs between nodes. Because of these cost matrices' asymmetric nature, they have to be symmetrized. To do these transformations, dummy nodes are introduced on all nodes in the set. For instance, node i is duplicated with name node i', and the energy consumption between these two nodes is set to $-M$, where M is a vast number and it actually results with an energy regeneration, which forces to traverse arc (i, i'). With the introduced dummy nodes, links from node i to node j can be represented with two different arcs such as arc (i, j) and arc (i', j) which are symmetric, and their energy consumptions are equal in the cost matrix. For more detailed expression of the method, please see the related literature (Jonker & Volgenant, 1983). The optimal objective value of the following integer programming model (IP), which is also known as DFJ formulation, is used as an upper bound for the main problem.

$$Minimize \sum_{i} \sum_{j} e'_{ij} x_{ij} \tag{1}$$

$subject\ to$:

$$\sum_{j, i \neq j} x_{ij} = 2 \quad \forall i \in N' \tag{2}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq N', |S| \geq 2 \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A' \tag{4}$$

The objective function (1) aims to minimize the total energy consumption. The constraint set (2) is known as degree-2 constraints, which ensures that the vehicle enters each node once and leave once. The constraint set (3) is the sub tour elimination constraints, which force the solution to return a single tour. The last constraint set (4) indicates the binary restrictions on the decision variables. At the end of the tests, found that upper bound values are close to the objective function values of best solutions obtained by metaheuristic applications.

4.5 Initial Solution Generation

As mentioned above in section 2, metaheuristics, or other words, improvement heuristics, are doing a local, global, or a combined search on the feasible region, starting from a chosen solution set. This solution set can be determined in a completely random fashion, or construction heuristics can be used to generate an initial solution. Due to the nature of many metaheuristics, there is a possibility to reach a local optimum, which may be difficult to escape and, in such situation, founding the global optimum is getting harder. In some cases, the result of construction heuristics can lead to a solution close to a difficult-to-escape local optimum, and the initial solution can affect the outcome. To avoid this possibility, five different initial solution generation methods are proposed. The first three methods can be classified as list-processing heuristics, which execute sorting mechanisms by using the properties affects the traveling costs. The fourth heuristic approach is called nearest neighbor algorithm, which is a well-known and widely used greedy heuristic. Besides them, the last and the fifth method constructs an initial solution by solving a basic TSP model considers the distances between the nodes on a given graph. Details of methods and algorithms are given in the following subsections.

4.5.1 Heuristic 1: Descending Loads

The first proposed list-processing heuristic is sorting the nodes according to a load of demands at nodes in descending order. First, the algorithm establishes a priority list, which is denoted as T in the following algorithm, that contains all nodes on the graph except depot locations. Then elements in T are sorted according to the nonincreasing order of customer demands (q_i), and the arcs between them are getting activated respectively. If all feasibility conditions are satisfied, such as the construction of a closed tour, the algorithm returns the solution set. The mathematical expression of the algorithm can be seen below.

Algorithm 1: Descending Loads Heuristic

Input: $G = (N, A);\ q_i, \quad \forall i \in N$

Output: $x^*_{ij}, \quad \forall (i,j) \in A$

Let list $T \subseteq N \setminus \{0, |N|-1\}$ indexed by $k = \{1, \ldots, |T|\}$

Sort the elements of T by nonincreasing value of q_i, if $q_j \geq q_i$ then q_j precedes q_i in T

Let $x^*_{ij} = 0, \quad \forall (i,j) \in A$

Define method $T(k)$, returns the value of the k^{th} element in T

Set $x^*_{0,T(1)} = 1$

Run through T, set $x^*_{T(k),T(k+1)} = 1\ , \ \ \forall k \in T$

Set $x^*_{T(|T|),|N|-1} = 1$

If the feasibility conditions are satisfied return $x^*_{ij}, \quad \forall (i,j) \in A$

4.5.2 Heuristic 2: Descending Road Grades

The second proposed list-processing heuristic is sorting the arcs according to their nonincreasing order of slopes. Similarly, the algorithm starts with the creation of a priority list (T) that contains all arcs on the defined graph. Arcs are sorted according to road grades (θ_{ij}) in nonincreasing order. Then the algorithm starts to activate arcs, under feasibility conditions from the beginning of priority list to the end. Once the activated arcs are constructed a closed-loop tour, which is not a sub tour of the given graph, the feasibility condition is getting satisfied. At the end, the algorithm returns a solution set that is a result of heuristic approximation. The expression of the algorithm as follows.

Algorithm 2: Descending Road Grades Heuristic

Input: $G = (N, A);\ \theta_{ij},\quad \forall (i,j) \in A$

Output: $x^*_{ij},\quad \forall (i,j) \in A$

1 Let list $T \subseteq A$ indexed by $k = \{1, \dots, |T|\}$

2 Sort the elements of T by nonincreasing value of θ_{ij}, if $\theta_{kl} \geq \theta_{ij}$, θ_{kl} precedes θ_{ij} in T

3 Let $x^*_{ij} = 0,\quad \forall (i,j) \in A$

4 Define method $T(k_1)$ and $T(k_2)$, return the first and second indices of the k^{th} arc in T

5 Run through T, set $x^*_{T(k_1),T(k_2)} = 1$,

if $\sum_j^{|N|-1} x_{T(k_1),j} = 0,\ \sum_i^{|N|-2} x_{i,T(k_2)} = 0,\quad \forall (k_1, k_2) \in T$

6 If the feasibility conditions are satisfied return $x^*_{ij},\quad \forall (i,j) \in A$

4.5.3 Heuristic 3: Ascending Road Grades

Ascending road grades heuristic has the same nature as the previous heuristic in section 4.5.2. Instead of sorting arcs in the priority list in nonincreasing order, the below algorithm sorts them in nondecreasing order. The expressions of steps are below.

Algorithm 3: Ascending Road Grades Heuristic

Input: $G = (N, A);\ \theta_{ij},\quad \forall (i,j) \in A$

Output: $x^*_{ij},\quad \forall (i,j) \in A$

1 Let list $T \subseteq A$ indexed by $k = \{1, \dots, |T|\}$

2 Sort the elements of T by nondecreasing value of θ_{ij}, if $\theta_{kl} \leq \theta_{ij}$, θ_{kl} precedes θ_{ij} in T

3 Let $x^*_{ij} = 0,\quad \forall (i,j) \in A$

4 Define method $T(k_1)$ and $T(k_2)$, return the first and second indices of the k^{th} arc in T

5 Run through T, set $x^*_{T(k_1),T(k_2)} = 1$,

if $\sum_j^{|N|-1} x_{T(k_1),j} = 0,\ \sum_i^{|N|-2} x_{i,T(k_2)} = 0,\quad \forall (k_1, k_2) \in T$

6 If the feasibility conditions are satisfied return $x^*_{ij},\quad \forall (i,j) \in A$

4.5.4 Heuristic 4: Nearest Neighbor

The nearest neighbor (NN) heuristic is a well-known greedy heuristic to solve TSP. It aims to come up with a good solution by starting from a randomly chosen location and visiting the closest unvisited node in each iteration. In this study, the starting point is indicated as a depot location, and the following algorithm finds the closest nodes in each iteration and activates the arc between them. In the end, the tour is closed by returning to the depot location. If all feasibility conditions are satisfied, the algorithm returns the found solution set. The mathematical expression of the algorithm can be found below.

Algorithm 4: Nearest Neighbor Heuristic

Input: $G = (N, A);\ d_{ij}, \quad \forall (i,j) \in A$

Output: $x^*_{ij}, \quad \forall (i,j) \in A$

1 Let $u = 0$;

2 Define method $F(u)$, returns the index of unvisited node s,
where $d_{us} = \min(d_{uj}, \ \forall j \in N \setminus |N| - 1)$

3 Set $x^*_{u,F(u)} = 1$

4 Set $u = s$, and repeat step 3 until $F(u) = \emptyset$

5 Set $x^*_{u,|N|-1} = 1$

6 If the feasibility conditions are satisfied return $x^*_{ij}, \quad \forall (i,j) \in A$

4.5.5 Optimal Tour on Distances

The last method to generate an initial solution used in this study is, solving the traveling salesman problem optimally according to the traveling cost, which is directly proportional to distances between nodes. Since it is an asymmetric TSP application, the cost matrices have to be symmetrized, as it is described in section 4.4. The DFJ formulation, which is used to find an optimal tour on distances is below.

The objective function (1) aims to minimize the total distance traveled. The constraint set (2) is degree-2 constraints, which imposes that the vehicle has to enter each node once and leave once. The constraint set (3) is the sub tour elimination constraints proposed in DFJ, which ensures that the solution constructs a single complete tour. Finally, the constraint set (4) indicates the binary restrictions on the decision variables. The optimal solution found is returned as the initial solution set to the program.

$$Minimize \sum_{i} \sum_{j} d_{ij} x_{ij} \tag{1}$$

$subject\ to$:

$$\sum_{j, i \neq j} x_{ij} = 2 \quad \forall i \in N \tag{2}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq N, |S| \geq 2 \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{4}$$

4.6 Neighborhood and Neighbors

Local search methods used in combinatorial optimization aim to find an optimal or near-optimal solution starting from a feasible initial solution, and moving to a close solution iteratively, among the feasible region. To perform local search algorithms, the ranges of these motions have to be well-structuralized, and the rule of movement has to be defined. The general neighborhood structure is defined as a collection of subsets of all possible solutions' set in the space. Where the neighborhood of a solution x, denoted as $N(x)$, this relation can be shown as $N(x) \subseteq X$, $x \in X$. In the traveling salesman problem, since a solution can be thought as a permutation of nodes and every permutation of the nodes corresponds to a feasible tour, the following neighborhood definition can be proposed.

$$N(x) = \{\bar{x} \mid \text{permutation } \bar{x} \text{ results from permutation } x \text{ by transposition of two nodes}\}$$

With the above neighborhood definition, some of the neighbors in the defined neighborhood of a given solution set $x = (i_1, i_2, i_3, i_4, i_5, i_6)$ will be as below.

$$\bar{x}_1 = (i_1, i_3, i_2, i_4, i_5, i_6)$$
$$\bar{x}_2 = (i_1, i_6, i_3, i_4, i_5, i_2)$$
$$\bar{x}_3 = (i_5, i_2, i_3, i_4, i_1, i_6)$$
$$\vdots$$

In this study, the above definition of the neighborhood is considered. During the run of local search algorithms, the neighbors are randomly drawn in each performed iteration from the neighborhood of current solution. For further researches, alternatively, the number of transposed nodes can be increased, or if we examine the tours as a list of arcs instead of a list of nodes as above, arcs activated between four nodes (i, j) and (k, l) can be exchanged to (i, k) and (j, l). This definition called the 2-exchange neighborhood concept (Crama, 2018).

4.7 Metaheuristic Applications

The ultimate aim of this study is obtaining near-optimal solutions for ETPS by using well-known metaheuristics, instead of solving the problem optimally in long durations. Under this subsection, the application of two local search methods is discussed. In subsection 4.7.1, the application of steepest ascent metaheuristic is described. In 4.7.2, the application of simulated annealing metaheuristic, which is comparably complicated than the steepest ascent, is described. Also, the idea behind parameter selection is provided.

4.7.1 Steepest Ascent

Steepest ascent metaheuristic is a very natural method for local searches. The idea behind steepest ascent is moving from the current solution to the best solution that can be found in its neighborhood until there is no further improvement. The algorithm starts with a selection of the initial solution x^1. Then, all neighbors in its neighborhood $N(x^1)$ are evaluated, and the neighbor with the highest objective function value $(\bar{x})$ is set as the current solution. This process is repeated until there is no better solution can be found in the current solution's neighborhood. In the end the algorithm returns the best solution found set x^*, and the associated objective function value F^*. The formal mathematical description of the algorithm is below.

Metaheuristic 1: Steepest Ascent

Input: x^1

Output: x^*, F^*

1 Select an initial solution $x^1 \in X$

2 Set the best objective function value $F^* = F(x^1)$, where $F(x)$ is the objective value of x

3 Set the best solution $x^* = x^1$, and number of iterations $k = 1$

Repeat:

4 | Find the best solution $\bar{x} \in N(x^k)$

5 | Set $F(\bar{x}) = \max\{F(x): x \in N(x^k)\}$

6 | If $F(\bar{x}) > F(x^k)$ set $x^{k+1} = \bar{x}$, $F^* = F(\bar{x})$, $x^* = \bar{x}$, and $k = k + 1$; Else go step 7

7 Return x^* and F^*

The trade-off with the steepest ascent metaheuristic is, it always searches for a local maximum, and in most of the cases it returns a local optimum, if this local optimum is not the global optimum at the same time; on the other hand, the required amount of time to run this metaheuristic is quite low, compared to others. In order to escape from local maximum, some additional actions can be taken; for example, an introduced method can replace the current solution with a randomly chosen neighbor instead of the neighbor with the highest objective function value at a random iteration of algorithm run. In this study, because of the simulated annealing metaheuristic has been implemented with an escape method, the steepest ascent has used as with its simple version. The results and the performance of steepest ascent application will be discussed later in section 5.

4.7.2 Simulated Annealing

Simulated annealing is a randomized algorithm, aiming to find the global optimum of combinatorial optimization problems. It is technically a local search method, however, allows to do searches in larger search spaces compared to steepest ascent metaheuristic by replacing a current solution with a randomly chosen solution from its neighborhood, which is a smart way to escape from the local maximum. Simulated annealing suggests doing these replacements more systematically rather than randomly proposing replacements at any iteration.

The idea behind simulated annealing comes from the annealing technique, which is used in material science and metallurgy as a result of thermodynamics. It is a heat treatment process to reduce materials hardness while increasing their ductility. In this process, first the materials are heated up to their recrystallization temperatures after they keep their heats for a reasonable amount of time, the materials are cooled slowly under a cooling schedule. The same logic is used in simulated annealing by introducing an imaginary starting temperature T_0, with a cooling rate α, where $\alpha \in (0,1)$. The simulated environment preserves its current temperature along L iterations, which is also the length of plateaus, before cooling it by α. A geometric cooling schedule is also visualized in Figure 5.

As with all local search methods, the simulated annealing algorithm starts with an initial solution, too. The algorithm is initiated with setting the best solution found and current solution to the initial solution, and the best solution found and current objective function values to the objective function value of the initial solution. Then the number of iterations, which denoted as k, and current temperature T are initiated. The initial temperature parameter T_0 is a user defined value.

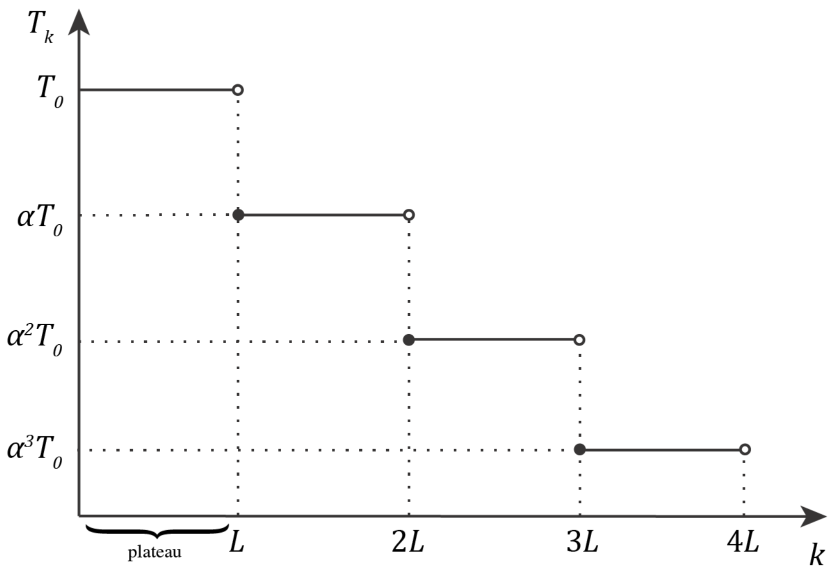

Figure 5: Geometric Cooling Schedule

After the initiation phase, the algorithm repeatedly draws a random neighbor from the current solution's neighborhood, compares its objective function values with the current one. If the objective function value of the neighbor is better than the current, the algorithm automatically updates the current solution with the neighbor. In the opposite case, a probabilistic method is used to generate a random number from a standard uniform distribution $U \sim [0,1]$, and compares it with a calculated threshold, which is called acceptance or transition probability. If the generated random number is less than a calculated transition probability, the current solution is replaced with the neighbor. This process is repeated until stopping conditions become satisfied, which will be discussed below. If stopping conditions are satisfied, the algorithm returns the best solution found set x^* and the associated objective function value F^*. The formal mathematical expression of the main body of simulated annealing metaheuristic is Metaheuristic 2, which is below.

As mentioned in line 5, in case of the chosen neighbor's objective function value is better than the current's, the algorithm calls the AcceptMove($\bar{x}$) method, which is formally described in Method 1.

Metaheuristic 2: Simulated Annealing

Input: $x^1, T_0, \alpha, L, K, \varepsilon, t_{max}$

Output: x^*, F^*

1 Select an initial solution $x^1 \in X$

2 Set the best objective function value $F^* = F(x^1)$, where $F(x)$ is the objective value of x

3 Set the best solution $x^* = x^1$, number of iterations $k = 1$, and current temperature $T = T_0$

Repeat:

4 | Draw $\bar{x}$ randomly from $N(x^k)$

5 | If $F(\bar{x}) > F(x^k)$ go method AcceptMove($\bar{x}$); Else go method Toss$(x^k, \bar{x})$

6 | Go method CheckStoppingConditions()

7 | If $terminate = true$, go step 8

8 Return x^* and F^*

Method 1: AcceptMove($\bar{x}$)

1 Set $x^{k+1} = \bar{x}$

2 If $F(\bar{x}) > F(x^*)$, set $F^* = F(\bar{x})$, and $x^* = \bar{x}$

On the same line, on the contrary situation algorithm calls Toss $(x^k, \bar{x})$ method. This method first calculates the difference between objective function values of the current solution set, and the candidate solution set. Accordingly, an acceptance probability p_k is calculated with the below formula.

$$p_k = e^{-\frac{\Delta F}{T}}$$

Since the computation time of the above formulation is quite high in complexity terms, an alternative and simpler formulation can also be used to calculate p_k, which is following.

$$p_k = 1 - \frac{\Delta F}{T}$$

The above formulation is indicated as 25 times faster than the standard formulation in the literature (Oliveira & Ferreira, 1993). However, considering the capability of used hardware and the interpreter, and the observed solution quality of results, the first formulation is preferred to calculate transition probabilities. The mathematical expression of the method is below.

Method 2: Toss$(x^k, \bar{x})$	
1	Calculate $\Delta F = F(\bar{x}) - F(x^k)$
2	Calculate transition probability $p_k = e^{-\frac{\Delta F}{T}}$
3	Draw a random number u, from a standard uniform distribution $U \sim [0,1]$
4	If $u \leq p_k$ set $x^{k+1} = \bar{x}$; Else continue;

Following the update decision of the current solution, in each iteration, the stopping conditions are getting checked. There are many approaches to decide terminating simulated annealing algorithm or not. The most basic stopping criterion is comparing the elapsed time with a user-defined time limit. Since the simulated annealing algorithm does its searching in a greater space, depending on also other stopping criteria, termination can take a quite long time. Another stopping criterion is checking the number of improvements on F^*, and the percentage of accepted moves during the last K plateaus, where K is a user-defined threshold. If there is no recorded improvement, and if fewer than $\varepsilon\%$ of moves have been accepted, where ε is another user-defined parameter, during the last K successively performed plateaus, stopping condition is assumed to be satisfied.

In the same method, the decision of cooling is also given. The condition of the cooling environment is if the algorithm will not be terminated at that point, and if the number of performed iterations since the beginning of plateau is greater than or equal to the plateau length L, the temperature will be increased by cooling factor α. The formal explanation of the method can be found in Method 3.

Another important of simulated annealing application is parameter selection. Unlike the steepest ascent metaheuristic, the simulated annealing approach requires several user-defined parameters, which have a direct effect on the outcome. As mentioned in the above explanation, the required inputs are an initial temperature (T_0), a cooling factor (α), a plateau length (L), a number of last plateaus getting checked (K), a percentage threshold value for checking the number of accepted moves in last K plateaus. And additionally, a user-defined time limit ($t_{\max}$) can be introduced, which is optional.

Method 3: CheckStoppingConditions()

If the elapsed time is greater than or equal to the user defined time limit t_{max}

Set $terminate = true$

If the number of iterations since the last temperature decreasing is less than L:

Set $k = k + 1$

Set $terminate = false$

If there is no improvement on F^* $\wedge$ less than $\varepsilon\%$ moves have been accepted in last K plateau

Set $terminate = true$

Else:

Set $T = \alpha T, k = k + 1$

Set $terminate = false$

There is not an optimal method for choosing these parameters: it is usually done empirically by trial-and-error adjustments. However, few ideas are introduced for some of the parameters, which experimentally provide good results. One of them is for deciding on T_0. Previous researches and textbooks advice to calculate T_0, with the following formula, where δ is an expected improvement amount in objective function value, and p_0 is the initial acceptance probability.

$$T_0 = -\frac{\delta}{\ln{(p_0)}}$$

If p_0 is set to 0.5, which is widely recommended in the literature, the above formula becomes $T_0 \approx 1.45 * \delta$. Because of this problem is dealing with the maximization of available energy at the end of the tour, and the unit of the result is in Watt-hour (Wh), an improvement of $1000\ Wh$ ($= 1\ kWh$), which is a reasonable amount for electric vehicles, can be expected on average. In this case, if δ is set to 1000, the T_0 value becomes 1450. Another recommendation in literature is for the plateau length L. The rule of choosing L value stated as this value has to be larger than the estimated size of a neighborhood to give a chance of being created to each neighbor. The usual choice is calculated as follows.

$$L \approx 10 \times |N(x)|$$

Since one of the aims of this study is providing a well-working common parameter set for simulated annealing application, instance size-independent parameters were looked for. To achieve this, for all instances the neighborhood sizes $|N(x)|$ are examined. Where the number of nodes in an instance denoted as n, the calculation can be shown as $|N(x)| = C(n, 2)$. This study is done on instances with $7, 11$ and 15 nodes, whose neighborhood sizes are $21, 55$ and 105, respectively. While setting the value of L, values from the interval $[210,1050]$ is tested, rather than using the number of neighbors from a single instance.

Other parameters are chosen from reasonable intervals with the trial-and-errors approach. α is chosen from the interval $[0.8, 1)$, K is chosen from the interval $[20, 60]$, ε is chosen from the interval $(0, 0.05]$. The user-defined time limit is set to $30.000\ ms$ $(= 5\ min)$. The final parameter set is as follows.

T_0	1,450
α	0.998
L	300
K	50
ε	0.01
t_{max}	$30{,}000\ ms$

Table 3: Simulated Annealing Parameter Set

The results and the performance of simulated annealing application will be discussed later in section 5.

5 Results and Analyzes

Under this section, the outputs of the above implementation are handled and analyzed from different angles. In subsection 5.1, the usages of different arc types, which were introduced in 4.2.4, were analyzed. In 5.2, calculated upper bounds and associated solution sets were given. Under subsection 5.3, performances of the introduced greedy heuristics and optimal TSP solution on distances, which were used to generate initial solutions, were analyzed. In 5.4 and 5.5, the obtained results by using metaheuristics were presented, and comments were shared about. Lastly, in 5.6, a comparison made between the classical TSP approach, which is mostly aiming to minimize the total distance traveled, and the ETSP application.

5.1 Arc Type Evaluation

As will be remembered, in subsection 4.2.4, three different types of arcs are proposed to solve this problem on real-life data. Since the information carried on arc contain mathematical components of energy consumption or energy regeneration calculation, its importance for ETSP application has to be tested. To emphasize the effect of arc types, both steepest ascent and simulated annealing applications are made on Instance 1 with mentioned three types of arcs. Where x^* indicates the best tour found and F^* indicates the objective function value, which is the available energy level at the end of the tour, results are shared in Table 4 and Table 5.

Arc Type	$\boldsymbol{x^*}$	$\boldsymbol{F^*}$ $(\boldsymbol{Wh})$	**Tour Length** $(\boldsymbol{m})$
Simple Arcs	D-1-6-2-3-5-4-D	21542.25	32181.85
High-Resolution Arcs (D = 10)	D-6-1-2-5-3-4-D	18889.56	31931.88
Real-Time Arcs	D-2-3-4-5-6-1-D	17107.80	46127.00

Table 4: Instance 1 - Steepest Ascent Results on Arc Types

Arc Type	$\boldsymbol{x^*}$	$\boldsymbol{F^*}$ $(\boldsymbol{Wh})$	**Tour Length** $(\boldsymbol{m})$
Simple Arcs	D-1-6-2-3-5-4-D	21542.25	32181.85
High-Resolution Arcs (D = 10)	D-6-1-2-5-3-4-D	18889.56	31931.88
Real-Time Arcs	D-2-3-4-5-6-1-D	17107.80	46127.00

Table 5: Instance 1 - Simulated Annealing Results on Arc Types

As can be clearly seen above, the objective function values and the total distances traveled differ significantly. Also, with these three different arc types, the resulting solution set, which is the most vital component of the solution since it is also the recommended action to be taken, are not the same. Despite arc construction is a complicated and time-consuming task for more complex arcs such as real-time ones, results show us the spent time is worth it. In order to provide more precise insights and results, the rest of the analyses are done with real-time arcs, in the following subsections.

5.2 Upper Bounds

As the upper bound calculation method was detailed in subsection 4.4, after relaxing customer demand related constraints, an integer programming model solved optimally for all instances on expected energy consumptions or regenerations on arcs. The objective function values of these infeasible solutions are assumed as upper bound values which can be found in the below table.

Instance	$UB\ (Wh)$	x^{UB}	$F^{UB}\ (Wh)$	$F^{UB}/_{UB}$
Instance 1	20943.89	D-2-3-4-5-6-1-D	17107.80	82%
Instance 2	19666.47	D-1-2-3-4-5-6-D	12413.96	63%
Instance 3	19202.53	D-2-1-4-5-7-6-8-9-10-3-D	11256.54	59%
Instance 4	18371.74	D-13-10-8-4-12-5-11-14-6-1-9-2-3-7-D	10370.83	56%

Table 6: Calculated Upper Bounds and Associated Solutions

Upper bounds are used to assess the final solution quality. If the convergence of upper bound values to the optimal objective function values are getting higher, the quality assessments can be done more precisely. According to the results obtained in the following subsections, the described method is a natural and proper way to calculate it; however, it is evident that the method is working better on smaller instances and instances with less customer demands.

n addition to upper bound calculation, the associated solution sets' real objective function values are calculated to check if they provide comparably good solutions. While this approach is providing the best solution found, for Instance 1, for other instances, better results were obtained. When the results are evaluated with the related instances side-by-side, the correlation between the results and standard deviation of loads is clearly visible. In general, it can be said that in case of customers' demands are not deviate a lot from each other; the same method has a possibility to provide optimal or near-optimal solution sets for the whole problem.

5.3 Performance Analysis of Solution Generation Methods

As discussed earlier, in subsection 4.5, five different initial solution generation methods were proposed. While the first four of them are construction heuristics, the latter method is introduced to solve the problem optimally with a classical TSP point of view. This idea allows us to compare a classical TSP approach, which is based on distance minimization, and its extension for the electric vehicles; while still generating considerably good solutions. All of these five methods can also be used to compute fast and good solutions without running an additional search algorithm on them. Therefore, their performances need to be assessed for future uses. The results coming from solution generation methods are provided for all instances in Table 7, Table 8, Table 9, and Table 10, respectively.

Sol. Generation Methods	$\boldsymbol{x^1}$	$\boldsymbol{F^1\ (Wh)}$	**Tour Length** $(\boldsymbol{m})$	$\boldsymbol{F^1/_{UB}}$
Descending Loads	D-5-1-3-2-4-6-D	13942.64	59843.00	67%
Ascending Road Grades	D-6-1-4-5-3-2-D	15079.68	54486.00	72%
Descending Road Grades	D-2-3-5-4-1-6-D	15236.87	58424.00	73%
Nearest Neighbor (NN)	D-1-5-4-6-2-3-D	15304.99	53168.00	73%
Optimal Tour on Distances	D-5-4-3-2-6-1-D	16365.95	45664.00	78%

Table 7: Instance 1 - Results of Initial Solution Generation Methods

The results of Instance 1, shows that the optimal tour on distances has provided the best result among all methods. The nearest candidate is nearest neighbor heuristic, which is a widely used alternative of optimally solving TSP instances. Another key point is, there is a moderate correlation between the tour lengths and objective function values. Its occurrence is natural since the energy consumption is directly depended on the integrations on time, while speed is not changing during the journey.

Sol. Generation Methods	x^1	$F^1\ (Wh)$	Tour Length (m)	$F^1/_{UB}$
Descending Loads	D-5-1-2-3-4-6-D	10619.63	89985.00	54%
Ascending Road Grades	D-3-5-4-1-2-6-D	9183.56	86065.00	71%
Descending Road Grades	D-6-2-1-4-5-3-D	7771.07	84589.00	40%
Nearest Neighbor (NN)	D-4-5-6-1-2-3-D	12898.55	65172.00	66%
Optimal Tour on Distances	D-6-5-4-3-2-1-D	11657.62	62498.00	59%

Table 8: Instance 2 - Results of Initial Solution Generation Methods

Instance 2's results tell that the elevation differences between nodes are high in average, since the difference between heuristic results coming from ascending road grades and descending road grades, which are exact opposite algorithms, is significant. There is also an effect of delivering the heaviest load at the earlier stages of the tour. On the other side the best solution still delivered by nearest neighbor heuristic then optimal tour on distances.

Sol. Generation Methods	x^1	$F^1\ (Wh)$	Tour Length (m)	$F^1/_{UB}$
Descending Loads	D-1-2-3-4-5-6-7-8-9-10-D	7178.14	87355.00	37%
Ascending Road Grades	D-7-9-5-3-8-6-1-2-4-10-D	4544.75	107544.00	24%
Descending Road Grades	D-10-4-2-1-6-8-3-5-9-7-D	4336.25	110363.00	23%
Nearest Neighbor (NN)	D-3-5-1-2-4-8-6-7-9-10-D	9222.41	78852.00	48%
Optimal Tour on Distances	D-3-1-2-4-5-7-6-8-9-10-D	11185.75	61952.00	58%

Table 9: Instance 3 - Results of Initial Solution Generation Methods

In Instance 3's results, the best objective function values are coming from the optimal tour on distances, and nearest neighbor heuristic, respectively. In spite of some of the nodes are same in Instance 2 and Instance 3, the same situation about road grades heuristics is not observed. The possible reason behind is, an increase in the number of customer nodes result up with more breakpoints that may decrease the deviation of arc slopes.

Sol. Generation Methods	x^1	$F^1\ (Wh)$	**Tour Length** (m)	$F^1/_{UB}$
Descending Loads	D-6-10-5-9-12-8-11-4-13-3-1-7-2-14-D	$Inf.$	175343.00	0%
Ascending Road Grades	D-9-5-2-7-8-14-11-10-12-13-6-1-3-4-D	$Inf.$	143773.00	0%
Descending Road Grades	D-4-3-1-6-13-12-10-11-14-8-7-2-5-9-D	$Inf.$	137808.00	0%
Nearest Neighbor (NN)	D-13-5-14-1-6-11-7-9-2-3-12-4-10-8-D	8522.18	78254.00	46%
Optimal Tour on Distances	D-8-10-4-12-3-7-2-9-1-6-11-14-5-13-D	10011.78	71779.00	54%

Table 10: Instance 4 - Results of Initial Solution Generation Methods

And, the last instance's results table tells another story. During the algorithm runs on Instance 4, many candidate solution sets had an infeasibility problem. Since a load of demands are high on average compared to the other instances, and the graph is much more complicated, the battery can become an empty situation in the earlier stages of a tour. The first three heuristic methods, which do not consider the length of the tour, has generated infeasible results under the rules of ETSP. Again, the optimal tour on distances has returned a good result by minimizing total tour length, and similarly nearest neighbor has provided a good solution, too.

On average, the specific heuristics proposed to generate initial solutions are seem like not as successful as the well-known nearest-neighbor heuristic and the exact method. On the other hand, due to the nature of search algorithms, in some cases, worse initial solutions can escape from local optimums and reach to better local optimums or global optimums easier. Their impact on final solutions will be discussed in 5.4 and 5.5.

5.4 Analysis of Steepest Ascent Metaheuristic Application

Steepest ascent (or for minimization problems steepest descent) is a very simple and natural metaheuristic. The idea behind steepest ascent is simply keeping moving to a better solution in the current neighborhood. It often runs very fast in the mean of computation; however, the main trade-off about steepest ascent is that there is no introduced escape procedure in its basic algorithm. In order to escape from local maximums or minimums, a few escape procedures can be introduced as discussed in 4.7.1; however, in this study, the basic version is considered. The final results obtained by applying steepest ascent metaheuristic can be found in Table 11, Table 12, Table 13, and Table 14 for all instances, respectively.

Sol. Generation Methods	$\boldsymbol{x}^*$	$\boldsymbol{F}^*$ $(\boldsymbol{Wh})$	**T. Length** $(\boldsymbol{m})$	$\boldsymbol{F}^*/_{\boldsymbol{UB}}$	$\boldsymbol{t}$ $(\boldsymbol{ms})$
Descending Loads	D-1-5-3-4-6-2-D	16659.20	50251.00	80%	4022
Ascending Road Grades	D-1-5-3-4-6-2-D	16659.20	50251.00	80%	4402
Descending Road Grades	D-2-3-4-5-6-1-D	17107.80	46127.00	82%	3879
Nearest Neighbor (NN)	D-1-5-6-2-3-4-D	16821.11	46185.00	80%	3841
Optimal Tour on Distances	D-5-4-3-2-6-1-D	16365.95	45664.00	78%	120

Table 11: Instance 1 - Steepest Ascent Metaheuristic Results

The steepest ascent results obtained on Instance 1, says that the best objective function value is found by moving in descending road grades heuristic generated solution's neighborhood. If the best solution found is compared to the initial solution generated by this algorithm, it is possible to see this prominent similarity, which means that these two solutions are quite close in the solution space. Besides, steepest ascent cannot find a better solution in the optimal tour on distances' neighborhood. It shows that the optimal tour is a solution at a local maximum in the space.

Sol. Generation Methods	x^*	$F^*\ (Wh)$	T. Length (m)	$F^*/_{UB}$	$t\ (ms)$
Descending Loads	D-1-5-2-3-4-6-D	12944.80	99193.00	66%	3833
Ascending Road Grades	D-1-5-2-3-4-6-D	12944.80	99193.00	66%	3433
Descending Road Grades	D-1-2-5-6-4-3-D	14032.33	76750.00	71%	4046
Nearest Neighbor (NN)	D-4-5-6-1-2-3-D	12898.55	65172.00	66%	4308
Optimal Tour on Distances	D-1-5-2-3-4-6-D	12944.80	99193.00	66%	4295

Table 12: Instance 2 - Steepest Ascent Metaheuristic Results

In the above table, the best objective function value is obtained by starting from the initial solution of descending road grades again. However, for Instance 2, descending road grades heuristic has constructed a tour with the worst objective function among others. In this scenario, the local search mechanism could find the best solution in the neighbor of a low-quality solution. It is an example of the idea presented at the end of 5.3.

Sol. Generation Methods	x^*	$F^*\ (Wh)$	T. Length (m)	$F^*/_{UB}$	$t\ (ms)$
Descending Loads	D-2-1-4-5-7-6-8-9-10-3-D	11256.54	65070.00	59%	3572
Ascending Road Grades	D-2-1-3-6-8-7-5-4-9-10-D	11398.99	86703.00	59%	4751
Descending Road Grades	D-2-1-9-10-8-6-3-5-4-7-D	11325.12	94441.00	59%	3620
Nearest Neighbor (NN)	D-3-5-1-2-4-7-6-8-9-10-D	10962.90	69309.00	57%	3805
Optimal Tour on Distances	D-3-1-2-4-5-7-6-8-9-10-D	11185.75	61952.00	58%	3812

Table 13: Instance 3 - Steepest Ascent Metaheuristic Results

A similar example can be given on the application on Instance 3. As can be seen in 5.3, despite the worst couple of solutions were descending road grades and ascending road grades, respectively, both steepest ascent results show that the improvements could be made easier in their neighborhoods.

Sol. Generation Methods	$\boldsymbol{x}^*$	$\boldsymbol{F}^*$ $(\boldsymbol{Wh})$	**T. Length** $(\boldsymbol{m})$	$\boldsymbol{F}^*/_{\boldsymbol{UB}}$	$\boldsymbol{t}$ $(\boldsymbol{ms})$
Descending Loads	D-6-10-5-9-12-8-11-4-13-3-1-7-2-14-D	$inf.$	175343.00	0%	320
Ascending Road Grades	D-13-5-11-6-10-8-7-14-1-4-12-3-2-9-D	9627.92	111473.00	52%	3668
Descending Road Grades	D-4-12-14-6-10-5-11-1-8-7-3-2-9-13-D	6668.10	138208.00	36%	4521
Nearest Neighbor (NN)	D-13-5-11-6-1-14-7-3-2-9-12-4-10-8-D	9464.94	82416.00	52%	3981
Optimal Tour on Distances	D-8-10-4-12-7-3-2-9-1-6-11-14-5-13-D	10354.75	72126.00	56%	3626

Table 14: Instance 4 - Steepest Ascent Metaheuristic Results

And lastly, for Instance 4, all initial solution except the solution coming from descending loads heuristic, are improved, and the best solution could be obtained with the solution set of the optimal tour on distances. The first three algorithms have returned infeasible solution sets at the beginning; however, in the neighbor of road grade related heuristics, good solutions could be detected.

An exciting outcome of steepest ascent metaheuristic application is, among these four different instances, the initial solutions coming from the road grade related heuristics were improved rapidly to quite good solutions. Optimal tours on distances are followed them in three applications, and in the last application, it returned the best solution found with the steepest ascent.

5.5 Analysis of Simulated Annealing Metaheuristic Application

Simulated annealing metaheuristic is another local search method like the steepest ascent; however, it is a more complicated algorithm, which follows a randomized procedure. The details of the algorithm, its auxiliary methods, and user-defined parameters were introduced in 4.7.2, and all of the following results, which can be found in Table 15, Table 16, Table 17 and Table 18, are generated by using same parameters. The only parameter change has been done for the last instance, to test the algorithm within a wider time interval, whose results are in Table 19. Since simulated annealing is a probabilistic method, in each run, it is natural to see different outcomes. In light of this fact, each of the following tests is done five times, and the best solution founds were recorded among all.

Sol. Generation Methods	x^*	F^* (Wh)	**T. Length** (m)	$F^*/_{UB}$	t (ms)
Descending Loads	D-2-3-4-5-6-1-D	17107.79	46127.00	82%	30000
Ascending Road Grades	D-2-3-4-5-6-1-D	17107.79	46127.00	82%	30000
Descending Road Grades	D-2-3-4-5-6-1-D	17107.79	46127.00	82%	30000
Nearest Neighbor (NN)	D-2-3-4-5-6-1-D	17107.79	46127.00	82%	30000
Optimal Tour on Distances	D-2-3-4-5-6-1-D	17107.79	46127.00	82%	30000

Table 15: Instance 1 - Simulated Annealing Metaheuristic Results

For the very first instance, all solutions were converged to the same solution set, which is probably the global optimum. It can be expected because of the instance have few nodes, which defines a less complicated computation environment. Each run was terminated because of the violation of the time limit, after running 30 seconds. The objective function value to upper bound ratio is not as high as expected; however, it is still a good ratio for a problem that has such complexity. An XY chart can be seen in Figure 6Figure 2 that presents the change in objective function value according to the number of iterations performed.

As can be seen in Table 16 the same situation applies to it, too. It is a test done on the same size instance, and the solutions resulted from all applications has converged to the same solution set, which is obviously optimal. This time, the objective function value to upper bound ratio is lower, as a result of a lot heavier total load compared to the first instance. All tests took

30 seconds, without an earlier termination due to the stopping criteria defined in 4.7.2. Another XY chart associated with Instance 2 can be seen in Figure 7.

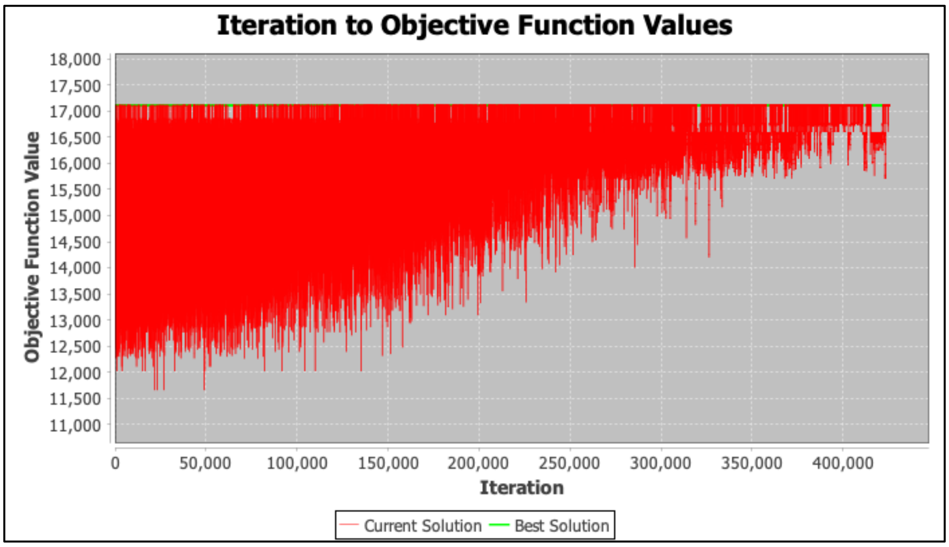

Figure 6: Instance 1 - Simulated Annealing Iteration to Objective Function Value Chart

Sol. Generation Methods	x^*	F^* (Wh)	**T. Length** (m)	$F^*/_{UB}$	t (ms)
Descending Loads	D-1-2-5-6-4-3-D	14032.33	76750.00	71%	30000
Ascending Road Grades	D-1-2-5-6-4-3-D	14032.33	76750.00	71%	30000
Descending Road Grades	D-1-2-5-6-4-3-D	14032.33	76750.00	71%	30000
Nearest Neighbor (NN)	D-1-2-5-6-4-3-D	14032.33	76750.00	71%	30000
Optimal Tour on Distances	D-1-2-5-6-4-3-D	14032.33	76750.00	71%	30000

Table 16: Instance 2 - Simulated Annealing Metaheuristic Results

At the end of the application on Instance 3, which can be seen in Table 17, all methods were again converged to the same optimal-likely solution. However, this time in two cases with descending loads and nearest neighbor beginnings, the algorithm is terminated before the introduced time limit, because no improvement could be recorded in last $K * L$ iterations, which corresponds to 15,000 iterations. In Figure 8, following the finding of the best solution, it is seen that the algorithm moved away from the best solution found.

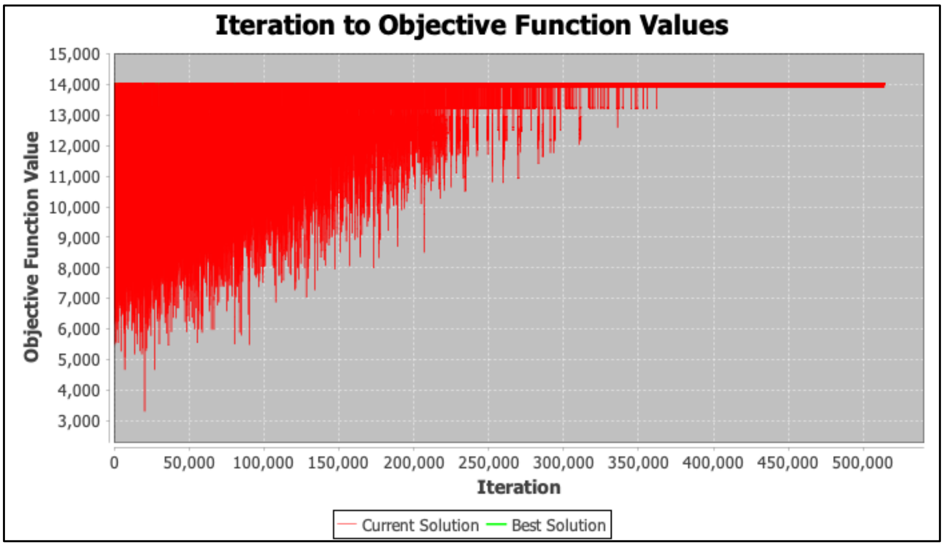

Figure 7: Instance 2 - Simulated Annealing Iteration to Objective Function Value Chart

Sol. Generation Methods	x^*	F^* (Wh)	**T. Length** (m)	F^*/UB	t (ms)
Descending Loads	D-2-1-4-9-10-3-5-7-8-6-D	12250.29	87461.00	64%	29367
Ascending Road Grades	D-2-1-4-9-10-3-5-7-8-6-D	12250.29	87461.00	64%	30000
Descending Road Grades	D-2-1-4-9-10-3-5-7-8-6-D	12250.29	87461.00	64%	30000
Nearest Neighbor (NN)	D-2-1-4-9-10-3-5-7-8-6-D	12250.29	87461.00	64%	29680
Optimal Tour on Distances	D-2-1-4-9-10-3-5-7-8-6-D	12250.29	87461.00	64%	30000

Table 17: Instance 3 - Simulated Annealing Metaheuristic Results

In Table 18 and Table 19, the results of the application on Instance 4 are given. Since Instance 4 is the largest instance among all instances, and due to the combinatorial difficulties appears when the size increases, solving the problem is getting harder. As can be seen in the first table, except descending road grades and optimal tour on distances initiations, all methods resulted up with different solution sets. Additionally, the XY chart in Figure 9 shows that there was a continuous improvement trend up to the end of the algorithm run. To investigate this situation and obtain a better solution, the time limit is increased to a minute, and results are tabulated again in the second table.

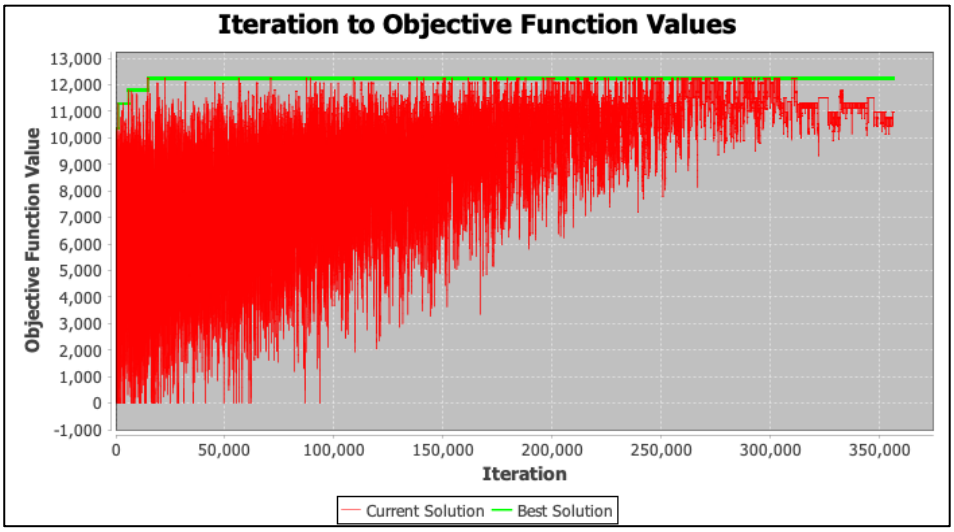

Figure 8: Instance 3 - Simulated Annealing Iteration to Objective Function Value Chart

Sol. Generation Methods	x^*	F^* (Wh)	**T. Length** (m)	$F^*/_{UB}$	t (ms)
Descending Loads	D-13-10-8-4-12-7-3-2-9-1-6-11-14-5-D	10549.80	77000.00	57.42%	30000
Ascending Road Grades	D-13-10-8-4-12-7-9-2-3-14-11-6-1-5-D	10414.55	85331.00	56.69%	30000
Descending Road Grades	D-13-10-8-7-14-11-6-1-4-12-3-2-9-5-D	10498.08	98459.00	57.14%	30000
Nearest Neighbor (NN)	D-13-10-8-4-12-14-11-6-1-9-2-3-7-5-D	10382.44	85359.00	56.51%	30000
Optimal Tour on Distances	D-13-10-8-7-14-11-6-1-4-12-3-2-9-5-D	10498.08	98459.00	57.14%	30000

Table 18: Instance 4 - Simulated Annealing Metaheuristic Results (t_{max} = 30 s)

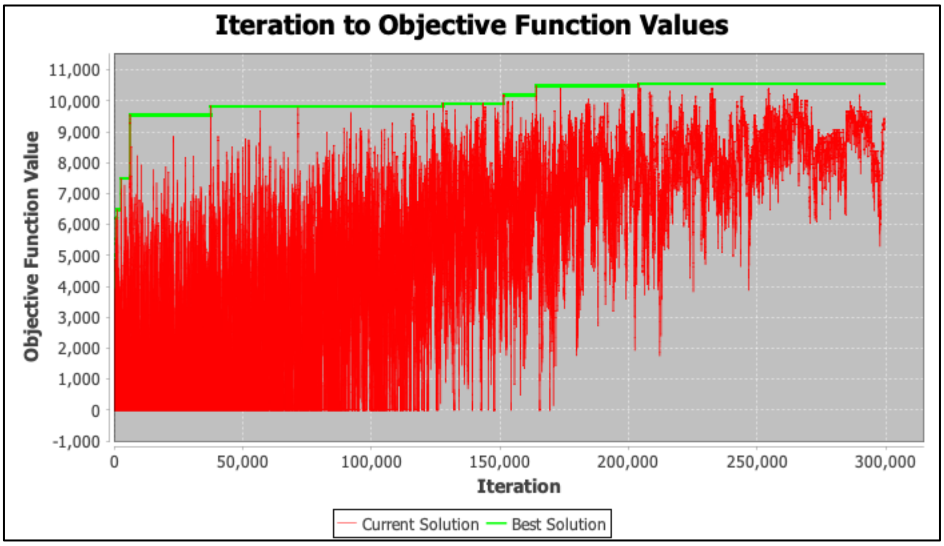

Figure 9: Instance 4 – S. Annealing Iteration to Objective F. Value Chart (t_{max} = 30 s)

Sol. Generation Methods	x^*	F^* (Wh)	**T. Length** (m)	F^*/UB	t (ms)
Descending Loads	D-13-10-8-4-12-7-3-2-9-1-6-11-14-5-D	10549.80	77000.00	57.4%	35980
Ascending Road Grades	D-13-10-8-4-12-7-3-2-9-1-6-11-14-5-D	10549.80	77000.00	57.4%	40582
Descending Road Grades	D-13-10-8-7-14-11-6-1-4-12-3-2-9-5-D	10498.08	98459.00	57.1%	32620
Nearest Neighbor (NN)	D-13-10-8-7-14-11-6-1-4-12-3-2-9-5-D	10498.08	98459.00	57.14%	36161
Optimal Tour on Distances	D-13-10-8-7-14-11-6-1-4-12-3-2-9-5-D	10498.08	98459.00	57.14%	37593

Table 19: Instance 4 - Simulated Annealing Metaheuristic Results (t_{max} = 1 min)

The results in Table 19 shows that this time, all tests were terminated earlier as a result of stopping criteria. The best solution found has not changed; however, significant improvements have been recorded on other tests. If the XY chart of objective function value change to the number of iterations for the best solution found is examined in Figure 10, it can be noticed that after obtaining the best solution, the search moved away.

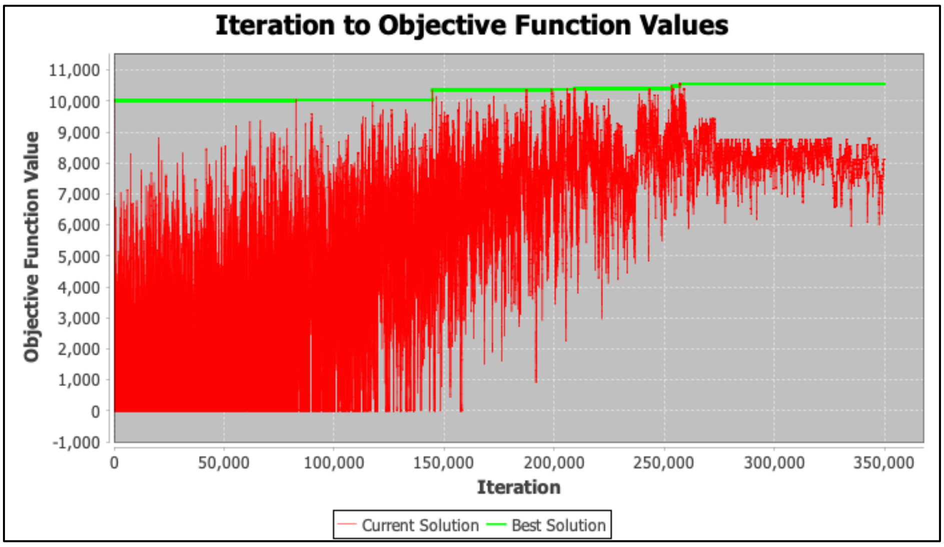

Figure 10: Instance 4 – S. Annealing Iteration to Objective F. Value Chart (t_{max} = 1 min)

Based on the all tests performed above, it can be said that the used parameter set introduced in 4.7.2 is worked well for associated instances, and within this instance size interval the same parameter set can be used in other similar studies. As a remark, to lengthen or shorten the run time, adjusting the cooling factor α has resulted in a better solution quality, instead of changing plateau length L. Another adjustment can be made by making changes on the parameter K, however, reducing it a lot or contrarily increasing the value causes an earlier termination or long computation times. While increasing the value of K, the time limit has to be adjusted accordingly.

5.6 Comparison of TSP Results and ETSP Results

Electric vehicle traveling salesman problem (ETSP) is an extension of the classical traveling salesman problem, which based on distances or traveling costs mainly. Basically, both problems seek to construct a closed tour visited given number of locations, under feasibility conditions. Classical TSP model is usually not considering the fuel level of a vehicle. The reasoning behind it is, driving range of petroleum-based vehicles are high, gas stations are quite common, and fuel acquisition times are negligible. On the other side, these conditions are not applying to electric vehicles. Depending on the size of the battery, driving ranges are considerably short, charge stations are still not common, and charging durations are quite long. When this is the case, ETSP needs to sustain the vehicles' battery levels in order to accomplish an uninterrupted tour. As it was repeatedly discussed above, the level of available energy can be sustained by doing energy regeneration and adjusting vehicle routes accordingly. Under this subsection, the differences of thought of the two problems having the same basis will be examined numerically and visually. For all comparisons, the optimal TSP solutions based on distances, and the best solutions found with metaheuristics for ETSP are considered. The locations and the routes shown on maps represent the exact locations in instances and the real routes between locations, which were collected during the arc creation process. Additionally, battery level information at each node visit calculated with the developed software, which is a component of this study.

The first comparison is made between the optimal tour on distance and the possible optimal solution of ETSP, for Instance 1. The results are shown in Table 20.

Problem	x^*	F^* (Wh)	**Total Length** (m)	F^*/UB
TSP	D-5-4-3-2-6-1-D	16365.95	45664.00	78%
ETSP	D-2-3-4-5-6-1-D	17107.80	46127.00	82%

Table 20: Instance 1 - TSP & ETSP Results Comparison

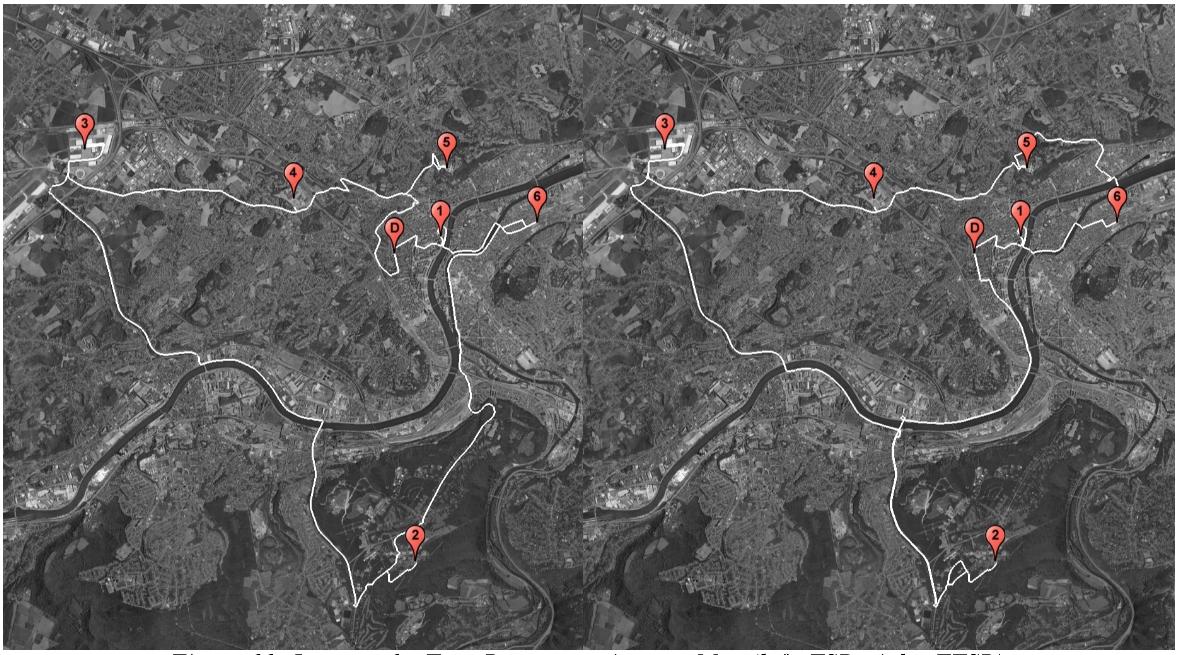

Figure 11: Instance 1 - Tour Representations on Map (left: TSP, right: ETSP)

Both tours were represented on a map in Figure 11. In the best ETSP tour, the vehicle first visits the location with the highest altitude with its all load. Contrarily, the optimal TSP tour visits the same node in the fourth-order. If the vehicle's battery level changes, which are presented in Figure 12, are examined it can be clearly seen that the decrease in battery level while climbing up to the location of customer 2, realized a bit costlier for TSP solution. At the end of the tour, despite the total travel distance is slightly higher, the best solution found of metaheuristic application can provide a higher level of available energy.

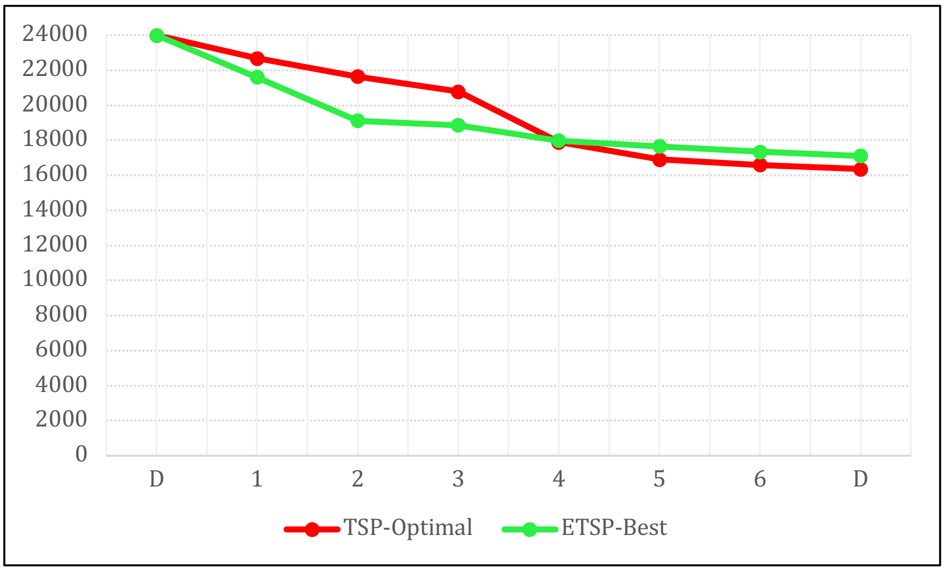

Figure 12: Instance 1 - TSP & ETSP Battery Levels Comparison

In Table 21 and Figure 13, the difference is more apparent for Instance 2. The tour for ETSP is significantly longer than the optimal tour length, while the differences in battery levels are much higher than the TSP result. Also, as it can be seen in Figure 14, battery levels at each visited location are better.

Problem	x^*	F^* (Wh)	**Total Length** (m)	$F^*/_{UB}$
TSP	D-6-5-4-3-2-1-D	11657.62	62498.00	59%
ETSP	D-1-2-5-6-4-3-D	14032.33	76750.00	71%

Table 21: Instance 2 - TSP & ETSP Results Comparison

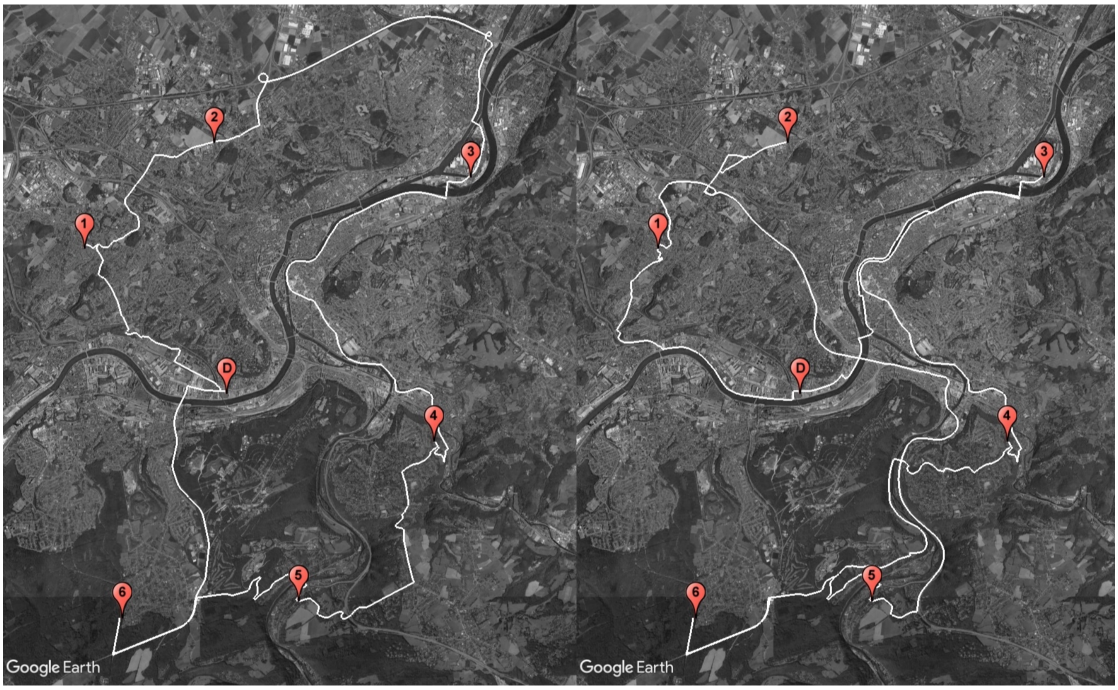

Figure 13: Instance 2 - Tour Representations on Map (left: TSP, right: ETSP)

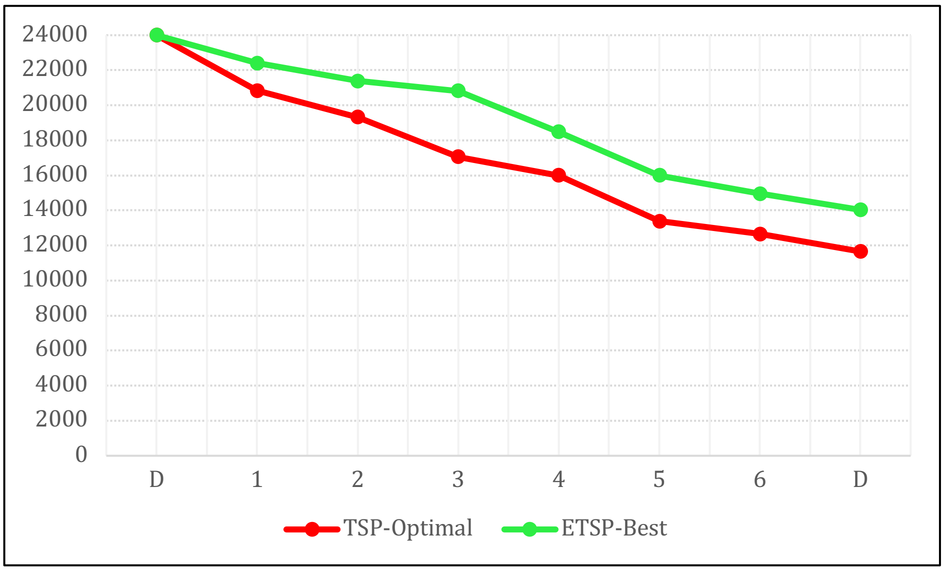

Figure 14: Instance 2 - TSP & ETSP Battery Levels Comparison

A similar situation applies for Instance 3. This time difference in objective function values are not as high as the previous example's; however, the difference in total tour length is increased quite a lot. The results related to Instance 3 are presented in Table 22. Also, tour representations can be found in Figure 15.

Problem	x^*	F^* (Wh)	**Total Length** (m)	$F^*/_{UB}$
TSP	D-3-1-2-4-5-7-6-8-9-10-D	11185.75	61952.00	58%
ETSP	D-2-1-4-9-10-3-5-7-8-6-D	12250.29	87461.00	64%

Table 22: Instance 3 - TSP & ETSP Results Comparison

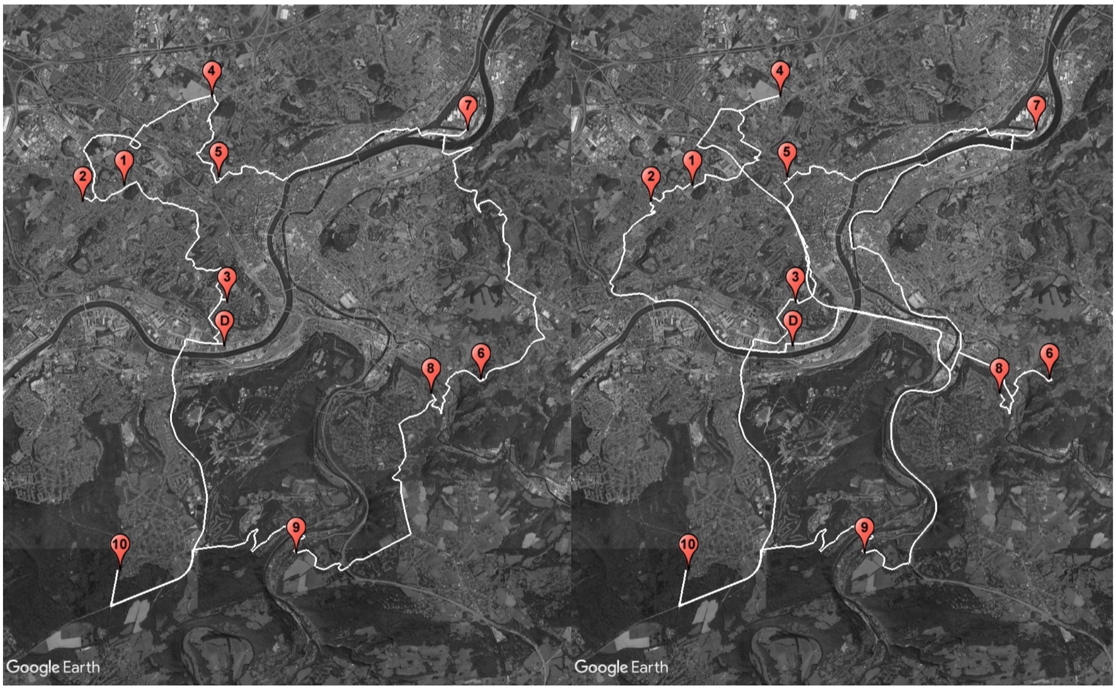

Figure 15: Instance 3 - Tour Representations on Map (left: TSP, right: ETSP)

According to the results provided in Figure 16, available energy levels are synchronized at some nodes, moreover, at the middle of the tour, the preserved battery level is significantly higher in TSP tour, however at the end the objective function value is higher in ETSP tour. A comparison for Instance 4 is also made, and it can be found in Appendix B; however, map representation of Instance 4, is preferred to be not included because of its visual complexity.

Each quantitative outcome of the metaheuristic applications and its prerequisite components were analyzed. Overall evaluation for the obtained results will be done briefly in section 7.

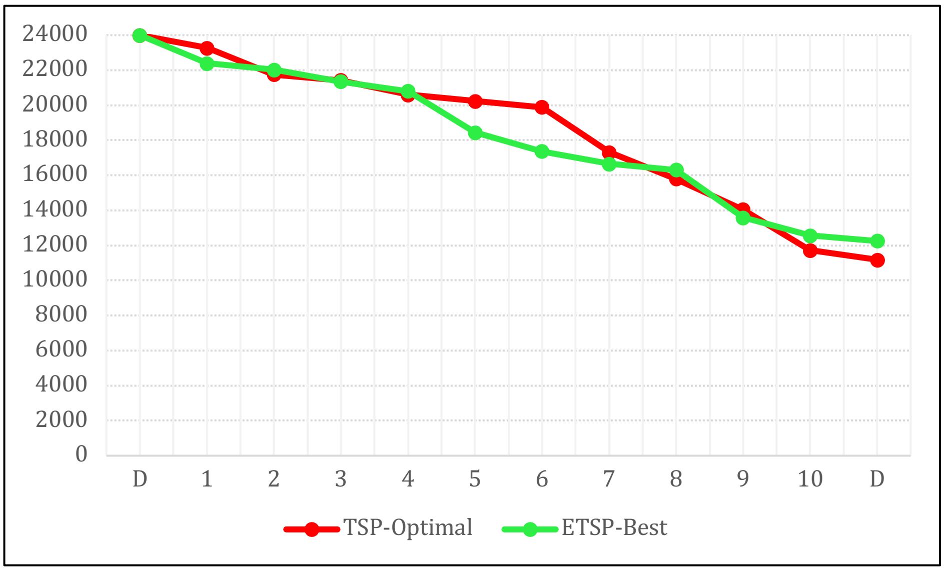

Figure 16: Instance 3 - TSP & ETSP Battery Levels Comparison

6 Project Management

Project management is an essential component of the process for all projects carried out. In the period following the education life, everyone who will create creative works in professional life should be trained in project management. Such a vital process must be carried out within specified standards. In this section, the management process of this project is discussed over the standardized project management process model introduced by the Project Management Institute. The detailed approach can be found in the guide published (Project Management Institute, 2013). In subsection 6.1, the initiation phase of the project was discussed. Under 6.2, the critical components of project planning and execution phases, which are mainly project scope definition, time, resource cost, and risk management, were investigated. In 6.3, the monitoring and controlling phase, and finally in 6.4, project closing phases were examined.

6.1 Project Initiation

During the initiation phase of the graduation project, multiple project options were evaluated. These projects were evaluated in line with the personal study domain preferences, unlike the projects in which the corporations had a revenue expectation. In addition, since this study is a research project, the aim of contributing to the current literature was effective in the evaluation process of the projects. As a result of this evaluation, it was decided that an operation research application on electric vehicles, which is one of the most prominent and current topics of today, is the most appropriate among all options.

This work was made possible by Professor Sabine Limbourg and Professor Maud Bay, who had previously worked on the routing of electric vehicles under this institution, with their willingness to supervise a study on the subject. This current research topic, which coincides with HEC Liège's mission of providing lasting benefits to the broader society, and the introduced 2019-2024 vision of strengthening the current position of the institution to become an internationally recognized research center, has been outlined. Besides, since this is also a graduation study, no team has been established, and the whole project has been carried out by the author of this dissertation.

6.2 Planning and Execution

The planning and execution phases represent the main body of this project. Although these two phases are divided into different processes, they progress in parallel within the duration of the project. The most important parts of these two processes are examined under this subsection.

6.2.1 Defining the Scope of Project

The scope of the project means that the specified outer frame of tasks that will be carried out during a project. It is essential to define requirements, road map, and development stages. The issues that are unclear during the initiation phase are placed in a much clearer frame, and the project is reduced to a more specific area. For this study, this process took place in two sub-stages. First, the application deficiencies in the literature were examined, and the method of operations research to be applied was outlined. It was decided to use a search method, taking into account the computational times of the previous exact methods. Also, a simulated annealing metaheuristic application was planned at this stage. In the second stage, the idea of a second metaheuristic application with different features was approved to compare search method performances, and the scope of the project was finalized.

6.2.2 Time Management

At the point of time management, a road map was initially determined. However, delays have arisen due to the lengthy implementation and development process of the project, and the extensive literature of the traveling salesman problem, perhaps the most studied topic of combinatorial optimization and computer science. As a result of these delays, the schedule was reviewed and updated several times. Eventually, the total project implementation time was extended by 1.5 months. With the experience gained during this project, time management of a similar project can be realized with higher accuracy.

6.2.3 Resource Management

During this project, a lot of technical resource needs emerged. These can be classified as a small amount of published literature, due to the fact that the project is on a current topic and hardware and software performance during the implementation phase. Besides, time was the most prominent resource as mentioned in 6.2.2. During the collection of literature resources, the abstract and citation databases of Web of Science, Elsevier Scopus and Google Scholar were used. Then, a reference management software called Mendeley was used to manage literature resources, and the feature of the software to propose resources similar to those added was often utilized (Mendeley Ltd., 2019). It is aimed to reduce the computational power-related resource problems by using a development computer with up-to-date hardware, and a programming language that is successful in memory management with a fast-running interpreter.

6.2.4 Cost Management

Since the vehicle-related parameters used in this project were collected from past research, a real vehicle was not used for testing. Besides, one of the costs involved in this project was the data collected from Google's cloud service using APIs. At this point, the cost of €397.42, which was incurred at the end of this project, was met with the free credits given for developers that provided by Google Cloud Platform. In order to prevent the increase of the existing cost, the data collected was saved to the databases and reused. Another thing that may be considered as the cost was spent time, whose management is already mentioned under 6.2.2.

6.2.5 Risks Management

The most prominent risk in this project was to increase the amount of mathematical detail to improve the quality of the outcome. Increased detail significantly increased both the time spent on the implementation and computation times during the tests. At this point, managing the risk taken was of great importance, and trade-offs were faced. Multiple initial solution generation methods, three different arc types, four different instance files, and two metaheuristic approaches resulted in 120 different individual test scenarios. In addition to this number of scenarios, repetitive runs were taken with various different parameter sets. By reducing the number of arc types used to reduce this number and fine-tune the parameter set at reasonable intervals, the risk related to time constraint was tried to be reduced. One of the most important reasons why the study could be completed without violating the deadline was the risk management done there.

6.3 Monitoring and Controlling

To monitor the process and to control the outputs, several meetings were held with the supervisors of this study. The advice of the supervisors remedied the deficiencies found in the conceptual issues, and the work took its present form. In addition, during the implementation process of the software to monitor and control the process of optimization, a little application of the spiral model in software development life cycle (SDLC) is preferred. This SDLC model allows to check the outcomes under requirements iteratively, makes it possible to develop more on the same software base, if a new requirement arises.

6.4 Closing Project

After the completion of the tasks within the scope of the project and checking the outputs, the project entered the closing phase. The management of the project from the very beginning to the very end was based on a standardized framework. If it is necessary to make self-criticism, more successful process management could be done in time management and risk management points. In contrast, cost management can be classified as successful. Success in this methodology used in other studies following this study will likely increase.

7 Conclusion

This research aimed to show performances of two local search methods for an extension of famous traveling salesman problem (TSP), called electric vehicle traveling salesman problem (ETSP) considering road grade, load, speed and acceleration, which was introduced in detail in earlier sections. The ultimate aim was solving the problem and observing the performances of steepest ascent and simulated annealing metaheuristics, which were used to obtain optimal or near-optimal results in a shorter time despite the long CPU times obtained using exact methods in the previous studies. In addition, the results of the classical TSP approach were compared with the results obtained by the application of search methods in order to reveal the difference significantly.

On the basis of the results obtained, it is possible to say that the quality of the metaheuristic results that work for much shorter periods is entirely satisfactory despite the optimal solutions provided by exact methods over long periods of time. At the end of tests with different initial solution generation methods, it was observed that the best solution found for the instances used, commonly converged to the same solutions. On the other hand, when comparing the performance of the two search methods, it is possible to say that steepest ascent metaheuristic provides good quality solutions in concise periods of time, but simulated annealing metaheuristic with the used parameter set provides much higher quality solutions by running a little longer.

Because of the nature of steepest ascent metaheuristic, it converges to a local optimum solution, which is sometimes quite far from the global optimum. Conversely, by applying simulated annealing, this problem seems to be easily avoided thanks to its randomized procedure. In addition, an interesting observation about the steepest ascent shows that the low-quality initial solutions can climb to good solutions more quickly. If one should be chosen between steepest ascent, simulated annealing, and exact method to solve the introduced problem, simulated annealing can be said to be a rational choice through solution quality and computation time trade-off.

Another observation is that as the number of customer nodes increases without increasing the total distances to travel needed on maps, the number of infeasible results is dramatically increased thanks to the problem’s combinatorial nature. Furthermore, when TSP optimal results and the best ETSP results are compared, it is seen that the total traveled distance is less important than the road grade as a result of the regenerative braking system. When looking at the effect of customer demands on the results, it is seen that if the deviation between the quantities of the demands is large, the TSP results in much worse results in the context of ETSP.

Future research on this study may include a population-based global search method, such as the genetic algorithm, or a combined approach, such as the memetic algorithm, in addition to the local search methods used. As a further idea, in addition to the arc types introduced in this study, a new method can be explored which would include more accurate slope and distance information at the point of calculating more accurate energy consumption or gain.

Moreover, in this study, it was assumed that acceleration and deceleration were performed at only the beginning and the end of each arc and the maximum speed is always equal to the optimal speed and constant over the journey. Instead, a more realistic consumption calculation can be made with a new method that can dynamically adapt to real-time traffic information and real speed limits of the roads used. As a final idea, a more realistic energy consumption data can be obtained by using an on-board diagnostics (OBD) dongle by connecting it to the OBD port of the vehicle, whose tour would be optimized.

Bibliography

ACEA. (2018). ACEA Report Vehicles in use Europe 2018. *European Automobile Manufacturers Association*, 1–18. Retrieved from https://www.acea.be/uploads/statistic_documents/ACEA_Report_Vehicles_in_use-Europe_2018.pdf

Antosiewicz, M., Koloch, G., & Kamiński, B. (2013). Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed. *Journal of Theoretical and Applied Computer Science*, *7*(1), 46–55.

Artmeier, A., Hasselmayr, J., Leucker, M., & Sachenbacher, M. (2010). The shortest path problem revisited: Optimal routing for electric vehicles. *KI 2010: Advances in Artificial Intelligence*, 309–316.

Balas, E., & Christofides, N. (1981). A restricted lagrangean approach to the traveling salesman problem. *Mathematical Programming*, *21*, 19–46.

Baum, M., Dibbelt, J., Pajor, T., & Wagner, D. (2013). Energy-optimal routes for electric vehicles. *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 54–63. ACM.

Bay, M., & Limbourg, S. (2015). TSP model for electric vehicle deliveries, considering speed, loading and road grades. *Sixth International Workshop on Freight Transportation and Logistics*. Ajaccio.

Bay, M., & Limbourg, S. (2017). *A mixed-integer nonlinear programming model for electric vehicle deliveries considering speed, loading and road grades*.

Bektaş, T., & Laporte, G. (2011). The pollution-routing problem. *Transportation Research Part B: Methodological*, *45*(8), 1232–1250.

Bland, R. G., & Shallcross, D. F. (1989). Large Travelling Salesman Problems Arising from Experiments in X-Ray Crystallography: A Preliminary Report on Computation. *Operations Research Letters*, *8*(3), 125–128.

Blazewicz, J., Eiselt, H. A., Gerd, F., Laporte, G., & Weglarz, J. (1991). Scheduling Tasks and Vehicles in a Flexible Manufacturing System. *International Journal of Flexible Manufacturing Systems*, *4*(1), 5–16.

Burkard, R. E., Deineko, V. G., Dal, R. van, Veen, J. A. A. van der, & Woeginger, G. J. (1998). Well-solvable special cases of the Traveling Salesman Problem : a survey. *SIAM Review*, *40*(3), 496–546.

Carpaneto, G., Martello, S., & Toth, P. (1988). Algorithms and codes for the assignment problem. *FORTRAN Codes for Network Optimization, Annals of Operations Research*, *13*, 193–223.

Carpaneto, G., & Toth, P. (1980). Some new branching and bounding criteria for the asymmetric travelling sales- man problem. *Management Science*, *26*, 736–743.

Christofides, N. (1976). *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*. Pittsburgh.

Crama, Y. (2018). *Computational Optimization Lecture Notes*.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Operations Research*, *2*, 363–410.

Ehsani, M., Gao, Y., Gay, S. E., & Emadi, A. (2005). *Modern Electric, Hybrid Electric & Fuel Cell Vehicles: Fundamentals, Theory, and Design*. Washington D.C.: CRC Press.

Eiselt, H. A., & Sandblom, C. L. (2000). Traveling Salesman Problems and Extensions. *Integer Programming and Network Models*, 315–341.

Eisner, J., Funke, S., & Storandt, S. (2011). Optimal route planning for electric vehicles in large networks. *Proceedings of the Twenty-Fifth AAAI Conference on Arti?Cial Intelli- Gence*, 1108–1113.

Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *48*(1), 100–114.

European Commission. (2011). *Roadmap to a Single European Transport Area - Towards a Competitive and Resource-Efficient Transport System*. Brussels.

Flood, M. M. (1956). The Traveling-Salesman Problem. *Operations Research*, *4*(1), 61–75.

Froger, A., Mendoza, J. E., Jabali, O., & Laporte, G. (2017). *A Matheuristic for the Electric Vehicle Routing Problem with Capacitated Charging Stations*. Montéal.

Garey, M. R., Graham, R. L., & Johnson, D. S. (1976). Some NP-complete geometric problems. *Proceedings of the Eight Annual ACM Symposium on Theory of Computing*, 10–22.

Garfinkel, R. S. (1977). Minimizing Wallpaper Waste, Part 1: A Class of Traveling Salesman Problems. *Operations Research*, *25*(5), 741–751.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, *2*(76), 60–68.

Google Inc. (2019a). Google Maps Platform Directions API Developer Guide. Retrieved from https://developers.google.com/maps/documentation/directions/intro

Google Inc. (2019b). Google Maps Platform Elevation API Developer Guide. Retrieved from https://developers.google.com/maps/documentation/elevation/intro

Gurobi Optimization. (2017). Gurobi Optimizer Reference Manual. Retrieved from https://www.gurobi.com/documentation/

Gutin, G. (2009). Traveling Salesman Problem. In C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (2nd ed., pp. 3935–3944). Springer.

Gutin, G., Yeo, A., & Zverovich, A. (2002). Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*, *117*, 81–86.

Held, M., & Karp, R. M. (1962). A Dynamic Programming Approach to Sequencing Problems. *Journal of the Society for Industrial and Applied Mathematics*, *10*(1), 196–210.

JetBrains s.r.o. (2018). IntelliJ IDEA Documentation. Retrieved from https://www.jetbrains.com/idea/documentation/

Johnson, D. S., Gutin, G., McGeoch, L. A., Yeo, A., Zhang, W., & Zverovich, A. (2002). Experimental Analysis of Heuristics for ATSP. In G. Gutin & A. Punnen (Eds.), *The Traveling Salesman Problem and Its Variations*. Boston: Kluwer Academic Publishers.

Johnson, D. S., & McGeoch, L. A. (2002). Experimental Analysis of Heuristics for STSP. In G. Gutin & A. Punnen (Eds.), *The Traveling Salesman Problem and Its Variations*. (pp. 369–443). Boston: Kluwer Academic Publishers.

Jonker, R., & Volgenant, T. (1983). Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, *2*(4), 161–163.

Kara, İ., Kara, B. Y., & Yetiş, M. K. (2007). Energy Minimizing Vehicle Routing Problem. *COCOA 2007: Combinatorial Optimization and Applications*, 62–71. Berlin: Springer.

Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, 85–103.

Keskin, M., & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, *65*, 111–127.

Keskin, M., Laporte, G., & Çatay, B. (2019). Electric Vehicle Routing Problem with Time-Dependent Waiting Times at Recharging Stations. *Computers & Operations Research*, *107*, 77–94.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680.

Laporte, G. (1992). The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*, 231–247.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1975). Some Simple Applications of the Travelling Salesman Problem. *Journal of the Operational Research Society*, *26*(4), 717–733.

Mendeley Ltd. (2019). Mendeley Reference Management System Guides. Retrieved from https://www.mendeley.com/guides

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, *7*(4), 326–329.

Miller, D. L., & Pekny, J. F. (1991). Exact solution of large asymmetric traveling salesman problems. *Science*, *251*, 754–761.

Misevičius, A. (2004). Using Iterated Tabu Search for the Traveling Salesman Problem. *INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS Informacinės Technologijos IR Valdymas*, *3*(32), 29–40.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.

MongoDB Inc. (2018a). MongoDB Documentation. Retrieved from https://docs.mongodb.com

MongoDB Inc. (2018b). MongoDB Java Driver Documentation. Retrieved from https://mongodb.github.io/mongo-java-driver/3.9/

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, *70*, 113–128.

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, *103*, 87–110.

Naddef, D. (2007). Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP. In G. Gutin & A. Punnen (Eds.), *The Traveling Salesman Problem and Its Variations*. (pp. 29–116). Boston: Springer.

Object Refinery Ltd. (2017). JFreeChart API Documentation. Retrieved from http://www.jfree.org/jfreechart/api/javadoc/

Oliveira, J. F. C., & Ferreira, J. A. S. (1993). Algorithms for Nesting Problems. In *Applied Simulated Annealing* (pp. 255–273). Berlin: Springer.

Oracle Corporation. (2014). Java Platform Standard Edition 8 Documentation. Retrieved from https://docs.oracle.com/javase/8/docs/

Pelletier, S., Jabali, O., & Laporte, G. (2014). *Goods Distribution with Electric Vehicles: Review and Research Perspectives*. Montéal.

Project Management Institute. (2013). *Project Management Body of Knowledge (PMBOK Guide)* (5th ed.). Delaware, PA.

Reinelt, G. (1992). Fast Heuristics for Large Geometric Traveling Salesman Problems. *ORSA Journal on Computing*, *4*(2), 206–217.

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin: Springer.

Roberti, R., & Toth, P. (2012). Models and algorithms for the Asymmetric Traveling Salesman Problem: an experimental comparison. *EURO Journal on Transportation and Logistics*, *1*(1–2), 113–133.

Sachenbacher, M., Leucker, M., Artmeier, A., & Haselmayr, J. (2011). Efficient energy-optimal routing for electric vehicles. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 1402–1407.

Schneider, M., Stenger, A., & Geoke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500–520.

Smith, T. H. C., Srinivasan, V., & Thompson, G. L. (1977). Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems. *Annals of Discrete Mathematis*, *1*, 495–506.

Sweda, T. M., & Klabjan, D. (2012). Finding minimum-cost paths for electric vehicles. *Electric Vehicle Conference (IEVC)*, 1–4. IEEE International.

Sze, S. N., & Tiong, W. K. (2007). A Comparison between Heuristic and Meta-Heuristic Methods for Solving the Multiple Traveling Salesman Problem. *International Journal of Mathematical and Computational Sciences*, *1*(1), 13–16.

Taha, M., Fors, M. N., & Shoukry, A. A. (2014). An exact solution for a class of green vehicle routing problem. *International Conference on Industrial Engineering and Operations Management*, 7–9.

Touati-Moungla, N., & Jost, V. (2012). Combinatorial optimization for electric vehicles management. *Journal of Energy and Power Engineering*, *6*(5), 738–743.

Velednistsky, M. (2017). Short combinatorial proof that the DFJ polytope is contained in the MTZ polytope for the Asymmetric Traveling Salesman Problem. *Operations Research Letters*, *45*(4), 323–324.

Wong, R. T. (1980). Integer programming formulations of the travelling salesman problem. *Proceedings of the IEEE International Conference on Circuits and Computers*, 149–152.

Zhang, C., Sun, J., Wang, Y., & Yang, Q. (2007). An Improved Discrete Particle Swarm Optimization Algo- rithm for TSP. *Proceedings of Web Intelligence/IAT Workshops*, 35–38.

Appendix A: Test Instances

i	Is Depot?	Latitude (φ_i)	Longitude (λ_i)	Demand (q_i)
0	1	50.637167	5.562959	0
1	0	50.640111	5.575585	50
2	0	50.584279	5.569376	30
3	0	50.655340	5.477703	40
4	0	50.646746	5.535289	10
5	0	50.652054	5.577203	60
6	0	50.642572	5.601972	5

Table 23: Test Instance 1 (6 Customers)

i	Is Depot?	Latitude (φ_i)	Longitude (λ_i)	Demand (q_i)
0	1	50.609083	5.558406	0
1	0	50.639576	5.510341	10
2	0	50.662215	5.552913	10
3	0	50.654815	5.638744	10
4	0	50.599060	5.627071	10
5	0	50.565924	5.583126	700
6	0	50.562435	5.525447	10

Table 24: Test Instance 2 (6 Customers)

i	Is Depot?	Latitude (φ_i)	Longitude (λ_i)	Demand (q_i)
0	1	50.609083	5.558406	0
1	0	50.643060	5.524074	700
2	0	50.639576	5.510341	7
3	0	50.618233	5.559093	7
4	0	50.662215	5.552913	7
5	0	50.644801	5.555660	7
6	0	50.602111	5.643550	7
7	0	50.654815	5.638744	7
8	0	50.599060	5.627071	7
9	0	50.565924	5.583126	7
10	0	50.562435	5.525447	7

Table 25: Test Instance 3 (10 Customers)

$\boldsymbol{i}$	Is Depot?	Latitude ($\boldsymbol{\varphi_i}$)	Longitude ($\boldsymbol{\lambda_i}$)	Demand ($\boldsymbol{q_i}$)
0	1	50.592086	5.581066	0
1	0	50.639576	5.541240	10
2	0	50.641447	5.613338	5
3	0	50.624976	5.640804	15
4	0	50.579879	5.617711	25
5	0	50.609083	5.558406	80
6	0	50.639576	5.510341	110
7	0	50.621283	5.602352	10
8	0	50.551093	5.558406	40
9	0	50.642189	5.592365	50
10	0	50.565924	5.583126	90
11	0	50.629124	5.484935	30
12	0	50.599060	5.627071	45
13	0	50.588163	5.535060	20
14	0	50.621718	5.526134	5

Table 26: Test Instance 4 (14 Customers)

Appendix B: Comparison of TSP and ETSP Solutions on Instance 4

Problem	x^*	F^* (Wh)	Total Length (m)	$F^*/_{UB}$
TSP	D-8-10-4-12-3-7-2-9-1-6-11-14-5-13-D	10011.78	71779.00	54,50%
ETSP	D-13-10-8-4-12-7-3-2-9-1-6-11-14-5-D	10549.80	77000.00	57,42%

Table 27: Instance 4 - TSP & ETSP Results Comparison

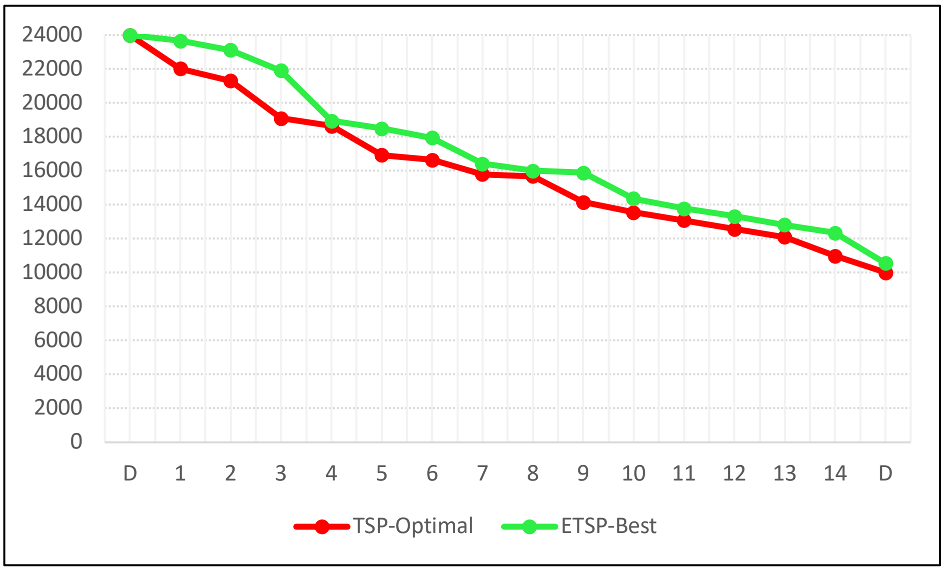

Figure 17: Instance 4 - TSP & ETSP Battery Levels Comparison

Executive Summary

Conventional vehicles with internal combustion engines have caused and are causing severe environmental and health problems such as greenhouse gas emissions, noise pollution, and oil dependency. As a solution to this problem, many European Union countries have introduced regulations for polluting vehicles into the city centers. They also plan to stop the use of diesel vehicles and then gasoline vehicles in the next 15 years. Simultaneously with this process, vehicles that are entirely powered by electric motors continue to be developed, and their market shares increase regularly.

The traveling salesman problem, which has been one of the most studied topics of computer science and combinatorial optimization for many years, has been extended to many different contexts but is developed in general with the logic of conventional vehicles. Contrary to the average fuel consumption of conventional vehicles, and the possibility of fast refueling from the extensive gas station network in the event that the fuel runs out; the energy consumption, even regeneration, of electric vehicles is severely affected by many factors, and in the event of a battery running-out, charging stations are not so common and the charging times are too long.

In this study, a problem aimed at optimizing the battery level at the end of the tour in an urban load distribution case, without considering the recharging of the vehicles, was identified. The energy consumption or regeneration of the vehicle is calculated by considering the road slope, transported load, vehicle speed, and acceleration-deceleration parameters. Besides, taking into account the high CPU times reported in the studies that form the basis of this study, two different metaheuristic approaches have been applied on the problem to find high-quality solutions with much shorter computation times.

At the end of the testing process, good results have been achieved by running steepest ascent and simulated annealing metaheuristics for a considerably short amount of time. When compared these two metaheuristics, simulated annealing provided much better results, as expected because of its randomized nature. Also, it was observed that different solution generation methods used to initiate both metaheuristics had a significant effect on output performance. In addition to all, the comparison of the distance-dependent classical TSP approach's results and the results of the proposed solution method showed that better battery levels could be achieved by quite elongating the tour. These results show the importance and applicability of the introduced approach.