FINDING AN ENERGY EFFICIENT PATH FOR A PLUG - IN ELECTRIC VEHICLE VIA SPEED OPTIMIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BİLGENUR ERDOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

FEBRUARY 2021

Approval of the thesis:

FINDING AN ENERGY EFFICIENT PATH FOR A PLUG - IN ELECTRIC VEHICLE VIA SPEED OPTIMIZATION

submitted by **BİLGENUR ERDOĞAN** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

ABSTRACT

FINDING AN ENERGY EFFICIENT PATH FOR A PLUG - IN ELECTRIC VEHICLE VIA SPEED OPTIMIZATION

Erdoğan, Bilgenur
M.S., Department of Industrial Engineering
Supervisor: Assist. Prof. Dr. Mustafa Kemal Tural

February 2021, 106 pages

Given an origin-destination pair over a directed network, the problem of determining a path joining origin and destination, the speed of a plug-in electric vehicle on each road segment, i.e., arc, along the path, the charging stations the vehicle will stop by, and how much to recharge at each stop so as to minimize the total amount energy consumption of the vehicle is considered. There are speed limits on each road segment, and the vehicle has to arrive at the destination on or before a given time-limit. For this problem, firstly, a mixed-integer second order cone programming formulation is proposed. Secondly, to be able to solve larger size instances, a matheuristic is developed. Lastly, a variable neighborhood search (VNS) heuristic is designed for this problem. Solution quality and computation times of the heuristics and the exact algorithm are compared on different instances.

Keywords: plug-in electric vehicle, min-cost path problem, second order cone programming, matheuristic, variable neighborhood search

ÖZ

FİŞLİ ELEKTRİKLİ ARAÇLAR İÇİN HIZ OPTİMİZASYONU İLE ENERJİ TÜKETİMİNİ ENAZLAYAN YOL BULMA

Erdoğan, Bilgenur
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Mustafa Kemal Tural

Şubat 2021 , 106 sayfa

Başlangıç ve bitişi önceden karar verilmiş bir ağ üzerinde, gidilecek yolu, her yol bölmesi üzerindeki hızı, aracı şarj etmek için durulacak istasyonları ve şarj miktarlarını belirleyen, fişli elektrikli araçlar için toplam enerji tüketimini enazlayan problem üzerinde çalışılmıştır. Her yol parçasında hız sınırları belirlenmiş, tüm yol için zaman kısıtı verilmiştir. İkinci dereceden konik programlama, matsezgisel, değişken komşu arama algoritmaları çözüm yöntemleri olarak kullanılmıştır. Çözüm kalitesi ve hesaplama zamanları belirlenen çözüm yöntemleri için farklı örnekler üzerinde karşılaştırılmıştır.

Anahtar Kelimeler: fişli elektrikli araç, minimum maliyetli yol bulma problemi, ikinci dereceden konik programlama, matsezgisel, değişken komşu arama

To my family...

ACKNOWLEDGMENTS

I would like to thank my supervisor Assist. Prof. Dr. Mustafa Kemal Tural for his support and guidance. He made great contributions to me during this research.

I would also like to present my gratitude to the examining committee members of this thesis Prof. Dr. Meral Azizoğlu, Assoc. Prof. Dr. Sedef Meral, Assist. Prof. Dr. Sakine Batun and Assist. Prof. Dr. Özlem Çavuş İyigün for their feedback and contributions to enhance this study. I am very grateful to be a student of such esteemed academicians. Also, I am proud of being a part of METU-IE department and very grateful to all professors of this department for their teachings and directions to academia.

I thank my parents who encourage me to make research. They never witheld their endless patience and love from me. I am indebted to my husband, Alper, who supported me in any case and any time. I feel grateful to my brother and sister for helping me feel better during stressful times. Furthermore, I really appreciate all the help Arsham Atashi Khoei given me in C++ and CPLEX. I thank also my dear friends Kadir Berkay Aydemir and Caner Sezgin for their supports in coding. Finally, I thank my dear friends and my dear Professors at METU-IE who are always supporting me emotionally. They have a special place in my life.

This study was supported by TÜBİTAK 2210/A National Graduate Scholarship Program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

EV	Electric Vehicle
HEV	Hybrid Electric Vehicle
IVNS	Iterated Variable Neighborhood Search
MCPP-PHEV	Minimum Cost Path Problem for a Plug-In Electric Vehicle
MIP	Mixed Integer Programming
MISOCP	Mixed Integer Second Order Cone Programming
PEV	Plug-In Electric Vehicle
PEVEEP	Plug-in Electric Vehicle Energy Efficient Path Problem
PEVEEP-MINLP	Mixed Integer Nonlinear Programming Type Energy Consumption Minimization Model
PEVEEP-MISOCP	Mixed Integer Second Order Conic Programming Type Energy Consumption Minimization Model
PEVFEEP-SOCP	Mixed Integer Second Order Conic Programming Type Energy Consumption Minimization Model along a fixed path
PHEV	Plug-In Hybrid Electric Vehicle
PEVTEPP-MIP	Mixed Integer Programming Type Time Minimization Model
PEVTEPP-MISOCP	Mixed Integer Second Order Conic Programming Type Time Minimization Model
SOCP	Second Order Cone Programming
TTRP	Truck and Trailer Routing Problem
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VNSB	Variable Neighborhood Search Branching
WCSPP	Weight Constrained Shortest Path Problem

CHAPTER 1

INTRODUCTION

In recent years, number of studies related with CO_2 emissions, sustainable systems and green problems involving environmental concerns in their objective functions increased. More research will be made in near future thanks to the increasing environmental awareness of humanity to protect "our home", the world.

In this direction, electric vehicles make a major contribution to have a greener environment. Due to their lower tailpipe carbon dioxide emissions compared to vehicles with internal combustion engines, they will be more attractive year by year.

A recent study shows that transportation, mainly fuel consumption of transportation, is responsible for 24% of global total energy consumption [2]. This consumption can be decreased by using fewer vehicles or adapting advanced engine technologies. In addition to saving from emissions, electric vehicles also do not produce noise pollution. Hence, governments start to take preventive actions to reduce harmful gases which causes global warming and extinction of living creatures in the long term. For example, Energy Union put some obligations and targets on emissions for new cars produced in European Countries between 2025 and 2030 [3], and it is estimated that these limitations will provide 15% to 37.5% reductions in emissions [3]. Therefore, the use of electric vehicles has gained popularity.

There are 3 types of electric vehicles that are currently in use. First one is plug-in electric vehicle (PEV). These vehicles only use electricity as an energy source. They should be plugged in to recharge itself. As they have a limited battery capacity, recharging stations are needed to address their charging needs. The second type of electric vehicles is hybrid electric vehicles (HEV). These vehicles use two types of energy resources, electricity and fuel. They have internal combustion engines which convert fuel energy to electrical energy while driving. Thirdly, plug-in hybrid electric vehicles (PHEV) include both internal combustion and electrical engines. They can be recharged by plugging in like PEVs. When they are out of electrical energy, they switch to internal combustion engine to provide continuous driving like HEVs. This classification can be seen in Table 1.1.

From both economic and environmental perspectives, electric vehicles have some benefits. However, they have also some limits for users. PEV users want to know more about driving ranges, driving costs and tax-related issues, i.e. governmental incentives, of PEVs. PEVs have limited battery capacities and can not make long distance trips without recharging due to their limited ranges. On the other hand, HEVs can recharge themselves using fuel energy while driving which provides fewer number of stops to recharge and shorter travel times. PHEVs can also make long distance trips using both engines while reducing emissions by electrical engine and switching to internal combustion engine when out of electrical energy. So, PEVs and HEVs need electrical recharging stations and fuel refilling stations on the road, respectively. On the other hand, PHEVs use both electrical recharging stations and fuel refilling stations on the road. Number of recharging stations around, traffic congestion, speed of the vehicle, distances of the roads, time windows, driver skills, physical conditions of the road are some of the factors affecting the performance of electric vehicles.

In this study, we used a PEV on a network with several stations where the vehicle can be recharged. There is a directed network where we are trying to find a path connecting predetermined origin and destination nodes. Moreover, the problem decides the speed of the vehicle optimizing energy consumption along the path. In order not to be out of energy while driving, the vehicle has to stop at electrical recharging stations whose locations are predetermined in this network. In the literature, many studies assume that the vehicles travel at a constant speed; in our study, however, speed is a decision variable. For each road segment there are speed limits and the driver can drive at any speed without violating these limits. These speed limits may be given as parameters to the problem reflecting the real life, legal speed limits. Speed limits may also be determined using traffic congestion. If there is a traffic congestion, a vehicle can drive at a speed level impacted by the traffic conditions. Moreover, there is a time

Table 1.1: Electric Vehicle Types

Vehicle Type	**Engine Type**	**Energy Outsource**
Plug-in Electric Vehicle (PEV)	Electrical Engine	Electricity
Hybrid Electric Vehicle (HEV)	Electrical Engine Internal Combustion Engine	Fuel
Plug-in Hybrid Electric Vehicle (PHEV)	Electrical Engine Internal Combustion Engine	Electricity Fuel

limitation on total travel and charging time. Toward these properties, an optimization problem finding an energy efficient path is analyzed, which decides the speed of the vehicle based on energy consumption while satisfying the time and speed limits. This problem is defined as finding an energy efficient path for a plug-in electric vehicle via speed optimization. Throughout the thesis, it is abbreviated as PEVEEP which means plug-in electric vehicle energy efficient path problem. To summarize, the following decisions are included in our model:

- Finding a path joining origin to destination considering connectivity of the nodes over the network
- Determining the speed of the vehicle on each road segment along the path
- Determining where to stop to recharge the battery and how much to recharge

In order to make these decisions a mathematical model is constructed. We assumed an electric vehicle with some energy at the beginning of the trip which has a pre-determined energy consumption behavior. This behavior is defined using an energy consumption function in the literature, which is a function of speed. This function is an equation of the third degree which makes the problem nonlinear. This problem aims to minimize the energy consumption while making sure that the destination is reached before the time limit. Also, violation of the speed limits is not allowed.

One of the minimum cost path problems is weight constrained shortest path problem (WCSPP). In WCSPP, each arc in the network has an associated weight in addition to distance. This problem tries to find the shortest path between origin and destination nodes where total of arc weights should be less than a predetermined value. This problem is known to be NP-hard [4]. The problem considered here is a generalization of the WCSPP. Consider an instance of the WCSPP, where each arc has a positive length and weight. In our problem, energy consumption over travel time per unit distance can be any value greater than or equal to some constant c assuming that there are no time limits. We scale the lengths in WCSPP such that length over the weight is at least c for every arc. Now the weights in WCSPP become our travel times and lengths become energy consumptions, where the speeds are fixed in every arc. Furthermore, we assume that the initial energy is large enough so that there is no need to stop at a station to recharge. Thus, we have converted an instance of the WCSPP to an instance of PEVEEP. So, the PEVEEP is NP-hard as well.

A mixed integer second order cone programming formulation is provided to solve the PEVEEP. Moreover, as the formulation becomes inadequate for larger instances, at matheuristic and a VNS heuristic are developed for the problem. Solution quality and computation times are analyzed. A contribution of this study is introduction of speed optimization while finding an energy efficient path. Up to now, related studies work with constant predetermined speeds on a network. On the other hand, speed is a decision variable in our problem. Based on this problem definition, we solved instances from the literature that are used for another problem previously. A matheuristic approach is also introduced in this thesis. This approach first tries to find a feasible path on the directed network. Feasibility in terms of time is provided by the help of a new problem definition: time efficient path problem for an electric vehicle (PEVTEPP). This problem is a mixed integer programming (MIP) problem that aims to find a path on a network which satisfies a total time restriction where speed is a parameter. On the other hand, feasibility in terms of energy is provided by the help of another problem definition: energy efficient path problem for an electric vehicle along the fixed path. This problem is modeled as second order cone programming (SOCP) problem which minimizes the energy consumption along the decided path previously (PEVFEEP-MISOCP). A third degree function of a speed is used in the objective function of this problem. At the same time, speed is a decision variable, and the model is nonlinear. After feasibility is provided, matheuristic makes improvements on energy consumption level by considering alternative paths and using PEVFEEP-MISOCP model.

Our second solution method is a variable neighborhood search (VNS) heuristic which includes different local search methods. This heuristic starts with an initial path which is constructed using an initialization method. Then, improvement on this path is made using the defined local search methods while satisfying time and energy feasibility. In other words, the algorithm starts with a feasible or infeasible path, and results with a feasible path whose energy consumption level is the smallest one among other tried paths throughout the VNS algorithm implementation.

Organization of the thesis is as follows: Chapter 2 includes a literature review of the related studies, instances, parameters and heuristic approaches. Problem definition is given in Chapter 3. Solution approaches for the PEVEEP are detailed in Chapter 4. The details of computational experiments are provided in Chapter 5. Finally, conclusion and future research directions are presented in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

Minimum cost path and routing problems, behavior of electrical vehicle batteries, matheuristic and metaheuristic approaches for these types of problems are reviewed in detail in this chapter.

2.1 Minimum Cost Path and Routing Problems for Electric Vehicles

In 2014, Schneider et al. studied a PEV routing problem with time windows and recharging stations [5]. They consider requirement of visits to recharge stations during a tour based on limited battery capacity. As a solution method, a heuristic combining two metaheuristics, neighborhood search and tabu search, is proposed. Positive effect of this hybrid metaheuristic is investigated. A mixed integer programming formulation is also provided which minimizes the total distance traveled on a network. In addition to classical vehicle routing problem, stopping decisions are made in this study. Connectivity of customer visits and recharging stations, and flow conservation constraints are included. Moreover, time limits for recharging stations and travels on each road segment are given as constraints.

In 2014, Baum et al. also studied a PEV routing problem. They stated that the energy-optimal tour can be too long, or the fastest route can be infeasible as energy consumed per distance increases [6]. So, a bicriteria optimization to obtain Pareto sets of routes is considered. Velocity can change on each road segment but a limited number of discrete speed values are available. Baum et al. aims to propose a full Pareto set of all nondominated paths where speed variables a re not continuous.

A minimum cost path problem for PHEVs without any time constraints or cycles is studied in [4]. Cycle occurs when PHEV drives at a loop starting and finishing at the same node in order to produce electrical energy using its hybrid property. The problem tries to find a path with minimum total energy consumption between origin and destination nodes. Shortest path problem is a special case of this problem where the cost is the total distance of the path. As the problem is NP - hard, a mixed integer quadratically constrained formulation, a discrete approximation dynamic programming heuristic, and a shortest path heuristic are presented in this paper as solution methodologies. Depreciation, degradation, stopping and route costs are included in the objective function. Energy consumption is only dependent to distance, the consumption rate between two nodes is taken as constant. This is the most related study with this thesis in the literature. The difference between these two problem definitions are the cost terms included and the definition of speed as a variable.

Likewise, a problem of energy - efficient shortest routes for PHEVs is studied in [7]. It is stated that recharging takes much more time than refueling; thus, there should be a balance between speed, range and selection of stops. A mathematical model of finding shortest routes on a network is provided in the paper. It differs from the other studies in the literature with respect to recharging type of the vehicle. Recharging can occur both at nodes and on edges. As there can be recharging on edges, the vehicle desires to make a cycle when the energy consumption is negative. Hence, the structure is more complicated than the classical shortest path problem, and the main concern is to prevent these cycles on a route. A fully polynomial-time approximation scheme is proposed for the problem.

A time dependent electric vehicle routing problem is studied in [8] which involves a fleet of electric vehicles and a set of customers to be served. Departure times and speed on each arc are decided while minimizing a cost function representing total energy consumption. An integer linear programming model is proposed. Also, an iterated variable neighborhood search (IVNS) and a variable neighborhood descent method (VND) are proposed in addition to a speed optimization method. Time dependency comes from time windows on each node and traffic congestion on each arc. An electric vehicle routing problem with time windows is proposed by Goeke and Schneider [5]. Cost function here includes energy cost (electric price), driver's wage for each vehicle, purchasing of vehicles and time windows penalties. VND and VNS procedures first decide optimal routes without optimizing speed. After that, departure time and speed optimization process starts to solve. Partial recharging is not allowed in this study. Here, IVNS finds new solutions for some benchmark instances available in the literature.

Another problem considering state-of-charge optimization of PHEVs is proposed by Vasant et al. [9]. This problem tries to make minimum number of stops and optimize the use of energy in the battery. Particle swarm optimization and Gravitational search algorithm are applied. Partial recharging is allowed, and each charging station has a capacity for the number of vehicles it can serve. State of charge and waiting tie at each station are decided under capacity limitations. Different nature-inspired algorithms are used to make charging decisions only.

2.2 Energy Consumption Behavior of Electric Vehicles

In this thesis, the minimum cost path problem is studied for a PEV. Differently from the literature, the problem aims to find a path with some recharging stations, and energy consumption also depends on speed. An energy consumption function for an electric vehicle battery is used as the objective function. Finding an accurate model for energy consumption is the most important part of the parameter selection.

There are different models in the literature for energy consumption function estimation with a lot of parameters to predict the behavior of the battery. In [10], a prediction function is suggested based on real world data. This study provides a formulation for energy requirement at the wheels. It includes 5 parts: the rolling resistance, potential energy, aerodynamic losses, kinetic energy and energy for the acceleration of rotational parts. It also notices the energy loss from heating and air - conditioning systems.

An analytical model determining a time-dependent optimal velocity profile for an EV so that it minimizes electricity usage along the path considering route characteristics and traffic conditions is solved in [11]. It is stated that battery consumption is lower with smoother acceleration and deceleration values. A formula for EV's instan-

Table 2.1: Energy Consumption Function Parameter Values used in [1]

Parameter	**Value**	**Unit**
M	1500	kg
f_r	0.015	-
r	0.110	Ω
A	1.800	m^2
g	9.810	m/s^2
R_t	0.300	m
P_a	2	kW

taneous power is provided, and energy consumption is estimated by integrating the function over time. The paper uses the parameters of another study from the literature which is published by Wu et al. in 2015 [12]. A data collection system is established in order to specify the values of vehicle weight, resistance of motor, radius of tire etc.

Recently, there is a study about estimation of the energy consumption behavior of the battery in an EV [1]. Instantaneous power (P) estimated in [1] is defined by the function;

$$P = (AV^2 + f_rMg + BV/R_t) \times V + rR_t^2/K^2 \times (AV^2 + f_rMg + BV/R_t)^2 + P_a$$

where V is the speed of the vehicle, r is the resistance of the conductor, M is the mass of the vehicle, g is the gravity acceleration, A is the aerodynamic constant, f_r is the rolling resistant constant, B is bearings' damping coefficient, K is armature constant, R_t is tire radius , and P_a is the ancillary loss including the loss sourcing from air condition, external lights and audio. Parameters of this model are provided in Table 2.1. As acceleration and deceleration are neglected in our study, instantaneous power estimation function is multiplied by t in order to generate energy consumption function of a battery of an electric vehicle. In this study, the model proposed by Li et al. in [1] is used in this thesis.

2.3 Matheuristic Approaches for Minimum Cost Path and Routing Problems

A matheuristic is a heuristic combining mathematical programs and heuristics. Heuristics generally ease the way of generating a solution compared to exact algorithms. Heuristic improvements make the solution algorithm run faster, while exact algorithms try to give solutions with minimum optimality gap. In this thesis, four different mathematical programs are used. One of them is a stand alone solution approach for the PEVEEP and the others used as subroutines within other solution approaches. In order to improve the solution, a heuristic based on edge exchange is applied at the end. Inspirational studies on similar problems in the literature are mentioned below.

In 2014, Archetti and Speranza provided a survey on matheuristics for routing problems. They classified the heuristics as decomposition, improvement and column generation-based approaches [13]. By the help of mathematical models, matheuristics obtain high quality solutions. There are more than twenty studies on routing problems under each classification in the last decade. As there are few studies related to minimum cost path problem for a PEV, matheuristics on routing problems can be inspirational. Likewise, Villegas et al. studied a matheuristic for truck and trailer routing problem (TTRP) in [14]. They proposed simple two phase matheuristic. First step is to find the local optima in a set – partitioning formulation of TTRP. Second phase uses the local optima for matheuristic. It is stated that studied matheuristic outperforms state of the art methods in terms of both solution quality and solution time.

Moreover, a matheuristic approach is proposed for the pollution – routing problem by Kramer et al. [15]. The objective of the problem is minimizing environmental costs considering capacity and time windows constraints. Matheuristic is constructed in three steps. First step is local search-based metaheuristic. Second step is integer programming approach that uses the routing decisions at first step. At the end, speed is decided based on a speed optimization algorithm. By the help of hybridization, both routing and speed decisions are made. It is stated that the proposed method is better than previous algorithms in terms of solution quality and time.

Travel times and CO_2 emissions are analyzed under time - dependent vehicle routing problem by Franceschetti et al. [16]. Concept of this vehicle routing problem is similar to the problem we study in terms of speed optimization. As the amount of CO_2 emission is correlated with speed, speed limit is applied. A tabu search procedure is proposed. Routing decisions are made assuming a constant speed. Then, speed, time and emission improvements are made in later steps.

When electric vehicles are concerned, there are 3 main studies involving heuristic approaches for EV routing problems. Montoya et al. proposed a metaheuristic considering nonlinear charging functions and limited ranges of electric vehicles [17]. This nonlinearity makes the solution process more difficult though there is no speed optimization. Therefore, infeasibility occurs frequently. Routing decisions and charging decisions are made iteratively by the help of iterated local search (ILS) and a heuristic concentration in order to ensure feasibility. Although the proposed metaheuristic may sometimes cause overly expensive solutions, it performs well on some instances.

Bruglieri et al. studied a matheuristic for EV routing problem with time windows [18]. They also proposed a time efficient route considering recharging requirements. Although they proposed a mixed integer linear programming formulation, a variable neighborhood search branching (VNSB) matheuristic is also designed because of the hardness of the problem. Numerical results are compared with previous VNSB methods in terms of solution quality and time.

Likewise, a matheuristic for EV routing problem with capacitated charging stations is proposed by Froger et al [19]. Differently from the literature, charging stations are not assumed to be uncapacitated. They can recharge a limited number of electric vehicles at the same time. This assumption affects routing decisions in terms of where to stop. There is again a non - linear charging function for each EV. Mixed integer programming formulation is proposed. Then, a matheuristic named route first and assemble second is developed for the problem. Relaxing capacity constraints, route decisions are made at first stage. Second stage works for assembling the routes according to charging decisions by the help of Bender's decomposition. Furthermore, minimum cost path for PHEVs without any time constraints is the most similar study to the problem in [4]. Speed decisions are excluded as charging function is only dependent to distance. Based on that, a shortest path heuristic is presented in this paper as a solution methodology [4]. In step 1, problem chooses a path by the help of the solution of a shortest path problem on a network. Then, where to stop decisions are made. Also, because of degradation cost, the heuristic considers the amount of recharge. Likewise, here we consider time limits, battery capacities and speed limits. As charging function of EV depends on velocity here, different solution approaches are proposed in this thesis. We proposed solution methods considering speed as a decision variable differently from the study in [4] which uses shortest path heuristics.

2.4 Meta-Heuristic Approaches for Routing and Energy Consumption Optimization Problems

Many meta-heuristic approaches come into play when exact algorithms become insufficient to find an optimal solution in a limited time. Exact solution approaches can take too much time to solve an optimization problem due to the complexity of these problems. Hence, many meta-heuristic methods have been developed in the recent past. Local and global search methods are the mainly used by meta-heuristic approaches. Simulated annealing, tabu search, variable neighborhood search and iterated local search methods are based on improving local solution in order to obtain better ones. Memetic algorithms, nature-inspired and metaphor-based heuristics can also be accepted as recently popular solution methods. Meta-heuristic methods balance precision, quality, and accuracy versus space and time efficiency [20].

Abousleiman and Rawashdeh propose two metaheuristics, ant colony optimization and particle swarm optimization, to find an energy efficient route for electric vehicles [20]. Dijkstra's algorithm can be accepted as the most popular algorithm used in routing problems. However, it becomes insufficient by growing concerns of real-life routing problems. This method needs positive edge costs and gives a shortest path route based on problem inputs. Though it gives an exact solution in a considerable time, optimal solution to the energy efficient routing problem is not always based on the shortest path of the instance. For example, in this study we are trying to include speed optimization and define velocity as another positive cost on edges.

Furthermore, Bellman-Ford's algorithm can cope with negative cycles over a network in order to find the shortest path but with increasing complexity of the algorithm. Negative cycles mainly occur while using HEVs as they are producing electrical energy by the help of combustion engine on driving mode with fuel. So, it is stated in [20] that traditional shortest path algorithms fail to solve the energy efficient path problems due to increased number of concerns and decision variables in those problems. Here, the paper finds a path from source to destination based on some selection parameters such as; speed limits, elevation changes, length of edges, battery capacity and traffic lights. Bellman-Ford's algorithm uses an energy consumption equation of the battery. This equation includes selection parameters mentioned. Moreover, Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) algorithms try to find minimum energy value using this equation. Both algorithms find optimal solutions fast on some specific instances in the literature. Simplicity of PSO provides lower solution times compared to ACO algorithm. The study of Abousleiman et al. is another paper about this ACO algorithm [21] on EVs. It is stated that some solutions reveal significant improvements in the energy consumption of the electric vehicle if recommended tour is driven compared to tours suggested by Google Maps and MapQuest. In addition to nature-inspired algorithms, a simulated annealing algorithm for PHEVs is proposed by Vincent et al. [22]. This study focuses on an extension of Green Vehicle Routing Problem for plug-in hybrid electric vehicles. It tries to optimize electrical energy and fuel consumption considering availability of electric charging and fuel stations [22]. It constructs initial solution using the Nearest Neighbor algorithm. Vehicle energy capacity and time constraint are also taken into consideration while taking a tour based on distances. Simulated algorithm defined here, uses penalty costs in the objective function to handle time and energy capacity constraints. Some capacitated vehicle routing benchmark problems are used to try proposed simulated annealing approaches. It is shown that average difference between solutions found and optimal ones is the lowest in the literature [22].

In 2005, Ropke and Pisinger suggested an adaptive large neighborhood search method for the Pickup and Delivery problem [23]. The problem tries to serve all customers whose pickup and delivery locations are known using limited number of vehicles. A cost function is constructed to minimize, and time windows and capacity constraints are taken into consideration in the mathematical model. Here, three different parts are involved in the objective function. Sum of distances traveled by vehicles, sum of the travel times of the vehicles and total number of requests met are multiplied by weights in the objective function. Proposed method uses insertion and removal algorithms. At each iteration, algorithm chooses the next customer using a measure including a distance term, a time term, a capacity term and the vehicles serving the customer [23]. In addition to this, adaptive large neighborhood search uses the information of past performance of each neighborhood. Then, it chooses the neighborhood search performing well. Therefore, algorithm finds slightly better solutions in a faster manner for some vehicle routing problem benchmarks in the literature.

CHAPTER 3

ENERGY EFFICIENT PATH PROBLEM FOR A PLUG – IN ELECTRIC VEHICLE VIA SPEED OPTIMIZATION

In this chapter, the PEVEEP is defined and a mathematical model is presented.

3.1 Problem Definition

A problem about energy optimization of a PEV over a network is proposed in this thesis. Minimization of energy consumption by optimizing speed over a network and making charging decisions under given time and speed limits are aimed in this problem. In the literature, in most of the related stuides energy consumption functions are only dependent on distance. These problems are similar to the Shortest Path Problem and Minimum Cost Path Problem. In this study, a nonlinear energy consumption function dependent also on speed in addition to distance is considered.

It is known that weight constrained shortest path problem (WCSPP) is NP–hard for directed graphs [4]. The problem tries to find a shortest path from source to sink node where total weight of the path is less than some predetermined value. Likewise, we are trying to find a path with minimum energy consumption between origin and destination nodes while satisfying time and energy constraints for each arc. Here, weights of arcs can be accepted as travel times. As every instance of the MCPP-PHEV can be turned into an instance of the PEVEEP, our problem is also NP-hard.

A nonlinear function representing the behavior of an electrical vehicle battery is taken as the objective function in our problem. This function is based on the model suggested by Li et al. [1]. Power loss of an electric vehicle is as follows:

$$P = (AV^2 + f_r Mg + BV/R_t) \times V + rR_t^2/K^2 \times (AV^2 + f_r Mg + BV/R_t)^2 + P_a$$

$$P_a = P_{ac} + P_{bm} + P_{el} + P_{au}$$

where P_a is the ancillary power loss. It includes power used by air conditioner, battery management, external lights and audio. It is stated that all of them are independent of velocity. P represents the total power loss including ancillary power loss. In this function, acceleration of the vehicle is included. As it is a term dependent to the time passes, by taking the integral of instantaneous power estimation function with respect to time passes, we find an energy consumption function of this PEV. However, it is assumed that acceleration is zero all the time, and speed is constant throughout an edge over the network in this problem. Hence, the terms including acceleration are neglected. At the end, instantaneous power estimation function is multiplied by the time spent in order to get the energy consumption function on each edge. After simplifications, energy consumption function per unit distance traveled becomes as follows:

$$E = a \times V^3 + b \times V^2 + c \times V + d + e/V$$

Corresponding coefficient values of the terms including orders of velocity, V, can be seen in Table 3.1. Values of coefficients are calculated based on the parameters given in the Table 2.1.

As there is no acceleration and the velocity remains the same throughout the edge once decided. However, an electrical vehicle can have different speed values on different edges because of various speed limits. Upon this scheme, an optimization problem is analyzed which decides the speed of the vehicle considering battery usage, time and speed limits. It is known that an electric vehicle has limited driving range due to its limitations on the number of batteries it can have, battery capacity and recharging stations around. Because of their technologies, batteries can offer a specified level of energy. Each battery has a weight. If M, weight of the vehicle, increases, then the energy consumption of the vehicle increases as well. Hence, the number of batteries on an electric vehicle should be kept at the minimum level. Therefore, number of stops can be too much with increasing speeds and distances. In this problem, vehicle wants to go faster due to time limitation. On the other hand, en-

Table 3.1: The Values of Coefficients in Energy Consumption Model

Coefficient	**Value**	**Unit**
$a = (rR_t^2/K^2)A^2$	1.023×10^{-5}	$kg \cdot s/m$
$b = A + (2ABrR_t/K^2)$	0.324	kg
$c = B/R_t + (2MgAf_r + B^2)r/K^2)$	3.348	$kg \cdot m/s$
$d = f_rMg(1 + 2BrR_t/K^2)$	220.868	$kg \cdot m^2/s^2$
$e = f_r^2(Mg)^2rR_t^2/K^2 + P_a$	2004.747	$kg \cdot m^3/s^3$

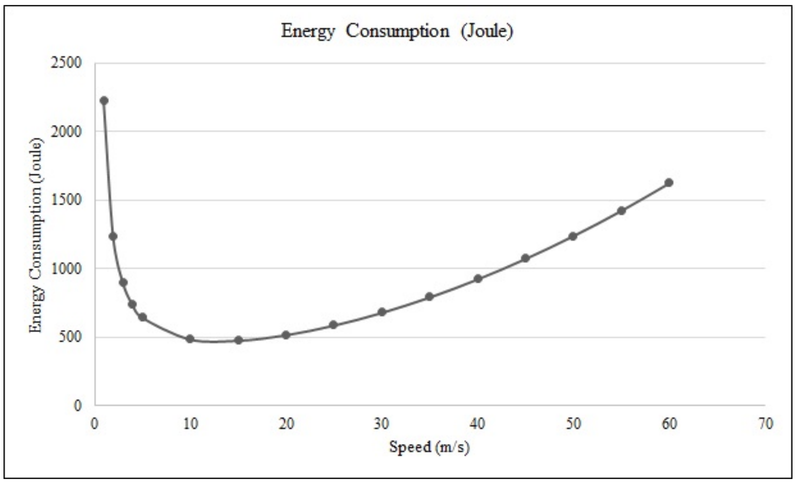


Figure 3.1: Energy Consumption Function per unit distance traveled of a PEV

ergy consumption should be lower in order to have a smaller number of requirements for recharging and energy saving purposes. Under these tradeoffs, a formulation is proposed to find the optimal speed, travel time, charging time, charging amount and consumed energy on each edge of the decided path.

Before proposing our solution methods for this problem, energy consumption behavior of the PEV is analyzed. The energy consumption function of a PEV with respect to speed is displayed in Figure 3.1 considering the parameter values in Table 3.1. The function provided has a convex shape where it takes different values according to changing speed values. It has a local minimum when V is equal to 13.035 m/s (46.926 km/h). This is an acceptable speed level for urban road segments, while the vehicle may be obliged to drive faster on inter-urban road segments in real life.

3.2 Mathematical Model

Using the energy consumption function as objective, the PEVEEP tries to find a path from source node to sink node. There is a directed network, $G = (\mathcal{N}, A)$, given to the problem where the model tries to find a path. $\mathcal{N}$ represents the set of nodes and A the set of arcs. To specify the decisions of path selection, some binary variables are added to the model. This path is decided according to V and D values of each arc to minimize energy consumption over the network. As objective function is a third degree function of V, it has a significant impact on the total energy consumption value. This makes the problem nonlinear. Moreover, V should be in between specified speed limits. Hence, we included constraints to put limits on decision variable V. In addition to these, the sum of travel times and charging times should be less than a given time limit. Energy balance should be constructed between entering and leaving each node. Hence, there are constraints including the energy levels of each node on the path and energy consumption levels on each arc. Moreover, PEV has to recharge according to its initial energy level in battery. This initial energy level is a predetermined value which is higher than the lowest energy level and lower than the highest energy level. In order not to be out of charge while driving, there is an amount where PEV has to reservoir. Battery on the PEV has also a capacity reflecting the real life conditions. All the equations related to energy level are written including the battery capacity and energy stock. Energy level, velocity and travel time variables are nonnegative variables in the proposed mathematical model. In addition to those, charging time is measured using the variables representing energy taken at each station. There is a fixed time for a PEV per stop at a recharging station. This waiting time may correspond to overcrowding at a station. As each recharging station has a capacity and recharging takes long times, a vehicle may enter the queue at the station. This fixed time is represented in the mathematical model using binary variables. In addition, PEV spends time per unit energy taken. Total charging time and total travel time spent over the network must be less than the total time limit. An arc cannot be involved in the route again if it is used previously. This path construction is provided by the help of classical shortest path problem constraints.

Notation and formulation are given next.

3.2.1 Notation

The notation used in the mathematical model over the directed graph $G = (\mathcal{N}, \mathcal{A})$ is as follows:

Sets:

$\mathcal{N}$: Sets of nodes, $\mathcal{N}$ = $\{1, ..., N\}$ where 1 is the origin node and N is the destination node

$\mathcal{A}$: Sets of arcs or road segments

S: Sets of stations, $\mathcal{N} \supseteq S$

Parameters:

$$S_i = \begin{cases} 1 \text{ if } i \in S, \\ 0 \text{ otherwise} \end{cases} \quad i \in \mathcal{N}$$

U_{ij}: Upper speed limit on each arc $(i, j) \in \mathcal{A}$ (m/sec)

L_{ij}: Lower speed limit on each arc $(i, j) \in \mathcal{A}$ (m/sec)

C: Capacity of the battery $(Joule)$

m: Minimum charge level of the battery $(Joule)$

I: Starting charge level of the battery $(Joule)$

A: Time spent per stop to recharge (sec)

B: Time spent per unit energy recharged (sec)

D_{ij}: Length of each arc $(i, j) \in \mathcal{A}$ $(meter)$

T: Limit on the time elapsed between origin and destination nodes (sec)

a: Coefficient of the variable $(V)^3$ in the energy consumption function $(kg \cdot s/m)$

b: Coefficient of the variable $(V)^2$ in the energy consumption function (kg)

c: Coefficient of the variable (V) in the energy consumption function $(kg \cdot m/s)$

d: A constant in the energy consumption function $(kg \cdot m^2/s^2)$

e: Coefficient of the variable $(V)^{-1}$ in the energy consumption function $(kg \cdot m^2/s^2)$

M: A large constant value

1: Origin node

N: Destination node

Decision Variables:

E_{ij}: Battery consumption of the vehicle per unit time on each arc $(i, j) \in \mathcal{A}$ $(Joule)$

V_{ij}: Velocity of the vehicle on each arc $(i, j) \in \mathcal{A}$ (m/s)

ec_i: Charge amount of the battery at node $i \in \mathcal{S}$ $(Joule)$

ea_i: Charge level of the battery at the arrival to node $i \in \mathcal{N}$ $(Joule)$

ed_i: Charge level of the battery at the departure from node $i \in \mathcal{N}$ $(Joule)$

t_{ij}: Travel time on arc $(i, j) \in \mathcal{A}$ (sec)

ct_i: Time spent on charging the battery at node $i \in \mathcal{S}$ (sec)

$$x_{ij} = \begin{cases} 1 \text{ if } (i,j) \in \mathcal{A} \text{ is included in chosen path,} \\ 0 \text{ otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 \text{ if PEV stops at node } i \in \mathcal{S} \text{ to recharge,} \\ 0 \text{ otherwise} \end{cases}$$

3.2.2 Formulation

A mathematical programming model is proposed for the PEVEEP here.

(PEVEEP-MINLP)

$$\text{Minimize} \sum_{i \in \mathcal{N}} \sum_{i \in \mathcal{N}} E_{ij} \tag{3.1}$$

subject to

$$E_{ij} \geq (a \times V_{ij}^3 + b \times V_{ij}^2 + c \times V_{ij} + x_{ij} \times d + e \times V_{ij}^{-1}) \times D_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{3.2}$$

$$\sum_{i \in \mathcal{N}} x_{1i} - \sum_{i \in \mathcal{N}} x_{i1} = 1 \tag{3.3}$$

$$\sum_{i \in \mathcal{N}} x_{Ni} - \sum_{i \in \mathcal{N}} x_{iN} = -1 \tag{3.4}$$

$$\sum_{i=2}^{N-1} x_{ij} - \sum_{i=2}^{N-1} x_{ji} = 0 \qquad j = 2, \ldots, N-1 \tag{3.5}$$

$$M \times (1 - x_{ij}) \geq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{3.6}$$

$$M \times (x_{ij} - 1) \leq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{3.7}$$

$$V_{ij} \leq U_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{3.8}$$

$$V_{ij} \geq L_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{3.9}$$

$$x_{ij} \times D_{ij} = V_{ij} \times t_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{3.10}$$

$$ea_1 = I \tag{3.11}$$

$$ec_N = 0 \tag{3.12}$$

$$ct_i = A \times y_i + B \times ec_i \qquad \forall i \in \mathcal{S} \tag{3.13}$$

$$ed_i = ea_i + ec_i \qquad \forall i \in \mathcal{N} \tag{3.14}$$

$$ed_i \leq C \qquad \forall i \in \mathcal{N} \tag{3.15}$$

$$ec_i \leq y_i \times (C - m) \qquad \forall i \in \mathcal{S} \tag{3.16}$$

$$y_j \leq \sum_{i=1}^{N-1} x_{ij} \qquad j = 2, \dots, N-1 \tag{3.17}$$

$$ea_j \leq \sum_{i=1}^{N-1} x_{ij} \times C \qquad j = 2, \dots, N-1 \tag{3.18}$$

$$ea_j \geq \sum_{i=1}^{N-1} x_{ij} \times m \qquad j = 2, \dots, N \tag{3.19}$$

$$T \geq \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} + \sum_{i=1}^{N} ct_j \tag{3.20}$$

$$\sum_{i=1}^{N} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{3.21}$$

$$V_{ij}, t_{ij}, E_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{3.22}$$

$$ec_i, ed_i, ea_i, ct_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{3.23}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{3.24}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{N} \tag{3.25}$$

The objective function includes the total energy consumption of the PEV. In other words, sum of energy consumption values on each arc $(i,j) \in \mathcal{A}$ is minimized in the objective function after the value of consumption on each road segment is determined.

Energy consumption of the PEV on each arc is provided in (3.2). It is calculated by multiplying the unit energy consumption with the total travel time for each arc (i,j). Moreover, we used the fact that distance over velocity gives the value of the travel time on each road segment. Therefore, the term with $V_{ij}^{(-1)}$ is written as tt_{ij} after it is multiplied by D_{ij} in inequality (3.2). It forms the first term in inequality (3.2). Remaining terms come from power function of a PEV defined in Chapter 3.1. They all are multiplied by D_{ij}. Total energy consumption on path is minimized by the help of this inequality in objective function (3.1).

Inequalities (3.3), (3.4) and (3.5) are classical shortest path constraints to choose a path over a network. Furthermore, constraints (3.6) and (3.7) give an energy balance between two nodes if the edge incident to these nodes is used. In other words, if PEV depletes its battery on an arc (i, j), the difference between energy levels of nodes i and j should be balanced. Moreover, inequalities (3.8) and (3.9) put limits on speed on each road segment (i, j). Constraint (3.10) establishes the relation between distance, velocity and travel time. If an arc is used, distance of it should be equal to multiplication of speed and travel time on this edge. Some initial and final conditions on energy levels of the battery are provided in constraints (3.11) and (3.12). If a node is reached from any node, energy level at arrival should be greater than m, energy level at departure should be lower than C, and there is no refueling at sink node. Inequality (3.13) computes the charging times. A represents the fixed time an electrical vehicle spends on charging. B is the term for unit time of recharging. By the help of these terms, an upper bound on charging time at each node i is provided. If y_i takes value of 0, charging amount, ec_i, will be equal to 0 for node i. As charging amount is equal to 0, charging times, ct_i, will be also 0.

If a PEV recharges itself at any node, energy level at the departure from node i should be equal to energy change at node i plus energy level at the arrival to node i. This balance equation is provided in inequality (3.14). Maximum battery level can be C while departing from a node i as stated in inequality (3.15). If a vehicle wants to recharge at a node i, it should be checked whether or not there is a recharging station at a node i. As stated before, C and m represent the maximum and minimum energy levels at the batteries. Hence, $(C - m)$ is the maximum amount of charging possible as enforced in equation (3.16). There is a linkage constraint (3.17) linking x_{ij} and y_i. Constraints (3.18) and (3.19) satisfy these criteria.

A limit on total charging time and travel time along the path is given in inequality

(3.20). Constraint (3.21) states that the path can have at most one arc between the nodes i and$j$. If an arc (i, j) is not used, its incident node cannot be used to stop. The remaining constraints (3.22), (3.23), (3.24) and (3.25) are sign and binary restrictions on decision variables.

In order to solve the PEVEEP-MINLP, we defined an MISOCP problem. SOCP problem is an optimization problem where we minimize a linear objective function over second order cone constraints of the from $||Ax+b|| \leq r^T x+d$. Linear constraints are second order cone constraints. Linear programs, convex quadratic programs and quadratically constrained convex quadratic programs are some of the problems which can be modeled as an SOCP problem [24]. SOCP problems are convex optimization problems and they can be solved in polynomial time.

MISOCP is an SOCP problem where some variables are restricted to take integer values. In our mathematical model, inequalities (3.2) and (3.10) are not linear. As all other constraints are second order cone constraints, we rewrote inequalities (3.2) and (3.10) using second order cone programming.

In order to make the energy consumption function conic, the terms V_{ij}^3 and V_{ij}^2 are represented as f_{ij} and l_{ij}, respectively. The term t_{ij}, travel time, corresponds to D_{ij}/V_{ij} and placed into inequality redefined. Conic representation of the inequality (3.2) can be seen below. This redefined inequality is replaced with (3.2).

$$E_{ij} = e \times t_{ij} + D_{ij} \times (b \times l_{ij} + a \times f_{ij} + c \times Vij + x_{ij} \times d)$$

Moreover, (3.10) is replaced with another redefined inequality. We wrote $l_{ij} \geq V_{ij}^2$ and $f_{ij} \times V_{ij} \geq l_{ij}^2$ implying $f_{ij} \geq V_{ij}^3$ for all $(i, j) \in \mathcal{A}$. First inequality is rewritten as:

$$\left\| \begin{matrix} V_{ij} \\ (l_{ij} - 1)/2 \end{matrix} \right\| \leq (l_{ij} + 1)/2 \quad \forall (i, j) \in \mathcal{A}$$

And second inequality is rewritten as:

$$\left\| \begin{matrix} l_{ij} \\ (f_{ij} - V_{ij})/2 \end{matrix} \right\| \leq (f_{ij} + V_{ij})/2 \quad \forall (i, j) \in \mathcal{A}$$

$l_{ij}, f_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}$

In addition to those, we wrote $x_{ij}^2 \times D_{ij} \leq V_{ij} \times t_{ij}$. It is rewritten as:

$$\left\| \begin{matrix} x_{ij} \times \sqrt{D_{ij}} \\ (t_{ij} - V_{ij})/2 \end{matrix} \right\| \leq (t_{ij} + V_{ij})/2 \quad \forall (i,j) \in \mathcal{A}$$

These are all SOCP constraints and the PEVEEP-MINLP is converted to an MISOCP model. Finalized version of the PEVEEP-MISOCP model can be seen in Appendix A.

CHAPTER 4

SOLUTION METHODOLOGY

In this chapter, we present 3 solution approaches for the PEVEEP. We first show that formulation (PEVEEP-MINLP) can be turned into a mixed integer second order cone programming (MISOCP) formulation. Then, we present a matheuristic and a VNS heuristic for the PEVEEP.

4.1 A Matheuristic Approach

Computational experiments show that the proposed MISOCP formulation is unable to solve large size instances optimally within reasonable times. Therefore, it is required to have heuristic approaches to be able to solve larger size instances. Matheuristics use the powerful sides of mathematical formulations and heuristics. Sometimes, model-based heuristics are used to define matheuristics since mathematical model of the problem is used for exploitation purposes in the heuristic [25]. On the other hand, metaheuristics offer the chance of have faster algorithms because of the simplicity in their definitions. In this chapter, a matheuristic is proposed for the PEVEEP combining three different mathematical models and neighborhood search methods to get good solutions in a faster way.

Based on experimental results, we have that an MISOCP determines values of energy levels, time passed and velocity of the vehicle in a shorter time on a previously determined path. In other words, MISOCP solutions are obtained much faster if a path is taken on rather than a network. Based on this information, a matheuristic approach is proposed to solve the PEVEEP. Models used, definitions and notation are provided next.

4.1.1 Definitions and Notation

In this thesis, a matheuristic that uses three different mathematical formulations is proposed. Algorithm uses MIP formulations over a network in order to construct an initial path. After a path is decided, MISOCP formulations are used to get values of other decision variables except x_{ij}s along a fixed path. Before the algorithm is constructed, two definitions are given.

Definition 1: A solution is energy feasible if energy requirement of an arc (i, j) is less than or equal to the difference between energy levels at node i and j, ed_i and ea_j, where ed_i must be less than C and ea_j must be greater than m. Otherwise, the solution is accepted as energy infeasible.

Definition 2: A solution is time feasible if total of travel times and charging times is less than or equal to T, and speed of each arc (i, j) is within the speed limits. Otherwise, the solution is accepted as time infeasible.

In this matheuristic, 3 mathematical programs are used. A problem of time optimization of a Plug – In Electric Vehicle (PEVTEPP - MIP) of a path is defined and a mathematical model is proposed. It is a mixed integer programming model that takes the velocity as a parameter and makes the initialization of the algorithm. As V_{ij}s are parameters, inequality (3.2) is not quadratic. Using the same constraints in Chapter 3, the model tries to find a path minimizing total travel and charging time over the network. There is no time limit on network so the problem tries to find a path where its total time spent is the minimum. For the remaining steps, algorithm uses 2 different optimization problems. First one is to make time improvements on a fixed path using a time minimization model where velocity is a decision variable (PEVTEPP – MISOCP) but x_{ij} is a parameter. Finally, a mathematical model to optimize energy usage along a fixed path (x_{ij} is a parameter) is defined (PEVFEEP – MISOCP). It determines velocity values on a previously constructed path under given time limit. Summary of these problems is in Table 4.1. Also, finalized versions of the mathematical models can be seen in Appendix B, C and D.

Table 4.1: Mathematical Models used in Matheuristic

Problem Name	**Problem Type**	**On a Fixed Path/Network**	**Time Limit Applied?**	**Is Velocity a DV/Parameter?**
PEVEEP-MISOCP	Energy Minimization	Network	Yes	DV
PEVTEPP - MIP	Total Time Minimization	Network	No	Parameter
PEVTEPP - MISOCP	Total Time Minimization	Fixed Path	No	DV
PEVFEEP – MISOCP	Energy Minimization	Fixed Path	Yes	DV

In addition to given in chapter 3.2.1, the following notation is used in the matheuristic approach.

k: Number of shortest paths constructed between a pair of nodes
K: Number of pairs of nodes chosen on a path
a: A counter in a while loop
b: A counter in a while loop

4.1.2 Algorithm Construction

As stated before, algorithm uses different optimization problems. Velocity of the vehicle is used as a parameter at initialization step. By this way, computational effort of the algorithm is tried to be lower. Matheuristic tries to exploit solutions using these optimization problems. The pseudocodes of our matheuristic algorithm is given in Algorithm 1. Also, in Figure 4.1 a decision tree for matheuristic algorithm with MIP Initialization is given.

The algorithm uses T-Improvement and E-Improvement algorithms. While doing this, it takes parameters k and K as given. Some other parameters of matheuristic algorithm are the same with the parameters used in PEVEEP-MISOCP, where N is the number of nodes on the network.

In step 3, PEVTEPP – MIP uses velocity as a parameter which is equal to maximum of lower bounds on each arc (i, j) and optimal V. This is the best case in terms of energy consumption over the network. As problem takes velocities as parameters, this model finds the solution faster than PEVEEP - MISOCP. After a path is found by PEVTEPP – MIP, energy feasibility is checked at step 5. If PEVTEPP – MIP is energy infeasible, algoritm terminates. This means that instance cannot be solved. If it is energy feasible, time feasibility is checked at step 8 and then problem continues to find improvements on that path.

Furthermore, if objective function value of PEVTEPP – MIP is less than or equal to deadline, problem is accepted as time feasible. In this case, E-Improvement is called in step 16 to minimize energy consumption on that path and decide velocities on each arc (i, j). E-Improvement algorithm tries to improve solution on hand in terms of energy consumption. If the problem is time infeasible at step 8, T-Improvement algorithm is called in order to have lower total time spent on the path than deadline. After a feasible path is constructed by T-Path improvement algorithm, E-Improvement algorithm is called in order to improve the solution on hand in terms of energy consumption at step 13.

If E-Path improvement algorithm is concerned, this algorithm tries to find more efficient path in terms of energy consumption over a network. A path is given to this algorithm by the help of x_{ij} values. While the condition is satisfied, algorithm chooses two nodes on that path in step 5. Then, k shortest paths between node i and j are listed in step 7. Yen's algorithm is an extremely efficient algorithm to list k shortest paths between the same origin and destination of a network [26]. This algorithm is used at each iteration to list k shortest paths between randomly chosen 2 nodes i, j.

After that, PEVFEEP - MISOCP is solved for all k paths. If one of the objective functions is lower than the current value, then the path is changed with the path with lower energy consumption. Until total number of iterations being equal to K, improvement continues. If an improvement is made at current loop, smaller loop is broken in step 12. Different i and j values are tried in further iterations. At the end, a path with the best objective function value is returned.

When T-Improvement algorithm is concerned, this algorithm tries to find a feasible path in terms of total time spent over a network. A path is given to this algorithm by the help of x_{ij} values. While the condition is satisfied, algorithm chooses two nodes on the path in step 5. Then, k shortest paths between nodes i and j is listed in step 5. Yen's algorithm is again used at each iteration to list the k shortest paths between randomly chosen 2 nodes i, j. After that, PEVTEPP - MISOCP is solved for all k

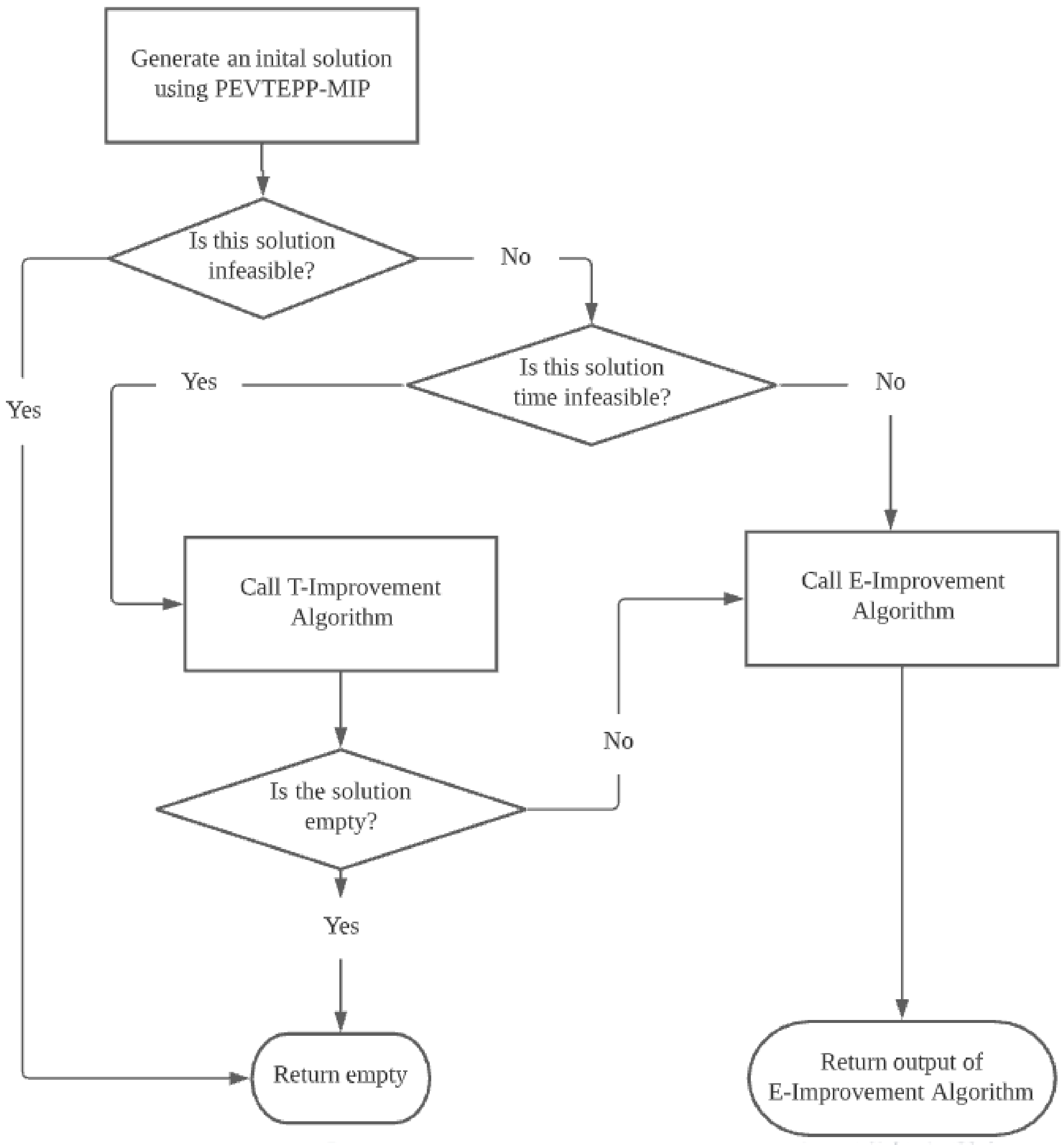

Generate an inital solution using PEVTEPP-MIP
Is this solution infeasible?
No
Yes
Is this solution time infeasible?
Yes
No
Call T-Improvement Algorithm
Call E-Improvement Algorithm
No
Is the solution empty?
Yes
Return empty
Return output of E-Improvement Algorithm

Figure 4.1: Matheuristic Algorithm with MIP Initialization

paths. If one of the objective functions is lower than the current value, then the path is changed with the path with lower time. If an improvement is made, both of two loops are broken in steps 12 and 18. The algorithm stops once a feasible path is found. If a feasible path whose total time of the path is less than or equal to T cannot be found, then algorithm returns empty. Otherwise, it returns the best path option in terms of energy.

Algorithm 1 Matheuristic for PEV

```
1: Input: All parameters in the PEVEEP-MINLP formulation
2: Output: Values of $Z, x_{ij}$, $y_i$, $E_{ij}$, $V_{ij}$, $ec_i$, $ea_i$, $ed_i$, $t_{ij}$, $ct_i$ that minimizes the energy consumption over the network
3: Call $PEVTEPP-MIP$ algorithm for the given network with fixed speeds $V_{ij}$ $\leftarrow$ max($V_{opt}$,$L_{ij}$).
4: Output of $PEVTEPP-MIP$: A path with $x_{PEVTEPP-MIP}$ and objective function value $z_{PEVTEPP-MIP}$.
5: if $PEVTEPP-MIP$ problem is infeasible then
6:     Terminate Matheuristic algorithm.
7: else
8:     if $PEVTEPP-MIP$ problem is time infeasible ($z_{PEVTEPP-MIP} > T$) then
9:         Call $T-Improvement$ algorithm for the path with previously decided $x_{PEVTEPP-MIP}$s.
10:        if $T-Improvement$ algorithm returns empty then
11:            Terminate Matheuristic algorithm.
12:        else
13:            Call $E-Improvement$ algorithm for the path with previously decided $x_{T-Improvement}$s.
14:        end if
15:    else
16:        Call $E-Improvement$ algorithm for the path with previously decided $x_{PEVTEPP-MIP}$s.
17:    end if
18: end if
19: Return output of $E-Improvement$ algorithm.
```

Algorithm 2 E-Improvement Algorithm for PEV

```
1: Input: All parameters in the PEVEEP-MINLP formulation, $k$, $K$, $x_{PEVTEPP-MIP}$ OR $x_{T-Improvement}$.
2: Output: Values of $Z_{E-Improvement}$, $y_i$, $E_{ij}$, $V_{ij}$, $ec_i$, $ea_i$, $ed_i$, $t_{ij}$, $ct_i$ that minimize the energy consumption over the path.
3: Call $PEVFEEP-MISOCP$ for given path. Objective function of the problem is equal to $Z_{PEVEEP-SOCP0}$. Let $x_{BEST-PEVEEP-SOCP} = x_{PEVEEP-SOCP0}$ and $Z_{BEST-PEVEEP-SOCP} = Z_{PEVEEP-SOCP0}$. $a = 0$.
4: while $a < K$ do
5:     Choose two nodes $i$ and $j$ randomly on the given path. List $k$ shortest paths from $i$ to $j$. $b = 0$.
6:     while $b < k$ do
7:         Call $PEVFEEP - MISOCP$ for reconstructed path with $x_{ij}^b$ where nodes between $i$ and $j$ are replaced with $b^{th}$ shortest path.
8:         Let $Z_{ij}^b$ be the corresponding value.
9:         if $Z_{ij}^b < Z_{BEST-PEVEEP-SOCP}$ then
10:            Change the path with the path from node $i$ to $j$ node.
11:            $Z_{BEST-PEVEEP-SOCP} = Z_{ij}^b$ and $x_{BEST-PEVEEP-SOCP} = x_{ij}^b$
12:            break
13:        end if
14:        $b = b + 1$.
15:    end while
16:    $a = a + 1$.
17: end while
18: Return $x_{BEST-PEVEEP-SOCP}$ and output of $PEVFEEP - MISOCP$ where last improvement is made.
```

Algorithm 3 T-Improvement Algorithm for PEV

```
1: Input: All parameters in the PEVEEP-MINLP formulation, $k$, $K$, $x_{PEVTEPP-MIP}$.
2: Output: Values of $Z_{T-Improvement}$, $y_i$, $E_{ij}$, $V_{ij}$, $ec_i$, $ea_i$, $ed_i$, $t_{ij}$, $ct_i$ that minimize the energy consumption over the path.
3: Call $PEVTEPP - MISOCP$ for given path. Objective function of the problem is equal to $Z_{PEVTEPP-MISOCP0}$. Let $x_{BEST-PEVTEPP-MISOCP} = x_{PEVTEPP-MISOCP0}$ and $Z_{BEST-PEVEEP-SOCP} = Z_{PEVEEP-SOCP0}$.
4: while $a < K$ do
5:     Choose two nodes $i$ and $j$ randomly on given path. List $k$ shortest paths from $i$ to $j$. $b = 0$.
6:     while $b < k$ do
7:         Call $PEVTEPP - MISOCP$ for reconstructed path with $x_{ij}^b$ where nodes between $i$ and $j$ are replaced with $b^{th}$ shortest path. Let $Z_{ij}^b$ be the corresponding objective function value.
8:         if $Z_{ij}^b < Z_{BEST-PEVTEPP-MISOCP}$ then
9:             Change the path with the path from node $i$ to $j$ node. $Z_{BEST-PEVTEPP-MISOCP} = Z_{ij}^b$ and $x_{BEST-PEVTEPP-MISOCP} = x_{ij}^b$
10:            break
11:        end if
12:        $b = b + 1$.
13:    end while
14:    if $Z_{BEST-PEVTEPP-MISOCP} \leq T$ then
15:        $feasibility = 1$
16:        break
17:    end if
18:    $a = a + 1$.
19: end while
20: if $feasibility = 1$ then
21:     Return $x_{BEST-PEVTEPP-MISOCP}$ and output of $PEVTEPP - MISOCP$ where last improvement is made.
22: else
23:     Return $\varnothing$
24: end if
```

4.2 A VNS Approach

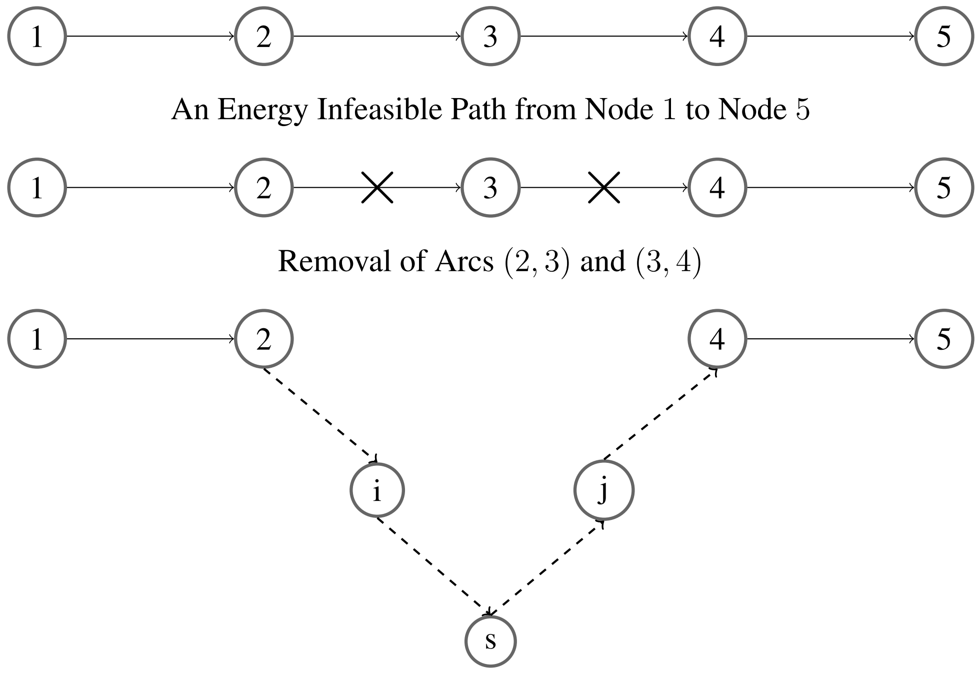
Variable Neighborhood Search method can be accepted as a meta-heuristic based on systematic changes of neighborhoods [27]. It tries to find a local minimum and a global minimum by perturbation techniques. VNS has been successfully applied to several non-linear and combinatorial optimization problems such as vehicle routing problems. The VNS method tries find local optima first using neighborhood structures defined. Then, exploitation is increased using these all different neighborhoods [27]. Once an incumbent solution is found, neighborhood structure is changed with another one. We next define the proposed VNS algorithm for the PEVEEP and introduce the related notation.

4.2.1 Definitions and Notation

Neighborhood definition is an important part of VNS algorithms as they provide the exploitation and exploration power of the VNS. They should be built as compatible with the objective function and lean to take solutions faster. Our problem has 3 main concerns: energy feasibility, time feasibility and energy optimization. In order to solve those, 3 different neighborhood structures are defined.

Station Insertion: This neighborhood definition partitions the set of nodes into two. S is used to represent station nodes over the network. This definition uses a local search method where a station node not on the decided path can be chosen from S and joined to the path using shortest path between nodes i, j and station s which are randomly chosen. This search continues until total number of iterations is reached or all arrival energy levels to all nodes on the path is greater than or equal to m, in other words energy feasibility is provided.

Furthermore, this search uses a parameter which is $TotalEA$ which is equal to sum of arrival energy levels to all nodes on the path, ea_j, which are less than or equal to m. If they are nonpositive, their absolute values are taken for calculations. It can be said that a path is energy feasible if $TotalEA = 0$. Steps of this search can be seen in Algorithm 6.


1
2
3
4
5
An Energy Infeasible Path from Node 1 to Node 5
Removal of Arcs $(2, 3)$ and $(3, 4)$
i
j
s

Connection of a Station Node $s \in S$ to the Path using unvisited Nodes i and $j \in N$

Figure 4.2: An Illustration of Station Insertion Algorithm

An illustration of Station Insertion is shown in Figure 4.2. There is an example path where node 1 is the source node, and node 5 is the destination node. The path between them is energy infeasible, where $TotalEA > 0$. Hence, Algorithm 6 tries to add a station node to the path in order to increase energy levels, ea and ed, at arrival and departure. First of all, algorithm starts with choosing 2 nodes on the path. Assuming that these nodes are 2 and 4, road segment between them is removed. Then, station $s \in N$ is chosen randomly. New path is constructed by the help of shortest paths between node s and nodes 2, 4. Dijkstra's algorithm works for the edges with positive costs proposed by Dijkstra in [28]. It is an efficient algorithm in terms of solution quality and time. It is used at every iteration until energy feasibility is provided. If the connection between s and chosen nodes cannot be provided, then algorithm starts from beginning by choosing a new random pair. Else, energy feasibility of the path is checked in the next step.

Replacement of Expensive Nodes: This neighborhood definition tries to make energy consumption lower on the path by exchanging road segments. It uses a local search method where two nodes are randomly chosen and the road segment between them is removed from the path, the chosen nodes are then reconnected using shortest path between them. Dijkstra's algorithm is again used at every iteration to find the shortest path between two randomly chosen nodes. This search continues until total number of iterations is reached. If an energy improvement is made, solution on hand jumps into new one. At the same time, sum of arrival energy levels of all nodes on the path is greater than or equal to the previous one, in other words new path should be closer to be energy feasibility than the best solution on hand. Also, total time spent on the path should be lower than T in order to keep the path time feasible. Steps of this search can be seen in Algorithm 7.

An illustration of Station Insertion is shown in Figure 4.3. In the figure, an example path is given where node 1 is the source node, and node 5 is the destination node. The path between them may be time or energy infeasible. Feasible paths may also be subject to replacement of expensive nodes to have an improvement on energy consumption. Algorithm 7 tries to change road segment between two randomly chosen nodes. First of all, algorithm starts with choosing a pair of nodes. Assuming that these nodes are 2 and 4, road segment between them is reconstructed by the help of shortest path between nodes 2 and 4. If energy consumption of the new path is larger than the previous one or energy infeasible, then algorithm starts from beginning with the path on hand by choosing a new random pair. Else, reconstructed path is accepted as the new solution on which improvements continue to be made.

Short Cut: This neighborhood definition benefits from short distances of direct edges to make energy consumption lower on the path by exchanging road segments with direct edges between two nodes, randomly chosen. This search continues until total number of iterations is reached or there is no direct edge between any two nodes on the path. If an energy improvement is made, solution on hand jumps into the new one. At the same time, sum of arrival energy levels of all nodes on the path is greater than or equal to the previous one, in other words new path should be closer to energy feasibility than the best solution on hand. Also, total time spent on the path should be lower than T in order to keep the path time feasible. Steps of this search can be seen at Algorithm 8.

An illustration of Short Cut is shown in Figure 4.4. Here, an example path is shown

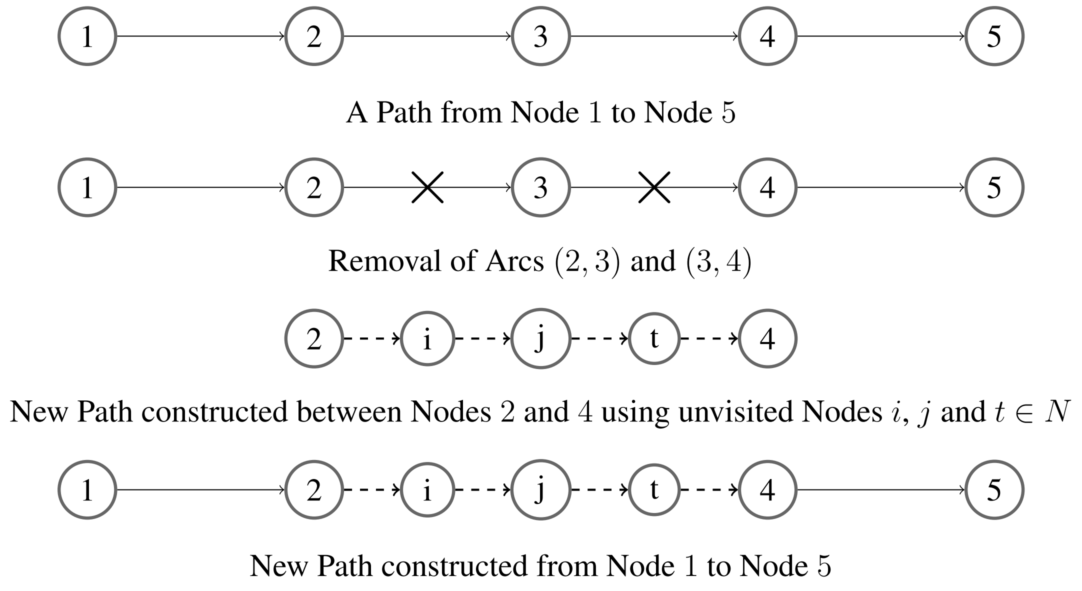
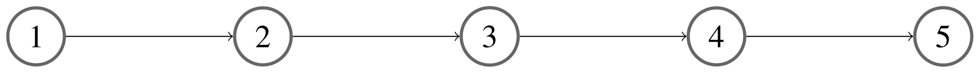
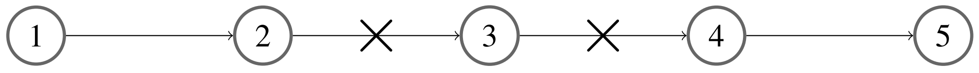

1
2
3
4
5
A Path from Node 1 to Node 5
Removal of Arcs (2, 3) and (3, 4)
i
j
t
New Path constructed between Nodes 2 and 4 using unvisited Nodes i, j and t ∈ N
New Path constructed from Node 1 to Node 5

Figure 4.3: An Illustration of Replacement of Expensive Nodes Algorithm

where node 1 is the source node, and node 5 is the destination node. All types of paths in terms of feasibility may again be subject to this algorithm to have an improvement on energy consumption. Algorithm 8 tries to change road segment between two randomly chosen nodes. First of all, algorithm starts with choosing a node on the path. Assuming that this node is 2, $Adjacents$ set is constructed from the nodes where there is a direct edge between node 2 and other nodes on the path. Then, a node from this set is chosen randomly. This node is 4 in this case. Road segment between these nodes is replaced with arc $(2, 4)$. If energy consumption of new path is larger than the previous one or energy infeasible, then algorithm starts from beginning with the path on hand by choosing a new random node. Else, reconstructed path is accepted as the new solution on which improvements continue to be made.

In addition to the notation in chapter 3.2.1, notation used in VNS approach is as follows:
x': An inherited adjacent matrix from the previous step at VNS algorithm
I: A predetermined value where a pair of nodes can be chosen on the path to replace
K: A predetermined value where a pair of nodes can be chosen on the path to replace
$iteration$: A counter in a while loop
t: A counter in a while loop
v: A counter in a while loop


A Path from Node 1 to Node 5


Removal of Arcs $(2, 3)$ and $(3, 4)$

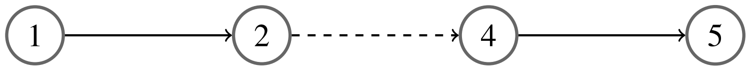

New Path constructed from Node 1 to Node 5

Figure 4.4: An Illustration of Short Cut Algorithm

4.2.2 Algorithm Construction

The pseudo code of the proposed VNS is given in Algorithm 4. Also, in Figure 4.5 a decision tree for VNS algorithm is shown.

VNS algorithm starts with a random initialization technique which is explained in Algorithm 5. This procedure starts from the origin node 1, and finishes at the destination node N. At each step two nodes are joined using their total distances to N. There is an adjacent set involving the nodes whose distances to N is lower than the distance of the current node to N. A node from the adjacent set of the current node is chosen randomly and an arc between them is constructed. At the end, a connected path starting from 1 and ending at N is obtained. After initial path is constructed in step 3 of Algorithm 4 , energy consumption on this path is calculated without looking for any feasibility, time or energy. Until total number of iterations reaches I, algorithm uses 3 different local search methods: Station Insertion, Replacement of Expensive Nodes and Short Cut. Algorithm starts with Station Insertion as this method increases the number of nodes on the path while connecting a station to the current path. In other words, VNS can explore more different areas thanks to starting with this insertion procedure. Also, this search tries to provide energy feasibility. Also, first concern should be thought during the algorithm construction is energy feasibility. If path is energy feasible ($TotalEA \geq 0$) or stopping criteria is satisfied in Algorithm 6, then the VNS algorithm continues with Replacement of Expensive Nodes.

Replacement of Expensive nodes takes the path which is the result of Station Insertion as an input. Then, two nodes on the path are chosen randomly. The road segment (i, j) is changed with the shortest path between them. New path constructed at each step is compared with the solution on hand in terms of their energy consumption values. At each step, in order to keep the path time feasible and energy feasible, $TotalT$ and $TotalEA$ values are also checked. If the condition in step 10 in algorithm 7 is satisfied, the new path is accepted as the best solution. Then, algorithm continues until total number of iterations reaches to K.

Finally in step 8, Short Cut method is used to provide energy improvement on the path resulted by Replacement of Expensive Nodes. This method searches for a set of nodes

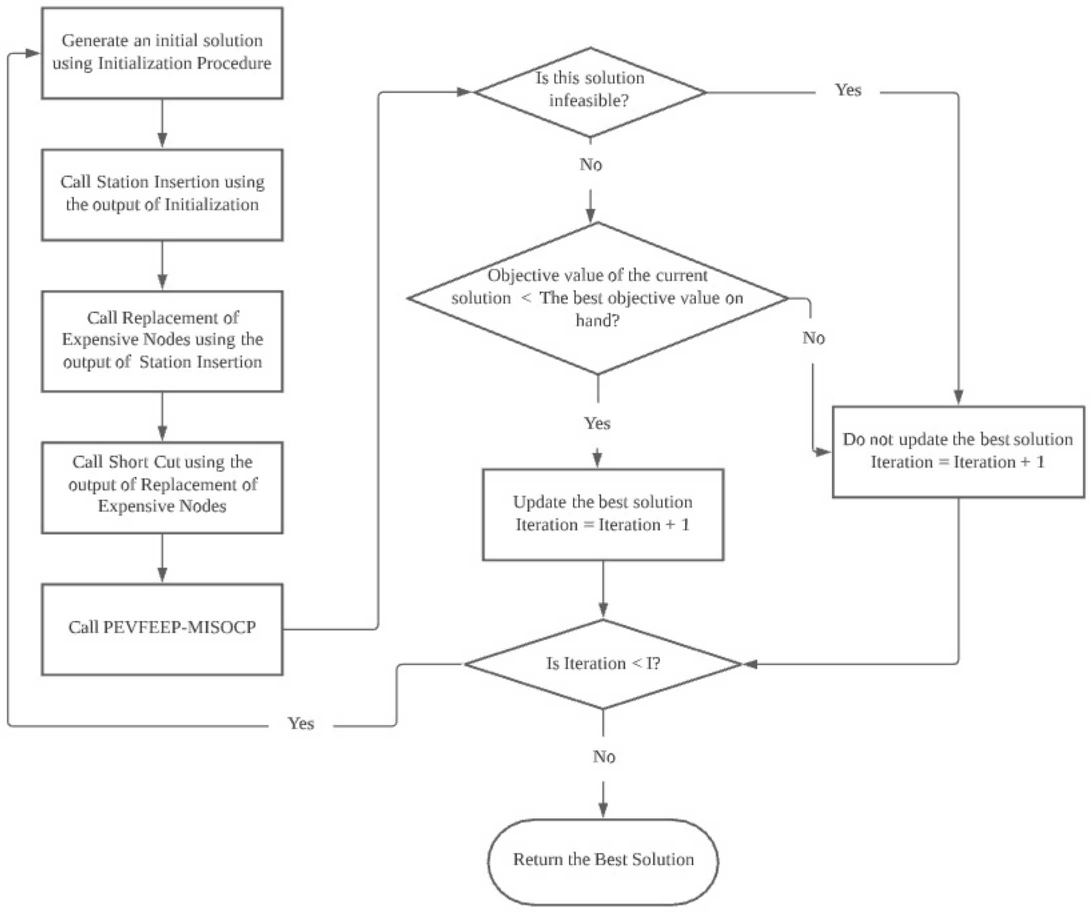

Generate an initial solution using Initialization Procedure
Call Station Insertion using the output of Initialization
Call Replacement of Expensive Nodes using the output of Station Insertion
Call Short Cut using the output of Replacement of Expensive Nodes
Call PEVFEEP-MISOCP
Is this solution infeasible?
Yes
No
Objective value of the current solution < The best objective value on hand?
No
Yes
Do not update the best solution Iteration = Iteration + 1
Update the best solution Iteration = Iteration + 1
Is Iteration < I?
Yes
No
Return the Best Solution

Figure 4.5: VNS Algorithm

where there is at least one direct edge between them. If this set is empty, algorithm 8 terminates. Otherwise, it continues until total number of iterations reaches to K.

After all local search methods are completed, the PEVFEEP-MISOCP is called to determine $TotalEnergyConsumption$, x_{ij}, y_i, E_{ij}, V_{ij}, ec_i, ea_i, ed_i, t_{ij}, ct_i where all $i, j \in N$ that minimizes the energy consumption for the selected path.

Algorithm 4 VNS Algorithm for PEV

```
1: Input: All parameters in the PEVEEP-MINLP formulation, $I$.
2: Output: Values of $Z$, $x_{ij}$, $y_i$, $E_{ij}$, $V_{ij}$, $ec_i$, $ea_i$, $ed_i$, $t_{ij}$, $ct_i$ that minimizes the energy consumption over the path.
3: Call $InitializationProcedure$. Let initial path constructed be named as $x_0$ and its total energy consumption as $Z_0$.
4: $x_{BEST-VNS} = x_0$ and $Z_{BEST-VNS} = Z_0$.
5: while $Iteration \leq I$ do
6:     Call $StationInsertion$ for given path $x_0$. Energy consumption of the decided path is equal to $Z_{SI}$, and the path decided by this neighborhood is $x_{SI}$.
7:     Call $ReplacementofExpensiveNodes$ for given path $x_{SI}$. Energy consumption of the decided path is equal to $Z_{RON}$, and the path decided by this neighborhood is $x_{RON}$.
8:     Call $ShortCut$ for given path $x_{RON}$. Energy consumption of the decided path is equal to $Z_{SC}$, and the path decided by this neighborhood is $x_{SC}$.
9:     Call $PEVFEEP - MISOCP$ for given path $x_{SC}$. Objective function of the problem is equal to $Z_{CURR}$.
10:    if $Z_{CURR} \leq Z_{BEST-VNS}$ then
11:        $x_{BEST-VNS} = x_{SC}$ and $Z_{BEST-VNS} = Z_{CURR}$.
12:    end if
13:    $Iteration = Iteration + 1$.
14: end while
15: Return $x_{BEST-VNS}$ and output of $PEVFEEP - MISOCP$ where last improvement is made.
```

Algorithm 5 Initialization Procedure

```
1: Input: $D$, $N$.
2: Output: Value of $x$ representing a path from 1 to $n$.
3: $i = 1, x = ZeroMatrice.$
4: Let $d_i$ be the shortest distances constructed using $D$ from each node representing total length of the path from node $i$ to node $N$.
5: $Adjacents_i$ is the number of neighbors including any node $j$ which are satisfying inequality $d_j \leq d_i$.
6: while $i \neq N$ do
7:     Construct the set of $Adjacents_i$. Make an ascending order of edges and give ordered numbers. $r = randbetw(0, 1)$.
8:     $p = 1/|Adjacents_i|$. Find the $k$ where $(k + 1)p$ is higher than $r$ for the first time, $r \leqslant (k + 1)p$. Then, edge $j$ is chosen by corresponding order of $(k + 1)$.
9:     $i = j, x_{ij} = 1$
10: end while
11: Return $x$.
```

Algorithm 6 Station Insertion Algorithm

```
1: Input: All parameters in the PEVEEP-MINLP formulation, $x_0$, $K$.
2: Output: Value of $x$ representing a path from 1 to $n$.
3: Calculate $TotalEA_0$ which is equal to sum of all arrival energy levels to node $i, ea_i, \in N$ of the initial path.
4: $x_{BEST-SI} = x_0$ and $TotalEA_{BEST-SI} = TotalEA_0$.
5: while $iteration \leq K$ do
6:     if $TotalEA_{BEST-SI} > 0$ then
7:         break
8:     end if
9:     Choose two nodes $i$ and $j$ randomly on given path. Choose a station node $s \in S$ randomly.
10:    First connect node $i$ to station $s$, then connect node $s$ to node $j$ using shortest paths between them. Construct a path whose adjacent matrix is $x_{iteration}$.
11:    Calculate $TotalEA_{iteration}$ which is equal to sum of arrival energy levels to node $i \in N$.
12:    if $TotalEA_{BEST-SI} < TotalEA_{iteration}$ then
13:        $x_{BEST-SI} = x_{iteration}$ and $TotalEA_{BEST-SI} = TotalEA_{iteration}$.
14:    end if
15:    $iteration = iteration + 1$.
16: end while
17: Return $x_{BEST-SI}$.
```

Algorithm 7 Replacement of Expensive Nodes Algorithm

```
1: Input: All parameters in the PEVEEP-MINLP formulation, $x_{SI}$, $K$.
2: Output: Value of $x$ representing a path from 1 to $n$.
3: Calculate $TotalEA_0$ sum of arrival energy levels to node $i \in N$, $TotalEnergy_0$ total energy consumed, $TotalT_0$ total time spent of the initial path.
4: $x_{BEST-RON} = x_{SI}$ and $TotalEnergy_{BEST-RON} = TotalEnergy_0$, $TotalT_{BEST-RON} = TotalT_0$, $TotalEA_{BEST-RON} = TotalEA_0$.
5: while $iteration \leq K$ do
6:     Choose two nodes $i$ and $j$ randomly on given path.
7:     Connect node $i$ to node $j$ using shortest path between them. Construct a path whose adjacent matrix is $x_{iteration}$.
8:     Calculate $TotalEA_{iteration}$ sum of arrival energy levels to node $i \in N$, $TotalEnergy_{iteration}$ total energy consumed, $TotalT_{iteration}$ total time spent of the constructed path.
9:     if $TotalEA_{BEST-RON} < TotalEA_{iteration}$ and $TotalEnergy_{iteration} < TotalEnergy_{BEST-RON}$ and ($TotalT_{iteration} < TotalT_{BEST-RON}$ or $TotalT_{iteration} < T$) then
10:        $x_{BEST-RON} = x_{iteration}$ and $TotalEnergy_{BEST-RON} = TotalEnergy_{iteration}$, $TotalT_{BEST-RON} = TotalT_{iteration}$, $TotalEA_{BEST-RON} = TotalEA_{iteration}$.
11:    end if
12:    $iteration = iteration + 1$.
13: end while
14: Return $x_{BEST-RON}$.
```

Algorithm 8 Short Cut Algorithm

```
1: **Input:** All parameters in the PEVEEP-MINLP formulation, $x_{RON}$, $K$.
2: **Output:** Value of $x$ representing a path from 1 to $n$.
3: Calculate $TotalEA_0$ sum of arrival energy levels to node $i \in N$, $TotalEnergy_0$ total energy consumed, $TotalT_0$ total time spent of the initial path.
4: $x_{BEST-SC} = x_{RON}$ and $TotalEnergy_{BEST-SC} = TotalEnergy_0$ , $TotalT_{BEST-SC} = TotalT_0, TotalEA_{BEST-SC} = TotalEA_0$.
5: while $iteration \leq K$ do
6:     while $t \neq N$ do
7:         for $v = 1$ to $N$
8:         if $x_{BEST-SC}(t, v) \geq 1$ where $v \neq t + 1$ then
9:             $v$ is in a set of nodes $Adjacents$ where there is a direct edge between $t$ and $v$.
10:        end if
11:        end for
12:        $t = t + 1$.
13:    end while
14:    if $Adjacentsisempty$ then
15:        break
16:    else
17:        Choose a node $i$ randomly on given path. Connect node $i$ to node $j$ using direct edge between them where $j \in Adjacents$ is chosen randomly. Construct a path whose adjacent matrix is $x_{iteration}$.
18:        Calculate $TotalEA_{iteration}$ sum of arrival energy levels to node $i \in N$, $TotalEnergy_{iteration}$ total energy consumed, $TotalT_{iteration}$ total time spent of the constructed path.
19:        if $TotalEA_{BEST-SC} < TotalEA_{iteration}$ and $TotalEnergy_{iteration} < TotalEnergy_{BEST-SC}$ and ($TotalT_{iteration} < TotalT_{BEST-SC}$ or $TotalT_{iteration} < T$) then
20:            $x_{BEST-SC} = x_{iteration}$ and $TotalEnergy_{BEST-SC} = TotalEnergy_{iteration}$ , $TotalT_{BEST-SC} = TotalT_{iteration}, TotalEA_{BEST-SC} = TotalEA_{iteration}$.
21:        end if
22:    end if
23:    $iteration = iteration + 1$.
24: end while
25: Return $x_{BEST-SC}$.
```

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

In this chapter, computational results of the solution methodologies proposed in Chapter 4 are presented. Random problem instances are created using MATLAB, and algorithms are tested using the library CPLEX 12.10.0 via C++. Test environment is an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 8GB RAM Windows 10 PC.

Solutions of the Exact Algorithm, Matheuristic Approach and VNS heuristic are compared in terms of their computational times. Moreover, effect of instance parameters on computational times for each algorithm is discussed. Instances taken from the literature and the randomly generated instances are discussed in Chapter 5.1. Preliminary experiments to set some parameters of the algorithm are detailed in Chapter 5.2. Discussions on computational times based on instance parameters and algorithm properties are presented in Chapter 5.3.

5.1 Instance Generation

During our literature review, we recognized that speed optimization on PEVs and finding an energy efficient path for PEVS are not studied at the same time. As it is our contribution to literature, there is no available benchmark instances defined for the PEVEEP. Hence, we used available network instances defined for another problem which is defined as the mixed capacitated arc routing problem by Belenguer and Gouevia et al. [29], [30]. These instances are publicly accessible. All the graphs defined here are incomplete, and the structure fits to minimum cost path problems. Also, a cost on each arc is defined. Graph is directed, and there can be no arc defined as (j, i) while there is an arc (i, j). We used these costs as the distances which is

D. Capacities defined by Belenguer and Gouevia et al. are not used in our problem. Various graphs are constructed here in terms of number of nodes and arcs. This also enriches our computational analysis.

These instances are modified for our problem definition. On these networks, we have introduced lower and upper speed limits for each arc. A time limit is also set. A PEV, with parameters defined in Chapter 3.1, is used in all instances. In addition to those, some nodes are defined as stations where a PEV can recharge its batteries by stopping at these nodes. Maximum and minimum battery capacities, which are C and m, of PEV are also determined. An initial energy level, I, is given to PEV to start its path. Some of these added parameters are decided using MATLAB. Others are defined during preliminary experiments. A detailed explanation on instance generation is given in Table 5.1.

For all sizes, lower and upper speed limits on each arc are generated randomly. L follows uniform distribution between 40 and 70 km/h, and U follows uniform distribution between 80 and 120 km/h for each arc. We put these limits considering real life instances. The values are acceptable for urban-roads. Also, energy consumption function of the PEV, Figure 3.1, has optimal energy level at V equals to 46.926 km/h. If an arc has a lower L value than optimal V, then the best speed value in terms of energy consumption is the optimal V. On the other hand, if an arc has a higher L value than optimal V, then the best speed is L.

Values of m, C and I are determined based on lower speed limits and distances of the arcs. In order to have an energy feasible path, I should be more than the total energy consumption until first station node where PEV can stop and recharge the battery. m and C decided with respect to I value. During preliminary experiments, these determined values give feasible paths for PEV. Therefore, they are used in all instances whose sizes are from 24 to 321 nodes. Likewise, T is decided so that there can be a time feasible path while algorithms try to solve the problem. Total charging time and total travel time of PEV from node 1 to node N is a respectable amount. T is determined based on preliminary results of each instance. Graphs whose sizes greater than 195 needs larger amount of time which is equal to 5×10^6 sec.

A station pattern for each graph is constructed using uniform distribution. In order to analyze the effect of number of stations on solutions, we defined different station densities for each node size, which are 20%, 30% and 40%. A node is a station node with a probability of 0.2 if station density is equal to 20% for a graph. Probabilities are 0.3 and 0.4 if station densities are 30% and 40% respectively. At the end, we expect totally Station Density $\times$ $|N|$ stations on hand. For instance, for the graphs whose node sizes are 41, there should be about 12 randomly chosen stations if station density is 30%.

In addition to those instances, a graph whose node size is equal to 40 is created where it has two node clusters with different properties. First cluster includes arcs whose lengths are uniformly distributed between 15 and 20 km. This cluster is assumed to be in a rural area whose road segments are shorter and not appropriate to drive faster. Hence, L follows a uniform distribution between 30 and 50 km/h, and U follows a uniform distribution between 60 and 90 km/h. Also, there is no electric recharge station in this area. Second cluster is assumed to be in an urban area whose road segments are longer and appropriate to drive faster. In this case, L follows a uniform distribution between 60 and 80 km/h, and U follows a uniform distribution between 90 and 120 km/h. Stations are available at some nodes, and distance of each road segment is uniformly distributed between 40 and 60 km. These two areas connected by the help of bridges defined between last 6 nodes of the 1^{st} cluster and first 6 nodes of the 2^{nd} cluster. These arcs have the same properties with the arcs in the 1^{st} cluster so that a vehicle can reach a station by the help of lower L and D values. Moreover, we used different m value than in the Table 5.1 which is equal to 10^8 joule. As C and I values are dependent to m, their values are also changed. In order to analyze this graph, we just used 40% density level.

In Chapter 5, computational experiments are presented using node sizes 24, 41, 50, 146, 195 and 321. Also, the PEVEEP is solved on the graph constructed by us to discuss the effect of graph structure.

Table 5.1: Summary of Parameters Generated Based on Number of Nodes over a Network

Parameter	**|N|=24**	**|N|=41**	**|N|=50**	**|N|=146**	**|N|=195**	**|N|=321**
L	U[40,70] km/h	U[40,70] km/h	U[40,70] km/h	U[40,70] km/h	U[40,70] km/h	U[40,70] km/h
U	U[80,120] km/h	U[80,120] km/h	U[80,120] km/h	U[80,120] km/h	U[80,120] km/h	U[80,120] km/h
m	1.5×10^8	1.5×10^8	1.5×10^8	1.5×10^8	1.5×10^8	1.5×10^8
C	$2 \times m$	$2 \times m$	$2 \times m$	$2 \times m$	$2 \times m$	$2 \times m$
I	$1.5 \times m$	$1.5 \times m$	$1.5 \times m$	$1.5 \times m$	$1.5 \times m$	$1.5 \times m$
T	$27.778\ h\ (10^6 sec)$	$27.778\ h\ (10^6 sec)$	$27.778\ h\ (10^6 sec)$	$27.778\ h\ (10^6 sec)$	$138.889\ h\ (5 \times 10^6 sec)$	$138.889\ h\ (5 \times 10^6 sec)$
S	Randomly selected nodes 20%, 30% and 40% of N	Randomly selected nodes 20%, 30% and 40% of N	Randomly selected nodes 20%, 30% and 40% of N	Randomly selected nodes 20%, 30% and 40% of N	Randomly selected nodes 20%, 30% and 40% of N	Randomly selected nodes 20%, 30% and 40% of N

5.2 Preliminary Experiments

In this thesis, we used two different heuristic approaches and an exact solution method. By changing parameters while taking runs using heuristic methods, effects of them on the solution quality and CPU time are analyzed.

For the matheuristic approach, two parameters are used which are k and K. We keep the values of these parameters same both subroutines, E-Improvement and T-Improvement. As they work with same logic in terms of construction of new paths, it is logical to keep the parameters same. K is the total number of random pairs chosen to make path exchanges between them. It offers to try various pair of i and j values which increases the probability of reaching a good solution. If it is too large, solution quality may be better, but computational times become high. Hence, it is important to choose K value large enough providing both good solution quality and reasonable computational times. On the other hand, there is a parameter k which is used for number of shortest paths constructed between chosen pair of nodes i and j. Algorithm chooses k^{th} shortest path from the beginning at each trial. If path exchange using k^{th} shortest path improves the objective function, then loop is broken in order to work with a new pair of nodes. Using different values of k and K, different instances with different patterns are tried. Graph sizes are chosen as medium size instances to make a reasonable comparison, 24 and 50. The representation $(|N| - Density - Pattern)$ is used for explanation of instances.

There are 3 tables containing summary information about Matheuristic which are Tables 5.2, 5.3 and 5.4. Matheuristic starts with PEVTEPP-MIP algorithm, then tries to provide feasibility and improvement on the initial solution. In Table 5.2, k and K are taken as 1, and algorithm is tried on instances whose sizes are 24 and 50. Sixth column represents whether the algorithm finds the optimal solution given by PEVEEP-MISOCP. While k and K equals to 1, only optimal solutions of the graph $(50 - 20 - 1)$ and $(50 - 20 - 2)$ are found. Solution times range between 0.475 and 4.378 sec. In this situation, value of k is increased to 5 in order to have better solution quality. This means that Algorithms 2 and 3 can try other paths a little bit longer than the shortest path. However, solution quality is not changed and solution times are worse in Table 5.3 than in Table 5.2. Therefore, K is increased 5 so that we can

Table 5.2: Preliminary Experiments for Matheuristic while $k = 1$ and $K = 1$

k	K	**|N|**	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap (%)**	**Solution Time (sec)**
1	1	24	20	1	-	10.327	0.475
			20	3	-	10.327	0.678
			40	2	-	10.327	0.691
			40	3	-	10.327	0.792
		50	20	1	+	0	4.378
			20	2	+	0	2.162
			30	1	-	1.800	3.490
			30	3	-	0.207	3.320

have better algorithm settings. Then, algorithms try more number of pairs randomly chosen. In other words, exploration increases by the help of the parameter set $k = 5$ and $K = 5$. Solution quality and solution times of the algorithm is presented in Table 5.4. It is seen that all the solutions for these chosen networks are optimal. It can also be said that algorithm is still fast as it runs in most 21.724 $seconds$ for the network $(50 - 20 - 1)$. It is an expected result in terms of both solution quality and computational times because number of total trials increased compared to previous parameter selections. So, parameter selection for Matheuristic, Algorithm 1, is made as $k = 5$ and $K = 5$.

In addition to matheuristic, parameters are also decided for the VNS algorithm. VNS, Algorithm 4 uses parameter I. Also, neighborhood searches, Algorithms 6, 7 and 8, uses K. VNS algorithm starts with an initial path decided by the algorithm 5. Then, totally I iterations are used for restart. At each iteration, improvement is made K times on a different initial path. Based on preliminary observations, it is seen that Replacement of Expensive Nodes definition makes better improvements compared to other neighborhood searchs. Hence, we defined another K, which is K_{RON}, to decide number of pairs randomly chosen on a path to make improvements.

In Table 5.5, K_{RON} and K are accepted as 5, and algorithm is tried on instances

Table 5.3: Preliminary Experiments for Matheuristic while $k = 5$ and $K = 1$

k	K	**|N|**	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap** (%)	**Solution Time** (sec)
5	1	24	20	1	-	10.327	1.483
			20	3	-	10.327	1.404
			40	2	-	10.327	2.079
			40	3	-	10.327	1.577
		50	20	1	+	0	8.417
			20	2	+	0	6.424
			30	1	-	1.800	6.198
			30	3	-	0.207	5.521

Table 5.4: Preliminary Experiments for Matheuristic while $k = 5$ and $K = 5$

k	K	**|N|**	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap** (%)	**Solution Time** (sec)
5	5	24	20	1	+	0	5.834
			20	3	+	0	4.951
			40	2	+	0	5.769
			40	3	+	0	6.230
		50	20	1	+	0	21.724
			20	2	+	0	19.438
			30	1	+	0	17.803
			30	3	+	0	18.088

Table 5.5: Preliminary Experiments for VNS heuristic while $K_{RON} = 5$ and $K = 5$

K_{RON}	K	|N|	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap (%)**	**Solution Time (sec)**
5	5	24	20	1	+	0	0.515
			20	3	+	0	0.472
			40	2	+	0	0.504
			40	3	+	0	0.520
		50	20	1	-	43.868	1.601
			20	2	+	0	1.291
			30	1	+	0	1.129
			30	3	-	32.539	2.352
		146	20	1	-	111.358	14.506
			30	1	-	79.932	10.485
			50	1	-	5.342	7.779

whose sizes are 24, 50 and 146. I is equal to 1 to make comments on other parameters first. Instances with node size 146 are also added to see effect of parameter selection on solution quality and solution time better. For this parameter selection, optimal solutions are found for 6 of 11 instances. Solution times range between 0.472 and 14.506 sec. Then, K_{RON} is accepted as 15 so that Replacement of Expensive nodes can make more number of searches to have better solution quality. As it can be seen in Table 5.6, same optimal solutions are found while solution time increases. Also, an extra optimal solution is found for $(146 - 50 - 1)$. Moreover, for the solutions not optimal, optimality gap becomes smaller for this parameter selection.

For these levels of K_{RON} and K, I is set different values, 3 and 5. Algorithm starts from beginning at each iteration up to I. As total number of trials is increased, solution times are a little bit higher for the values of I, 3 and 5. In Tables 5.7 and 5.8, solutions of VNS algorithm can be seen. While I is equal to 5, optimal solutions are found for all instances except the instance $(146 - 30 - 1)$. Furthermore, algorithm still has reasonable solution times compared to Matheuristic results in Table 5.4. Thus, K_{RON}, K and I are accepted as 15, 5 and 5, respectively.

Table 5.6: Preliminary Experiments for VNS heuristic while $K_{RON} = 15$ and $K = 5$

K_{RON}	K	|N|	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap (%)**	**Solution Time (sec)**
15	5	24	20	1	+	0	1.121
			20	3	+	0	1.280
			40	2	+	0	1.114
			40	3	+	0	1.139
		50	20	1	-	43.868	3.690
			20	2	+	0	3.467
			30	1	+	0	3.310
			30	3	-	28.290	4.331
		146	20	1	-	107.771	32.626
			30	1	-	15.371	24.910
			50	1	+	0	22.453

Table 5.7: Preliminary Experiments for multistart ($I = 3$) VNS heuristic while $K_{RON} = 15$ and $K = 5$

K_{RON}	K	|N|	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap (%)**	**Solution Time (sec)**
15	5	24	20	1	+	0	5.795
			20	3	+	0	5.385
			40	2	+	0	5.191
			40	3	+	0	5.390
		50	20	1	-	19.342	26.781
			20	2	+	0	17.318
			30	1	+	0	18.556
			30	3	-	44.980	23.877
		146	20	1	-	2.478	102.553
			30	1	-	2.478	81.838
			50	1	+	0	73.830

Table 5.8: Preliminary Experiments for multistart ($I = 5$) VNS heuristic while $K_{RON} = 15$ and $K = 5$

K_{RON}	K	|N|	**Density**	**Pattern**	**Is the Best Solution Found?**	**Optimality Gap (%)**	**Solution Time (sec)**
15	5	24	20	1	+	0	8.772
			20	3	+	0	9.077
			40	2	+	0	8.340
			40	3	+	0	8.986
		50	20	1	+	0	21.646
			20	2	+	0	23.642
			30	1	+	0	22.563
			30	3	+	0	22.754
		146	20	1	+	0	186.683
			30	1	-	2.478	158.006
			50	1	+	0	137.856

5.3 Computational Results

In this chapter, we analyzed instances with different sizes and densities. In total, 6 different network size and 3 different densities for each network size are used. As mentioned in Chapter 5.1, stations are randomly assigned at each network. So, station pattern is another factor on objective function values and solution times. In this study, it is aimed to make observations on computational times of two different algorithms proposed and the exact method PEVEEP-MISOCP.

PEV properties are accepted as the same for all instances. Velocity remains the same on each arc once it is decided. L and U values are the same for networks with same node size. It does not change for the network while its pattern and station density is changed. First of all, PEVEEP-MISOCP problem is solved using the mathematical model and algorithm proposed in Chapters 3 and 4. For each instance, exact solutions and solution times are in Tables 5.9, 5.10, 5.11 and 5.12. Solution times and solution quality of the exact algorithm changes by node size and station pattern. At each row of the table, there is a different station pattern while node size or density may

Table 5.9: Optimal Solutions for the Instances with nodes 24 and 41 using PEVEEP-MISOCP

|N|	**Density** %	**Pattern**	**Optimal Path**	**Total Time Spent** (sec)	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)
24	20	1	1-20-21-24	10363	0.582	1.426
24	20	2	1-20-21-24	11492	0.582	1.491
24	20	3	1-20-21-24	9601	0.582	1.451
24	30	1	1-19-22-23-24	12195	0.642	2.271
24	30	2	1-20-21-24	9602	0.582	1.392
24	30	3	1-20-21-24	10364	0.582	1.661
24	40	1	1-20-21-24	10329	0.582	1.575
24	40	2	1-20-21-24	10363	0.582	1.543
24	40	3	1-20-21-24	10330	0.582	1.430
41	20	1	1-2-3-9-10-11-17-20-27-32-36-40-41	77221	1.940	31.030
41	20	2	1-2-3-9-10-11-17-20-27-32-36-40-41	54087	1.940	67.160
41	20	3	1-2-3-9-10-11-17-20-27-32-36-40-41	55670	1.940	48.327
41	30	1	1-2-3-9-10-11-17-20-27-32-36-40-41	55798	1.940	11.905
41	30	2	1-2-3-9-10-11-17-20-27-32-36-40-41	56752	1.940	9.530
41	30	3	1-2-3-9-10-11-17-20-27-32-36-40-41	60470	1.940	6.326
41	40	1	1-2-3-9-10-15-25-31-35-36-40-41	60029	2.060	95.480
41	40	2	1-2-3-9-10-11-17-20-27-32-36-40-41	55534	1.940	81.580
41	40	3	1-2-3-9-10-15-16-19-26-32-36-40-41	60956	2.120	84.411

Table 5.10: Optimal Solutions for the Instances with nodes 50 using PEVEEP-MISOCP

|N|	**Density** $\%$	**Pattern**	**Optimal Path**	**Total Time Spent** (sec)	**Energy Consumption Value** ($\times 10^7$)	**Solution Time** (sec)
50	20	1	1-15-23-28-27-35 -36-38-46-45-50	24087	10.030	14.863
50	20	2	1-15-24-29-37-44-45-50	18044	9.143	25.875
50	20	3	1-15-24-29-37-44-45-50	37367	9.143	11.556
50	30	1	1-15-24-30-38-46-45-50	39834	9.501	32.610
50	30	2	1-15-24-29-37-44-45-50	18336	9.143	32.524
50	30	3	1-15-23-29-37-44-45-50	34968	9.650	15.536
50	40	1	1-15-24-29-37-44-45-50	42899	9.143	32.009
50	40	2	1-15-24-29-37-44-45-50	18149	9.143	22.517
50	40	3	1-15-24-29-37-44-45-50	18044	9.143	16.244

remain same. So, 6 $\times$ 3 $\times$ 3 = 54 different instances are tried. Node sizes, densities and patterns are shown in first, second and third columns of the Tables, respectively. Fourth column includes the chosen paths for each instance. Objective function value, total time spent and solution times are shown in remaining columns. For all the solutions, parameters given in Table 5.1 are used.

For the instances whose node sizes are 24, solution times range between 1.39 and 2.27 $seconds$. Station density does not have a big impact on solutions as there is not too much difference between solution times with varying densities. Optimal solution is found as the path $1-19-22-23-24$ for the instance $(24-30-1)$ where its objective function value is equal to 6.420×10^7 $joule$. For other networks whose sizes are 24, objective function value is equal to 5.821×10^7 $joule$. Optimal path for those is 1 – 20 – 21 – 24. Total time spent on the network is around 10^6 $seconds$ for node size 24. It includes charging time and total travel time in it.

On the other hand, it is seen that different objective function values are calculated for the graphs whose node sizes are 41. Instances $(41-40-1)$ and $(41-40-3)$ have

Table 5.11: Optimal Solutions for the Instances with nodes 146 using PEVEEP-MISOCP

|N|	**Density** $\%$	**Pattern**	**Optimal Path**	**Total Time Spent** (sec)	**Energy Consumption Value** **($\times 10^8$)**	**Solution Time** (sec)
146	20	1	1-69-70-71-72-83-98-99-112-113-127-135-146	56718	1.171	2482.880
146	20	2	1-69-70-71-72-83-98-99-112-113-127-135-146	27435	1.171	3954.210
146	20	3	1-69-70-71-72-83-98-99-112-113-127-135-146	28833	1.171	5377.590
146	30	1	1-69-70-71-72-83-98-99-112-113-127-135-146	28833	1.171	4890.101
146	30	2	1-69-70-71-72-83-98-99-112-113-127-135-146	30655	1.171	3549.090
146	30	3	1-79-80-94-95-122-123-124-144-145-146	24356	1.142	5307.320
146	40	1	1-79-80-94-95-122-123-124-144-145-146	26301	1.142	2627.290
146	40	2	1-79-80-94-95-122-123-124-144-145-146	24844	1.142	5820.510
146	40	3	1-79-80-94-95-122-123-124-144-145-146	29791	1.142	5343.900

Table 5.12: Optimal Solutions for the Instances with nodes 195 using PEVEEP-MISOCP

|N|	**Density** $\%$	**Pattern**	**Optimal Path**	**Total Time Spent** (sec)	**Energy Consumption Value** $(\times 10^8)$	**Solution Time** (sec)
195	20	1	1-17-31-32-33-34-35-51 -59-74-87-88-89-90-91-92-105- 123-124-141-159-160- 176-193-194-195	106415	2.528	30229.600
195	20	2	1-17-31-32-33-34-35-51 -59-60-76-77-89-90 -91-92-105-123-124 -141-159-160-161-177-194-195	94130	2.516	15103.100
195	20	3	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	73810	2.516	21604.600
195	30	1	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	98867	2.516	25729.300
195	30	2	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	81015	2.516	10502.800
195	30	3	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	112894	2.516	25216.400
195	40	1	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	79549	2.516	50945.000
195	40	2	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	79269	2.516	46526.300
195	40	3	1-17-31-32-49-56 -57-73-74-75-87-88- 89-90-91-92-105-123-124 -141-159-160-176-193-194-195	80493	2.534	13823.000

higher energy consumption values compared to remaining networks, which are 2.060×10^8 $joule$ and 2.120×10^8 $joule$, respectively. This sources due to the locations of stations on the graph. If PEV needs energy on the path, but the station is not near, then PEV deviates from the energy optimal path to recharge its battery. In this case, path selection can change for different densities and patterns. As expected, due to longer distances energy consumption increases.

Another reason which increases energy consumption level is higher values of lower limits, L. PEV wants to be driven at optimal speed which is equal to 13.035 m/sec. Due to limitations on speed, L and U, sometimes it has to deviate from the optimal speed. Time limit also causes a change in this situation, and PEV sometimes needs to drive faster. If optimal path on a network has higher values for L compared to other instances with same properties, energy consumption can be higher. As stated in Chapter 3, energy consumption function of a PEV is a third degree function of V. Hence, L values are important while the algorithm selects optimal path. Total time spent on network is around 60000 $seconds$ for node size 41 except the instance $(41-20-1)$ whose total time spent on the network is 77221.1 $seconds$. This variation is also due to total distance and V values of the chosen path. As the path gets longer, total time spent on the path has a higher value. If we look at the impact of densities on solution times, it can be said that solution times become higher while density is larger. Solution time for instances whose densities are 20% is in between 31 and 68 $seconds$, while it is in between 81 and 96 $seconds$ for the instances with 40% density. Here we can say that number of alternative paths to be constructed increases if there are more number of stations on a network. In other words, PEV can be driven at many different roads whose energy consumption levels are close. Therefore, exact algorithm has difficulty in path selection increasing the computational effort.

For the graphs whose node sizes are 50, instances $(50-20-1)$, $(50-30-1)$ and $(50-30-3)$ have higher energy consumption values, which are 1.003×10^8 $joule$, 9.501×10^7 $joule$ and 9.650×10^7 $joule$, respectively. Speed at each edge is equal to 13.035 m/sec if it is higher than the value of L. Otherwise, PEV is driven at L. For each network, different levels of L and different station patterns cause these different path selections. Algorithm tends to choose an edge whose L is lower. Solution times range between 11.56 sec and 32.61 sec. For the instances, $(50-30-1)$, $(50-30-1)$

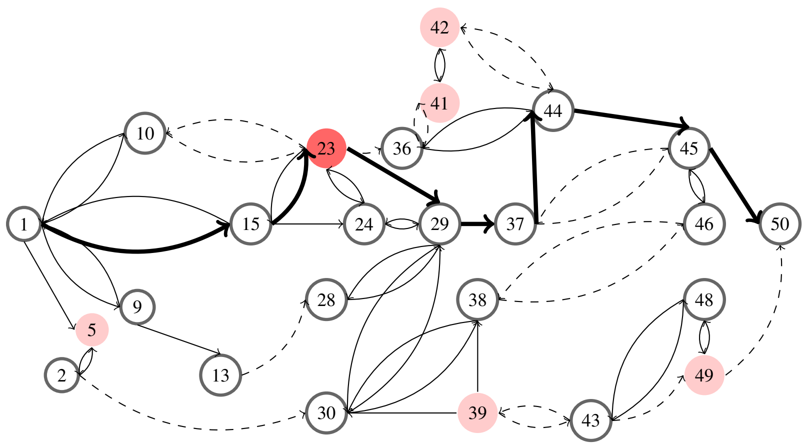


Figure 5.1: Solution of the Instance $(50-30-3)$

and $(50-30-1)$, station pattern probably causes a lot of feasible path options. Hence, algorithm gets difficulty in choosing the best one and computational time increases. As 41 and 50 are near values, solution times of instances whose node sizes are 50 is near to solution times of instances whose node sizes are 41. Optimal solution is found as path $1-15-23-28-27-35-36-38-46-45-50$ for the instance $(50-20-1)$. For $(50-30-1)$ and $(50-30-3)$, the optimal paths are $1-15-24-30-38-46-45-50$ and $1-15-23-29-37-44-45-50$. Optimal path is chosen as $1-15-24-29-37-44-45-50$ for the remaining instances whose objective function values are 9.143×10^7 $joule$. Total time spent on each network ranges between 18044 and 42890 $seconds$. In Figure 5.1 an illustration of the solution (50-30-3) is shown. All the arcs of the network are not given. A representation is used to illustrate this solution. Thick lines show the chosen path over the network, and dashed lines represent some other possible paths between two nodes. Tiny lines also show some of the real arcs of the network. Pink nodes are station nodes, and there are more number of stations which are not shown. The path is starting from the node 1, and ends at the node 50. The PEV stops at node 23 in order to recharge its battery.

For the graphs with 146 nodes, solution times increase substantially compared to smaller size instances which are 24, 41 and 50. Solution times for these instances are in between 2482.88 and 5820.51 sec. It can be said that solution times becomes much higher if number of nodes on a network increases too much. Solving the instances with 146 nodes can be accepted as much more demanding in terms of solution times compared to previous instances. Here, there are a lot of path options for each instance due to larger instance size. Algorithm again tends to choose an edge whose L is lower. Instances $(146-20-1)$, $(146-20-2)$, $(146-20-3)$, $(146-30-1)$ and $(146-30-2)$ have higher energy consumption values, which are 1.171×10^8 $joule$. Optimal path is chosen as $1-69-70-71-72-83-98-99-112-113-127-135-146$. Remaining instances use the path $1-79-80-94-95-122-123-124-144-145-146$ whose objective function value is equal to 1.142×10^8 $joule$. Total time spent on the network is around 30000 $seconds$ except the instance $(146-20-1)$ whose total time is nearly equal to 57000 $seconds$.

Likewise, instances with 195 nodes have also higher computational times. Solution time is equal to 50945 seconds for the instance $(195-40-1)$. Station pattern assigned to this network may cause difficulty in path selection. Other ones have solution times which are still much higher than the solution times of instances whose node sizes are 146. It can be said that size of the network has a big impact on the solution performance of the exact algorithm. Instances $(195-20-1)$ and $(195-40-3)$ have higher energy consumption values, which are 2.528×10^8 $joule$ and 2.534×10^8 $joule$, respectively. Total energy consumption value for optimal path is equal to 2.516×10^8 $joule$ for other instances. Optimal paths chosen are much longer than the ones in previous solutions with smaller sizes. Hence, energy consumption values are higher as total distance increases. Total time spent on the network is in between 73000 and 113000 $seconds$ due to longer optimal paths whose total distances are higher compared to previous graphs.

For the graphs whose node sizes are 321, exact algorithm could not solve the instances. Although algorithms are run for 3 days, even a solution with an optimality gap cannot be reached. For the exact algorithm solutions, it can be said that taking exact solutions becomes more difficult while node sizes increase. Densities and station patterns affect the solution time differently. Hence, it is difficult to generalize the effect of these parameters for all instances. Station pattern is the main property of the graph deciding number of feasible paths over a network and computational effort to reach an optimal solution.

As computational effort becomes higher for the larger instances, heuristic approaches are proposed in this thesis. Two different experiments are conducted using matheuristic approach. First experiment includes the application of matheuristic starting with an MIP solution of PEVTEPP-MIP in Appendix B. This initialization methods solve an MIP model for a given instance. Second initialization method uses k^{th} shortest path over the network. If the chosen path is energy feasible, then matheuristic starts to improve the solution on hand. It is expected to have higher computational times when MIP initialization is used due to the computational effort behind it. For each instance, heuristic solutions and solution times are in Tables 5.13, 5.14 and 5.15. Solution times and solution quality of the matheuristic algorithm are recorded. At each row of the table, there is a different station pattern while node size or density may remain same. So, all instances used for the exact algorithm are tried again. Node sizes, densities and patterns are shown in the first, second and third columns of the Tables. Fourth column includes the chosen paths for each instance. Objective function value, total time spent and solution times are shown in remaining columns. For all the solutions, parameters given in Table 5.1 and Chapter 5.2 are used. Optimality gap is calculated for each instance.

For the instances whose node sizes are 24, solution time is around 5 $seconds$ for each instance. Station density does not affect solution times too much as seen. Optimal solutions are found for each network. Hence, for all instances whose node sizes are 24, optimal paths are reached. This heuristic solves PEVFEEP-MISOCP at the end of the algorithm to decide energy and speed variables. After the path is chosen, this model is solved for each instance. Optimality Gap column is equal to 0 at each row in Table 5.13. Matheuristic with MIP initialization works with a little bit higher computational times compared to exact algorithm for these networks.

For the instances where number of nodes is equal to 41, solution times range between 13.039 $seconds$ and 17.740 $seconds$. Optimality gap is also 0 for these networks. A big impact of density levels on networks cannot be observed. Solution times of Matheuristic with MIP initialization is lower compared to solution times of exact algorithm for these networks while the solution quality is same. Hence, Mathheuristic with MIP initialization can be preferred. Likewise, all solutions taken by matheuristic for graphs with 50 nodes are exact. Solution times are lower compared to solution

Table 5.13: Solutions for the Instances with nodes 24 and 41 using Matheuristic with MIP Initialization

|N|	**Density** $\%$	**Pattern**	**Path Found**	**Energy Consumption Value** **($\times 10^8$)**	**Solution Time** (sec)	**Optimality Gap** $(\%)$
24	20	1	1-20-21-24	0.582	5.481	0
24	20	2	1-20-21-24	0.582	5.660	0
24	20	3	1-20-21-24	0.582	4.800	0
24	30	1	1-19-22-23-24	0.642	6.003	0
24	30	2	1-20-21-24	0.582	5.911	0
24	30	3	1-20-21-24	0.582	5.599	0
24	40	1	1-20-21-24	0.582	5.962	0
24	40	2	1-20-21-24	0.582	5.877	0
24	40	3	1-20-21-24	0.582	5.907	0
41	20	1	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	17.740	0
41	20	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	16.269	0
41	20	3	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	17.072	0
41	30	1	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	16.865	0
41	30	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	16.231	0
41	30	3	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	16.464	0
41	40	1	1-2-3-9-10-15-25-31-35-36-40-41	2.060	14.834	0
41	40	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	16.409	0
41	40	3	1-2-3-9-10-15-16-19-26-32-36-40-41	2.120	13.039	0

Table 5.14: Solutions for the Instances with nodes 50 using Matheuristic with MIP Initialization

|N|	**Density** %	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^7$)	**Solution Time** (sec)	**Optimality Gap** (%)
50	20	1	1-15-23-28-27-35 -36-38-46-45-50	10.030	21.877	0
50	20	2	1-15-24-29-37-44-45-50	9.143	19.895	0
50	20	3	1-15-24-29-37-44-45-50	9.143	21.715	0
50	30	1	1-15-24-30-38-46-45-50	9.501	18.813	0
50	30	2	1-15-24-29-37-44-45-50	9.143	12.147	0
50	30	3	1-15-23-29-37-44-45-50	9.650	18.355	0
50	40	1	1-15-24-29-37-44-45-50	9.143	19.444	0
50	40	2	1-15-24-29-37-44-45-50	9.143	19.259	0
50	40	3	1-15-24-29-37-44-45-50	9.143	20.819	0

Table 5.15: Solutions for the Instances with nodes 146 using Matheuristic with MIP Initialization

|N|	**Density** %	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Optimality Gap** (%)
146	20	1	1-69-70-71-72-83-97- 110-111-125-134-145-146	1.194	246.635	1.96
146	20	2	1-79-92-93-94-95- 122-123-124-144-145-146	1.171	235.025	0
146	20	3	1-79-80-94-108-121- 122-123-124-144-145-146	1.180	177.819	0.77
146	30	1	1-69-70-71-72-83- 97-110-111-125-134-145-146	1.194	661.030	1.96
146	30	2	1-79-80-94-108-121- 122-123-124-144-145-146	1.180	283.124	0.77
146	30	3	1-79-80-94-95-122- 123-124-144-145-146	1.142	302.958	0
146	40	1	1-79-80-94-95-122- 123-124-144-145-146	1.142	349.652	0
146	40	2	1-79-80-94-95-122- 123-124-144-145-146	1.142	468.248	0
146	40	3	1-79-80-94-95-122- 123-124-144-145-146	1.142	420.103	0

times of exact algorithm.

If we come to larger instances, performance of Matheuristic with MIP initialization strictly dominates the performance of exact algorithm in terms of solution times. Solution times range between 177.819 $seconds$ and 661.030 $seconds$. This difference is due to the initialization part of the algorithm. For the instance $(146 - 40 - 1)$, algorithm solves PEVTEPP-MIP problem slower compared to other instances. Due to station pattern differences, solution times are a little bit different from each other though node size is still same. However, solution quality of the heuristic is not same with the exact algorithm. For the instances $(146-20-1)$, $(146-20-3)$, $(146-30-1)$ and $(146-30-2)$ optimality gaps are 1.96%, 0.77%, 1.96% and 0.77%, respectively. Algorithm gives exact solutions for remaining instances. Although solution quality is not better, the algorithm can still be used because of its lower solution times.

PEVTEPP-MIP problem takes maximum of optimal speed and L as V_{ij} values at the beginning of the algorithm execution. Then, it tries to find a path over the network minimizing total time spent on. It needs a significant effort, and it cannot give a solution for the instances whose sizes are 195. Therefore, we tried another initialization technique based on randomly chosen k shortest paths over the network. This method lists k paths while initiating the algorithm, then one of them is randomly chosen. If it is an energy feasible path, algorithm starts.

Solutions of this method can be seen in Tables 5.16, 5.17, 5.18, 5.19 and 5.20. For the instances whose node sizes are 24, 41 and 50, performance of algorithms with two different initialization methods are nearly the same. Neither of these two dominates the other. However, k^{th} path initialization method is faster for graphs with 146 nodes. All the solutions are exact except for the instances $(146 - 20 - 1)$ and $(146 - 30 - 1)$. Optimality gaps can be accepted as significant which are 16.99%. Compared to exact solution method, there is huge difference for all the instances in terms of computational times. It is also observed that algorithm gives exact solutions in a faster way for the instances with 195 nodes. Also, it dominates the previous method with its lower computational effort. Matheuristic with k^{th} path initialization also finds solutions for the instances with 321 nodes. They are best known solutions on hand, and computation times are reasonable which are around 300 $seconds$. They will be compared with the solutions of VNS algorithm.

It can be said that energy consumption levels for these instances are much more higher than the smaller ones due to longer distances of best paths. PEV drives through approximately 30 nodes which increases the total energy consumption as expected. Also, it stops more compared to other paths to recharge the battery at stations. Solution times range between 262 and 304 $seconds$. All the solutions are better than the solutions of VNS algorithm except for the instance $(321-20-1)$. Energy consumption level is 3.515×10^8 $joule$ for this network. Moreover, objective function is equal to 3.478×10^8 $joule$ for $(321-40-2)$ and $(321-40-3)$. PEV consumes 3.480×10^8 $joule$ at first instance, $(321-20-1)$. Objective functions of remaining instances is equal to 3.471×10^8 $joule$. As algorithm does not spend too much time on initialization, solutions are taken in shorter time intervals. Also, it is more probable to have feasible node combinations while exchanging the path between two chosen nodes because of longer path selections. There can be more number of trials giving better and feasible path exchanges. In addition to parameters, feasible path exchanges also affect the solution time. Algorithm randomly selects a pair of nodes where there is an exchange between. If this exchange makes the path feasible or better in terms of energy, the while loop is broken. So, computational effort is lower in these cases by chance. For both initialization methods, there can be solutions affected by this property.

Likewise, we proposed VNS algorithm starting with a path which is randomly constructed based on an initialization method. We have while loops for this algorithm again under three different neighborhood search definitions. For the neighborhoods, $StationInsertion$ and $ShortCut$, not performing the changes if the path is energy feasible is possible. Also, shortcut cannot be executed when the candidate set, including nodes where there is a direct edge between them, is empty. This situation may cause significant differences in solution times between instances with same size. VNS solutions can be seen on Tables 5.22, 5.23, 5.24, 5.25 and 5.26.

All the solutions found by VNS for instances with 24 nodes are exact. Solution times are around 9 $seconds$ except for the instance $(24-30-1)$. This instance is solved in about 15 $seconds$. This is probably sources from energy efficient but infeasible

Table 5.16: Solutions for the Instances with nodes 24 and 41 using Matheuristic with k^{th} Path Initialization

|N|	**Density** %	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Optimality Gap** (%)
24	20	1	1-20-21-24	0.582	5.824	0
24	20	2	1-20-21-24	0.582	7.585	0
24	20	3	1-20-21-24	0.582	7.074	0
24	30	1	1-19-22-23-24	0.642	6.795	0
24	30	2	1-20-21-24	0.582	5.803	0
24	30	3	1-20-21-24	0.582	7.553	0
24	40	1	1-20-21-24	0.582	4.460	0
24	40	2	1-20-21-24	0.582	6.289	0
24	40	3	1-20-21-24	0.582	6.266	0
41	20	1	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	10.278	0
41	20	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	13.650	0
41	20	3	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	10.488	0
41	30	1	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	21.049	0
41	30	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	19.353	0
41	30	3	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	15.690	0
41	40	1	1-2-3-9-10-15-25-31-35-36-40-41	2.060	8.223	0
41	40	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	10.061	0
41	40	3	1-2-3-9-10-15-16-19-26-32-36-40-41	2.120	8.952	0

Table 5.17: Solutions for the Instances with nodes 50 using Matheuristic with k^{th} Path Initialization

|N|	Density %	Pattern	Path Found	Energy Consumption Value ($\times 10^7$)	Solution Time (sec)	Optimality Gap (%)
50	20	1	1-15-23-28-27-35-36-44-45-50	10.510	21.615	4.79
50	20	2	1-15-24-29-37-44-45-50	9.143	16.567	0
50	20	3	1-15-24-29-37-44-45-50	9.143	16.705	0
50	30	1	1-15-24-30-38-46-45-50	9.501	17.262	0
50	30	2	1-15-24-29-37-44-45-50	9.143	16.196	0
50	30	3	1-15-23-29-37-44-45-50	9.650	18.427	0
50	40	1	1-15-24-29-37-44-45-50	9.143	15.628	0
50	40	2	1-15-24-29-37-44-45-50	9.143	15.669	0
50	40	3	1-15-24-29-37-44-45-50	9.143	13.644	0

Table 5.18: Solutions for the Instances with nodes 146 using Matheuristic with k^{th} Path Initialization

|N|	Density %	Pattern	Path Found	Energy Consumption Value ($\times 10^8$)	Solution Time (sec)	Optimality Gap (%)
146	20	1	1-79-80-94-108-121-131-132 -122-123-124-125-126-127-135-146	1.370	125.660	16.99
146	20	2	1-69-70-71-72-83-98- 99-112-113-127-135-146	1.171	137.632	0
146	20	3	1-69-70-71-72-83-98- 99-112-113-127-135-146	1.171	153.105	0
146	30	1	1-79-80-94-108-121-131-132 -122-123-124-125-126-127-135-146	1.370	127.060	16.99
146	30	2	1-69-70-71-72-83-98- 99-112-113-127-135-146	1.171	132.348	0
146	30	3	1-79-80-94-95-122- 123-124-144-145-146	1.142	133.020	0
146	40	1	1-79-80-94-95-122- 123-124-144-145-146	1.142	185.200	0
146	40	2	1-79-80-94-95-122- 123-124-144-145-146	1.142	134.205	0
146	40	3	1-79-80-94-95-122- 123-124-144-145-146	1.142	147.486	0

Table 5.19: Solutions for the Instances with nodes 195 using Matheuristic with k^{th} Path Initialization

|N|	**Density** %	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Optimality Gap** **(%)**
195	20	1	1-17-31-32-33-34-35-51 -59-74-87-88-89-90-91-92-105- 123-124-141-159-160- 176-193-194-195	2.528	252.230	0
195	20	2	1-17-31-32-33-34-35-51 -59-60-76-77-89-90 -91-92-105-123-124 -141-159-160-161-177-194-195	2.516	274.882	0
195	20	3	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	2.516	238.417	0
195	30	1	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	2.516	101.717	0
195	30	2	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	2.516	247.438	0
195	30	3	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	2.516	277.815	0
195	40	1	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	2.516	137.172	0
195	40	2	1-17-31-32-33-34-35-51 -59-60-76-77-89-90- 91-92-105-123-124-141 -159-160-161-177-194-195	2.516	276.057	0
195	40	3	1-17-31-32-49-56 -57-73-74-75-87-88- 89-90-91-92-105-123-124 -141-159-160-176-193-194-195	2.534	112.697	0

Table 5.20: Solutions for the Instances with nodes 321 using Matheuristic with k^{th} Path Initialization

|N|	**Density** $\%$	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Optimality Gap*** **(%)**
321	20	1	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	270.150	0
321	20	2	1-18-28-61-62-99-120-121-135-136 -137-155-156-173-174-175-176 -212-213-214-231-248-249-250-262 -283-284-285-301-302-318-319-320-321	3.471	303.200	0
321	20	3	1-18-28-61-62-99-120-121-135-136 -137-155-156-173-174-175-176 -212-213-214-231-248-249-250-262- 283-284-285-301-302-318-319-320-321	3.471	291.064	0
321	30	1	1-2-3-4-20-21-22-23-24-25-37-38 -72-90-91-110-129-144-158-178- 195-216-217-218-219-220 -253-266-287-288-289-321	3.515	290.656	0.40
321	30	2	1-18-28-61-62-99-120-121-135-136-137 -155-156-173-174-175-176-212- 213-214-231-248-249-250-262-283- 284-285-301-302-318-319-320-321	3.471	285.319	0
321	30	3	1-18-28-61-62-99-120-121-135-136- 137-155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283-284 -285-301-302-318-319-320-321	3.471	288.900	0
321	40	1	1-18-28-61-62-99-120-121-135-136- 137-155-156-173-174-175-176- 212-213-214-231-248-249-250-262- 283-284-285-301-302-318-319-320-321	3.471	299.961	0
321	40	2	1-18-28-61-62-99-120-121 -135-136-137-155-156-173-174-175- 176-177-178-195-216-217-218-219 -220-253-266-287-288-289-321	3.477	262.030	0
321	40	3	1-18-28-61-62-99-120-121-135 -136-137-155-156-173-174-175-176- 177-178-195-216-217-218-219-220 -253-266-287-288-289-321	3.477	271.869	0

**(Optimality gaps are calculated according to best known solutions)*

Table 5.21: Analysis of PEV Speed with changing T on networks with 50 nodes

|N|	**Density** %	**Pattern**	α	T (sec)	**Path Found**	**Energy Consumption Value** $(\times 10^8)$	**Average Speed Along the Path** (m/sec)	**Average Increase in Average Speed** (%)	**Solution Time** (sec)
50	20	1	1	24087	1-15-23-28-27-35 -36-38-46-45-50	1.003	14.263	0	19.561
50	20	1	0.9	21678	1-15-23-28-27-35 -36-38-46-45-50	1.004	14.598	2.35	21.854
50	20	1	0.85	20474	1-15-23-28-27-35 -36-38-46-45-50	1.013	15.922	11.63	22.682
50	20	1	0.8	19269	1-15-23-28-27-35 -36-38-46-45-50	1.052	18.401	29.01	22.662
50	30	2	1	18336	1-15-24-29-37-44-45-50	0.914	14.796	0	17.238
50	30	2	0.9	16502	1-15-24-29-37 -44-45-50	0.935	17.276	16.76	20.001
50	30	2	0.89	16319	1-15-24-29-37 -44-45-50	0.943	17.829	20.50	19.942
50	30	2	0.88	16135	1-15-24-29-37 -44-45-50	0.956	18.588	25.63	20.519
50	30	3	1	34968	1-15-23-29-37 -44-45-50	0.965	15.550	0	12.354
50	30	3	0.55	19232	1-15-23-29-37 -44-45-50	0.965	15.801	1.62	18.417
50	30	3	0.53	18533	1-15-23-29-37 -44-45-50	0.973	16.668	7.19	19.421
50	30	3	0.52	18183	1-15-23-24-29- 37-44-45-50	0.981	17.173	9.21	19.803

paths. In this case, algorithm tries more number of options to reach the best path. Moreover, it is possible to have insignificant differences in solution times while density is changing. This is because of the structure of the algorithm. Independently from the density, algorithm makes K and I number of iterations for all options. However, node size affects the solution time as path constructions are more difficult due to a lot of combinations. For the instances whose node sizes are 41 and 50, solution times are around 20 $seconds$. All the solutions are exact except for the instance $(41 - 40 - 3)$. Energy consumption level is found as 2.153×10^8 $joule$, and the gap is 1.56%. Gap can be accepted as small, and solution time is one fourth of the solution time of exact algorithm. If we look at the solutions of instances whose sizes are 146, VNS works with much more smaller computational times compared to the exact algorithm. Exact solutions are obtained except for the instances $(146 - 20 - 2)$ and $(146 - 30 - 1)$. Optimality gaps are 1.54% and 2.56%, respectively. Instances $(146 - 40 - 2)$ and $(146 - 40 - 3)$ have higher computational times compared to other instances with the same node size. This probably results from not executing some of the neighborhood searches for other solutions.

First and last network with node size 195 have solutions with gaps 1.58% and 0.24%, respectively. Energy consumption value of the instance $(195 - 20 - 1)$ is equal to 2.569×10^8 $joule$. Gap between optimal solution is not too much large. Objective function value is 2.540×10^8 $joule$ for the network $(195 - 40 - 3)$. There is a huge difference in terms of solution times between VNS and exact solution method for this node size. Hence, VNS is more preferable compared to exact method considering small optimality gaps.

Node size has a big impact on solutions provided by VNS heuristic. Since number of path exchanges are affected by the size of candidate set, solution time increases while the number of nodes increases. As expected, solutions of instances with 321 nodes are taken in longer time intervals. Due to their heavy computational requirements, Matheuristic with k^{th} path initialization and VNS algorithm only find solutions for the instances with 321 nodes. Matheuristic approach gives the best solutions except for the instance $(321 - 30 - 1)$. VNS solves this instance in 786.187 $seconds$. Total energy consumption is equal to 3.501×10^8 $joule$. Other instances also have very small gaps, ranging between 0.09% and 0.26%. Solution of $(321 - 20 - 1)$ is found same by VNS and Matheuristic with k^{th} path initialization, whose objective function value is equal to $3.480 \times 10^8\ joule$. Furthermore, some experiments are made on instances whose node sizes are 50 using a solution method to comment on velocity decisions of the PEV under different total time limits. Table 5.21 shows the results of experiments. Table includes instances whose upper time limits are generated using total time of the tour found in exact solutions. We used a parameter α at different values to get tight upper time limits. Fourth column represents the value of α, which makes the analysis on PEV speed more clear. α should be chosen in a way that the problem is still feasible. Therefore, each instance is analyzed under different values of α. Fifth column is the multiplication of total time spent along the exact path and α. Here, it is seen that solution times, in last column, increases with decreasing value of T. Moreover, average speed along the selected path is given in column 7. First row shows the results of experiments under untight time limit for each instance. When time limit is not tight, speed on each arc is equal to maximum of L_{ij} and optimal speed. Hence, average increase in speed is calculated based on the average speed at first row of each different instance result. It is expected that as the time limit gets lower, PEV drives faster to ensure time feasibility while trying to balance energy consumption on arcs. In other words, as α becomes smaller, average speed and the energy consumption along the path increases. For each instance except the $(50-30-3)$, chosen path is the same while time limit gets smaller. However, values of speeds increase. If we look at the solution of the instance $(50-30-3)$ while α is equal to 0.52, PEV changes its path because of lower energy consumption on that changed part of the road segment.

In addition to these instances, we created a network with a different structure. As defined in Chapter 5.1, this network includes two different node clusters. These two areas connected by the help of bridges defined between last 6 nodes of 1^{st} cluster and first 6 nodes of 2^{nd} cluster. 40% level is used to analyze behavior of the PEV on this graph. Our purpose is to show that driving on the shortest path is not always the optimal solution for energy efficient path problem using speed optimization. As values of speeds at each road segment can change, energy consumption may have significant changes. We ordered 10000 shortest paths of the network using Yen's algorithm to make observations on each. It can be stated that there is no energy feasible path in

Table 5.22: Solutions for the Instances with nodes 24 and 41 using VNS heuristic

|N|	**Density** $\%$	**Pattern**	**Path Found**	**Energy Consumption Value** **(**$\times 10^8$**)**	**Solution Time** (sec)	**Optimality Gap** $(\%)$
24	20	1	1-20-21-24	0.582	8.820	0
24	20	2	1-20-21-24	0.582	9.341	0
24	20	3	1-20-21-24	0.582	8.870	0
24	30	1	1-19-22-23-24	0.642	15.584	0
24	30	2	1-20-21-24	0.582	9.415	0
24	30	3	1-20-21-24	0.582	9.925	0
24	40	1	1-20-21-24	0.582	8.053	0
24	40	2	1-20-21-24	0.582	9.660	0
24	40	3	1-20-21-24	0.582	9.002	0
41	20	1	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	20.913	0
41	20	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	20.304	0
41	20	3	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	20.275	0
41	30	1	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	17.476	0
41	30	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	16.718	0
41	30	3	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	17.147	0
41	40	1	1-2-3-9-10-15-25-31-35-36-40-41	2.060	21.327	0
41	40	2	1-2-3-9-10-11-17-20-27-32-36-40-41	1.940	19.167	0
41	40	3	1-2-3-9-14-23-24-25-31-35-36-40-41	2.153	22.943	1.56

Table 5.23: Solutions for the Instances with nodes 50 using VNS heuristic

|N|	Density %	Pattern	Path Found	Energy Consumption Value ($\times 10^7$)	Solution Time (sec)	Optimality Gap (%)
50	20	1	1-15-23-28-27-35 -36-38-46-45-50	10.030	26.801	0
50	20	2	1-15-24-29-37-44-45-50	9.143	16.693	0
50	20	3	1-15-24-29-37-44-45-50	9.143	16.057	0
50	30	1	1-15-24-30-38-46-45-50	9.501	17.723	0
50	30	2	1-15-24-29-37-44-45-50	9.143	19.174	0
50	30	3	1-15-23-29-37-44-45-50	9.650	27.636	0
50	40	1	1-15-24-29-37-44-45-50	9.143	25.534	0
50	40	2	1-15-24-29-37-44-45-50	9.143	23.837	0
50	40	3	1-15-24-29-37-44-45-50	9.143	22.682	0

Table 5.24: Solutions for the Instances with nodes 146 using VNS heuristic

|N|	Density %	Pattern	Path Found	Energy Consumption Value ($\times 10^8$)	Solution Time (sec)	Optimality Gap (%)
146	20	1	1-69-70-71-72-83-98- 99-112-113-127-135-146	1.171	173.915	0
146	20	2	1-79-92-93-107-108-121- 122-123-124-144-145-146	1.189	128.362	1.54
146	20	3	1-69-70-71-72-83-98- 99-112-113-127-135-146	1.171	149.095	0
146	30	1	1-79-80-94-95-122-123 -133-143-144-145-146	1.201	156.894	2.56
146	30	2	1-69-70-71-72-83-98- 99-112-113-127-135-146	1.171	139.795	0
146	30	3	1-79-80-94-95-122- 123-124-144-145-146	1.142	135.365	0
146	40	1	1-79-80-94-95-122- 123-124-144-145-146	1.142	116.507	0
146	40	2	1-79-80-94-95-122- 123-124-144-145-146	1.142	295.054	0
146	40	3	1-79-80-94-95-122- 123-124-144-145-146	1.142	295.957	0

Table 5.25: Solutions for the Instances with nodes 195 using VNS heuristic

|N|	**Density** %	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Optimality Gap (%)**
195	20	1	1-17-31-32-33-34-35 -51-59-60-76-77-89-102-116-133 -150-168-186-187-188-189 -190-191-192-193-194-195	2.569	289.926	1.58
195	20	2	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	580.705	0
195	20	3	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	564.908	0
195	30	1	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	251.959	0
195	30	2	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	576.781	0
195	30	3	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	598.117	0
195	40	1	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	252.989	0
195	40	2	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92 -105-123-124-141-159 -160-161-177-194-195	2.516	579.988	0
195	40	3	1-17-31-48-49-56-57- 58-74-75-87-88-89-90-91- 92-105-123-124-141- 159-160-176-193-194-195	2.540	596.368	0.24

Table 5.26: Solutions for the Instances with nodes 321 using VNS heuristic

|N|	**Density** %	**Pattern**	**Path Found**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Optimality Gap*** **(%)**
321	20	1	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	1179.040	0
321	20	2	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	1040.010	0.26
321	20	3	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	670.037	0.26
321	30	1	1-18-28-29-30-49-64-84-101-136 -137-155-156-173-174-175-176 -212-213-214-231-248-249-250-262 -283-284-285-301-302-318-319-320-321	3.501	786.187	0
321	30	2	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	760.605	0.26
321	30	3	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	680.517	0.26
321	40	1	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	706.225	0.26
321	40	2	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	689.348	0.09
321	40	3	1-18-28-61-62-63-83-100-101-136-137 -155-156-173-174-175-176-212-213 -214-231-248-249-250-262-283 -284-285-301-302-318-319-320-321	3.480	667.223	0.09

**(Optimality gaps are calculated according to best known solutions)*

Table 5.27: Solutions for the Instance with 40 nodes

|N|	Density %	Algorithm Used	Path Found	Energy Consumption Value ($\times 10^8$)	Solution Time (sec)	Optimality Gap (%)
40	40	Exact	1-4-7-10-13-16-23-27-28-32-36-40	1.634	433.479	0
		Matheuristic with MIP Initialization	1-4-6-9-12-15-23-27-34-35-39-40	1.650	41.732	0.98
		Matheuristic with k^{th} Path Initialization	-	-	-	-
		VNS heuristic	1-4-7-10-13-16-23-27-34-35-39-40	1.642	20.010	0.49

the first 10000 shortest paths. Thus, none of the algorithms selects one of the shortest paths of the network. This makes the processes for each algorithm difficult as there are few feasible path options. Solution of this network using PEVEEP-MISOCP and 2 heuristics can be seen on Table 5.27.

Objective function value is equal to 1.634×10^8 $joule$ in exact solution. Solution is taken in approximately 434 $seconds$. If we compare this network with the networks with same size in Table 5.9, solution time is much larger. This is because of the complexity of the constructed graph. There are a lot of paths which are energy or time infeasible. Hence, algorithms have difficulty in choosing the best one. Moreover, VNS heuristic performs better compared to matheuristic approach with MIP initialization. Its gap and solution time is smaller than the results of the matheuristic. VNS heuristic is preferable when it is compared to the matheuristic, but both heuristic methods can be used in place of PEVEEP-MISOCP in order to get the solution in a short period of time. As our second approach starts with a randomly chosen shortest path, here we cannot use it to solve the problem on this network. When it is executed, algorithm cannot start to solve the problem due to not being able to find an energy feasible path. This result strongly supports our argument, a shortest path may not be the optimal solution for the energy efficient path problem while velocity is a decision variable.

Comparison is made between heuristics proposed, and performances can be seen in Tables 5.28, 5.29 and 5.30. Dominating alternatives are shown using bold font. When we compared the instances whose sizes are 24, optimality gaps for all heuristics are 0

Table 5.28: Comparison of Heuristic Performances on Networks whose Node sizes are 24 and 41

			Matheuristic with MIP Initialization		**Matheuristic with k^{th} Path Initialization**		**VNS heuristic**	
|N|	**Density** %	**Pattern**	**Optimality Gap** (%)	**Solution Time** (sec)	**Optimality Gap** (%)	**Solution Time** (sec)	**Optimality Gap** (%)	**Solution Time** (sec)
24	20	1	**0**	**5.481**	0	5.824	0	8.820
24	20	2	**0**	**5.660**	0	7.585	0	9.341
24	20	3	**0**	**4.800**	0	7.074	0	8.870
24	30	1	**0**	**6.003**	0	6.795	0	15.584
24	30	2	0	5.911	**0**	**5.803**	0	9.415
24	30	3	**0**	**5.599**	0	7.553	0	9.925
24	40	1	0	5.962	**0**	**4.460**	0	8.053
24	40	2	**0**	**5.877**	0	6.289	0	9.660
24	40	3	**0**	**5.907**	0	6.266	0	9.002
41	20	1	0	17.740	**0**	**10.278**	0	20.913
41	20	2	0	16.269	**0**	**13.650**	0	20.304
41	20	3	0	17.072	**0**	**10.488**	0	20.275
41	30	1	**0**	**16.865**	0	21.049	0	17.476
41	30	2	**0**	**16.231**	0	19.353	0	16.718
41	30	3	0	16.464	**0**	**15.690**	0	17.147
41	40	1	0	14.834	**0**	**8.223**	0	21.327
41	40	2	0	16.409	**0**	**10.061**	0	19.167
41	40	3	0	13.039	**0**	**8.952**	1.56	22.943

Table 5.29: Comparison of Heuristic Performances on Networks whose Node sizes are 50 and 146

			Matheuristic with MIP Initialization		**Matheuristic with k^{th} Path Initialization**		**VNS heuristic**	
|N|	**Density** $\%$	**Pattern**	**Optimality Gap** $(\%)$	**Solution Time** (sec)	**Optimality Gap** $(\%)$	**Solution Time** (sec)	**Optimality Gap** $(\%)$	**Solution Time** (sec)
50	20	1	**0**	**21.877**	4.79	21.615	0	26.801
50	20	2	0	19.895	**0**	**16.567**	0	16.693
50	20	3	0	21.715	0	16.705	**0**	**16.057**
50	30	1	0	18.813	**0**	**17.262**	0	17.723
50	30	2	**0**	**12.147**	0	16.196	0	19.174
50	30	3	**0**	**18.355**	0	18.427	0	27.636
50	40	1	0	19.444	**0**	**15.628**	0	25.534
50	40	2	0	19.259	**0**	**15.669**	0	23.837
50	40	3	0	20.819	**0**	**13.644**	0	22.682
146	20	1	1.96	246.635	16.99	125.660	**0**	**173.915**
146	20	2	0	235.025	**0**	**137.632**	1.54	128.362
146	20	3	0.77	177.819	0	153.105	**0**	**149.095**
146	30	1	1.96	661.030	16.99	127.060	**2.56**	**156.894**
146	30	2	0.77	283.124	**0**	**132.348**	0	139.795
146	30	3	0	302.958	**0**	**133.020**	0	135.365
146	40	1	0	349.652	0	185.200	**0**	**116.507**
146	40	2	0	468.248	**0**	**134.205**	0	295.054
146	40	3	0	420.103	**0**	**147.486**	0	295.957

Table 5.30: Comparison of Heuristic Performances on Networks whose node sizes are 195 and 321

			Matheuristic with k^{th} Path Initialization		**VNS heuristic**	
|N|	**Density** %	**Pattern**	**Optimality Gap** (%)	**Solution Time** (sec)	**Optimality Gap*** (%)	**Solution Time** (sec)
195	20	1	**0**	**252.230**	1.58	289.926
195	20	2	**0**	**274.882**	0	580.705
195	20	3	**0**	**238.417**	0	564.908
195	30	1	**0**	**101.717**	0	251.959
195	30	2	**0**	**247.438**	0	576.781
195	30	3	**0**	**277.815**	0	598.117
195	40	1	**0**	**137.172**	0	252.989
195	40	2	**0**	**276.057**	0	579.988
195	40	3	**0**	**112.697**	0.24	596.368
321	20	1	**0**	**270.150**	0	1179.040
321	20	2	**0**	**303.200**	0.26	1040.010
321	20	3	**0**	**291.064**	0.26	670.037
321	30	1	0.40	290.656	**0**	**786.187**
321	30	2	**0**	**285.319**	0.26	760.605
321	30	3	**0**	**288.900**	0.26	680.517
321	40	1	**0**	**299.961**	0.26	706.225
321	40	2	**0**	**262.030**	0.09	689.348
321	40	3	**0**	**271.869**	0.09	667.223

**(Optimality gaps are calculated according to best known solutions)*

in Table 5.28. Instances up to $(24-30-3)$ except $(24-30-2)$, are solved faster by matheuristic with MIP Initialization. For the instances whose sizes are 24 and densities are 40, first instance is solved faster by matheuristic with k^{th} Path Initialization, while the others are solved faster by matheuristic with MIP Initialization. It can also be stated that difference between solution times is not too large. Moreover, for the instances whose sizes are 41, VNS heuristic is again dominated in terms of both solution quality and solution times by other 2 heuristics. All of them have 0 optimality gap, but matheuristic with k^{th} Path Initialization solves all the instances faster except $(41-30-1)$ and $(41-30-2)$.

When we look at Table 5.29, VNS heuristic is the best option for some of the instances. For the instance $(50-20-3)$, it works faster with 0 optimality gap. For other instances whose sizes are 50, except the instance $(50-20-1)$, $(50-30-2)$ and $(50-30-3)$, matheuristic with k^{th} Path Initialization is the dominating alternative. It is important to state that there are no too significant differences between these heuristics for the networks with 50 nodes. For larger instances with 146 nodes, matheuristic with MIP Initialization does not perform well as there are optimality gaps and higher solution times. VNS heuristic performs better for the instances $(146-20-1)$, $(146-20-3)$, $(146-30-1)$ and $(146-40-1)$. For remaining instances, matheuristic with k^{th} Path Initialization is the dominating one in terms of solution quality and time.

For the networks with more than 195 nodes, matheuristic with MIP Initialization cannot generate a solution due to initialization problems. At first step, PEVTEPP-MIP tries to solve the instance in order to have an initial path which is energy feasible. However, this takes too much time. So, performance of matheuristic with MIP Initialization is already dominated by other 2 heuristics. Therefore, they are not available in Table 5.30. Matheuristic with k^{th} Path Initialization performs better in terms of solution quality and time for all instances except $(321-20-1)$. This instance is solved by matheuristic with k^{th} Path Initialization in 290.656 $seconds$ with 0.40% optimality gap. On the other hand, VNS gives the exact solution though its solution time is higher.

To summarize, VNS heuristic performs well for some of the larger instances with more than 41 nodes. Generally, matheuristic with k^{th} Path Initialization is the best option for all instances. Since initialization takes too much time using Matheuristic with MIP Initialization, it is the best option only for the instances whose sizes are smaller than 50 as expected.

Finally, total time spent along a fixed path, T, is changed to compare algorithm performances under different conditions. We put tighter upper bounds on each network to check whether PEV increases its speed or decrease number of stops to recharge. For each instance, total time spent is measured and kept as a decision variable. As a second step, these values are used to determine new time limits, T_{new}. A fraction α is multiplied by value of this total time spent. For example, if a PEV completes the path in 100000 $seconds$, and α is equal to 0.9, then T_{new} is determined as 90000 $seconds$. Solution method takes this value as a parameter. If objective function does not change with this T_{new}, α gets smaller until we get a change in the energy consumption level. Some of the instances are tried for this purpose. To make observation, instances with two different node sizes are solved. Results can be seen in Tables 5.31, 5.32, 5.33 and 5.34.

Some of the instances cannot be solved by some of the heuristics. This probably sources from not being able to perform path exchanges due to tight time limitation on each path. In other words, algorithms cannot make too much path exchanges to reach the best solution. Also, we made again fine tuning for Matheuristic approach to get a feasible solution. K is used as 10 to explore more areas while k is still 5. On the other hand, VNS heuristic is able to solve the instances with the same algorithm paramaters. Hence, the values of I, K and K_{RON} are not changed. For all the instances, it is expected to have an increase in energy level due to lower travel times and higher values of speeds. Also, energy taken at recharging stations should be less than the values taken previously. As there is a tight time limit over the network, PEV tends to take one of these actions. When tight upper time limits are applied, the best path over a network can change. An arc of the network with lower distance but higher lower speed limit may be a better option under tight upper time limit. In other words, it will be a better alternative although the PEV has to drive faster in order to satisfy feasibility conditions. Moreover, another observation is made on the behavior of PEV while it is under tight time limit. As mentioned, PEV tries to increase speed

$$\begin{tikzcd}1 \arrow[r, "V_{12} = 15.25"] & 2 \arrow[r, "V_{23} = 13.035"] & 3 \arrow[r, "V_{34} = 13.035"] & 4 \arrow[r, "V_{45} = 16.50"] & 5\end{tikzcd}$$

A Path from Node 1 to Node 5 with different V_{ij} values

$$\begin{tikzcd}1 \arrow[r, "V_{12} = 15.25"] & 2 \arrow[r, "V_{23} = 15.25"] & 3 \arrow[r, "V_{34} = 15.25"] & 4 \arrow[r, "V_{45} = 16.50"] & 5\end{tikzcd}$$

Change in Values of V_{23} and V_{34}

Figure 5.2: An Illustration of Increase in Variable V_{ij} along a Path

on some of the arcs. It starts from the arc with smallest V_{ij}. V_{ij} is increased until the second smallest speed value. After that, algorithm again checks each arc in order to find smallest V_{ij} value again. This iteration continues until time limitation is satisfied. As the energy consumption behavior of the PEV is based on a third degree function of V, it is better to have small increments on V while searching for a feasible solution. If increment in V_{ij} values causes energy infeasibility due to excess amount of energy consumption, then the instance may not be solved for given time limit, T.

An illustration is shown in Figure 5.2. If there is a path from 1 to 5 with different V_{ij} values assigned to each arc, increase in velocity values starts from the minimum one, which are V_{23} and V_{34}. It is assumed that time requirement can be met increasing V_{23} and V_{34} values up to $V_{12} = 15.25\ m/sec$ which is second smallest value of V. If there needs to be more increment on other V values to be under given time limit T_{new}, then V_{12}, V_{23} and V_{34} will be evaluated together to increase their values.

Table 5.31 shows the exact solutions when tight upper time limit is applied on different instances. Last column of the table shows the energy increase as a percentage. As shown in the table instance $(41 - 30 - 2)$ shows the largest increment. Also, all the instances except $(146 - 20 - 2)$ drive on the same path. PEV changes its optimal path on instance $(146 - 20 - 2)$. For other instances, PEV drives at a larger speed on each road segment. Moreover, Table 5.32 shows the solutions given by matheuristic with MIP Initialization under tight upper time limit. All the solutions are exact as shown in the last column of the Table. Compared to exact solution times, matheuristic

Table 5.31: Exact Solutions of some Networks with Tight Upper Time Limit T_{new}

|N|	**Density** %	**Pattern**	α	**Path**	**Energy Consumption Value** ($\times 10^8$)	**Solution Time** (sec)	**Energy Consumption Increase** (%)
24	30	1	0.70	1-19-22-23-24	6.631	2.621	3.30
24	30	2	0.90	1-20-21-24	5.926	2.820	1.80
41	30	1	0.88	1-2-3-9-10-11-17-20-27-32-36-40-41	1.976	140.311	1.80
41	30	2	0.85	1-2-3-9-10-11-17-20-27-32-36-40-41	2.063	30.333	6.29
41	30	3	0.80	1-2-3-9-10-11-17-20-27-32-36-40-41	2.034	15.763	4.79
50	20	1	0.80	1-15-23-28-27-35-36-38-46-45-50	1.052	32.941	4.89
50	20	2	0.90	1-15-24-29-37-44-45-50	9.479	25.199	3.68
50	20	3	0.45	1-15-24-29-37-44-45-50	9.261	19.965	1.29
146	20	2	0.90	1-79-80-81-95-122-123-124-144-145-146	1.180	4274.790	0.68
195	40	1	0.90	1-17-31-32-33-34-35-51-59 -60-76-77-89-90-91-92-105-123 -124-141-159-160-161-177-194-195	2.529	15994.601	0.52

with MIP Initialization heuristic gives solutions faster. Likewise, Table 5.33 shows the solutions given by matheuristic with k^{th} Path Initialization under tight upper time limit. This algorithm also finds all the solutions with 0 gap. It dominates the previous heuristic for larger instances whose node sizes are 146 and 195. Matheuristic with MIP Initialization does not solve the instance $(195 - 40 - 1)$. On the other hand, this heuristic solves all instances in a faster way. VNS solutions can be seen in Table 5.34. There are 3 instances, $(24 - 30 - 1)$, $(50 - 20 - 2)$ and $(146 - 20 - 2)$, not solved by VNS. Also, instance $(195 - 40 - 1)$ is solved with optimality gap 4.03%. It can be said that VNS solves these instances under time limitation faster but it gives significant optimality gaps.

Table 5.32: Matheuristic with MIP Initialization Solutions of some Networks with Tight Upper Time Limit T_{new}

|N|	**Density** $\%$	**Pattern**	α	**Path**	**Energy Consumption Value** $(\times 10^8)$	**Solution Time** (sec)	**Optimality Gap** $(\%)$
24	30	1	0.70	1-19-22-23-24	6.631	10.51	0
24	30	2	0.90	1-20-21-24	5.926	11.047	0
41	30	1	0.88	1-2-3-9-10-11-17-20-27-32-36-40-41	1.976	27.013	0
41	30	2	0.85	1-2-3-9-10-11-17-20-27-32-36-40-41	2.063	25.708	0
41	30	3	0.80	1-2-3-9-10-11-17-20-27-32-36-40-41	2.034	29.968	0
50	20	1	0.80	1-15-23-28-27-35-36-38-46-45-50	1.052	35.587	0
50	20	2	0.90	1-15-24-29-37-44-45-50	9.479	38.862	0
50	20	3	0.45	1-15-24-29-37-44-45-50	9.261	33.952	0
146	20	2	0.90	1-79-80-81-95-122-123-124-144-145-146	1.180	382.372	0
195	40	1	0.90	-	-	-	-

Table 5.33: Matheuristic with k^{th} Path Initialization Solutions of some Networks with Tight Upper Time Limit T_{new}

|N|	**Density** $\%$	**Pattern**	α	**Path**	**Energy Consumption Value** $(\times 10^8)$	**Solution Time** (sec)	**Optimality Gap** $(\%)$
24	30	1	0.70	1-19-22-23-24	6.631	22.147	0
24	30	2	0.90	1-20-21-24	5.926	5.537	0
41	30	1	0.88	1-2-3-9-10-11-17-20-27-32-36-40-41	1.976	30.181	0
41	30	2	0.85	1-2-3-9-10-11-17-20-27-32-36-40-41	2.063	21.631	0
41	30	3	0.80	1-2-3-9-10-11-17-20-27-32-36-40-41	2.034	11.462	0
50	20	1	0.80	1-15-23-28-27-35-36-38-46-45-50	1.052	28.724	0
50	20	2	0.90	1-15-24-29-37-44-45-50	9.479	38.591	0
50	20	3	0.45	1-15-24-29-37-44-45-50	9.261	40.195	0
146	20	2	0.90	1-79-80-81-95-122-123 -124-144-145-146	1.180	191.576	0
195	40	1	0.90	1-17-31-32-33-34-35 -51-59-60-76-77-89-90-91 -92-105-123-124-141-159 -160-161-177-194-195	2.529	192.736	0

Table 5.34: VNS Heuristic Solutions of some Networks with Tight Upper Time Limit T_{new}

|N|	**Density** %	**Pattern**	α	**Path**	**Energy Consumption Value** $(\times 10^8)$	**Solution Time** (sec)	**Optimality Gap** $(\%)$
24	30	1	0.70	-	-	-	100
24	30	2	0.90	1-20-21-24	5.926	8.916	0
41	30	1	0.88	1-2-3-9-10-11-17-20-27-32-36-40-41	1.976	21.525	0
41	30	2	0.85	1-2-3-9-10-11-17-20-27-32-36-40-41	2.063	20.847	0
41	30	3	0.80	1-2-3-9-10-11-17-20-27-32-36-40-41	2.034	23.172	0
50	20	1	0.80	1-15-23-28-27-35-36-38-46-45-50	1.052	26.116	0
50	20	2	0.90	-	-	-	100
50	20	3	0.45	1-15-24-29-37-44-45-50	9.261	26.743	0
146	20	2	0.90	-	-	-	100
195	40	1	0.90	1-2-3-4-19-20-21-37-38-39-40-63-64-92-105-123-124-141-159-160-161-177-194-195	2.631	268.13	4.03

CHAPTER 6

CONCLUSIONS

Electric vehicles are important part of today's technology and energy consumption field. Their usage rates increase day by day because of the economic and environmental benefits. Hence, both theoretically challenging and practically important problems arise. In the direction of this trend, a problem about energy optimization of a Plug-In Electric Vehicle (PEV) over a network is analyzed in this thesis. Selection of stops and velocity to reach minimum level of energy consumption are decided. In the literature, all of the studies trying to find an energy efficient path use velocity as a parameter. Here, it is considered as a decision variable and our objective is a third degree function of the speed of the vehicle. After a literature review, energy consumption function of the PEV is taken as objective in our problem. This function represents the behaviour of the battery on the PEV based on some parameters, mainly V.

An MISOCP model is used in order to solve the problem. It is seen that the formulation becomes inadequate for larger instances such as the graph with 321 nodes. Hence, a heuristic approach is applied on the problem. Matheuristic proposed is composed of 3 steps. Each step includes a different mathematical formulation. By the help of these optimization problems, exploitation can be satisfied while finding a solution. Moreover, improvement heuristics are applied between these steps. Fine tuning of the parameters is an important part in order to reach different solutions and exact solution at the end in the solution area. The proposed heuristic gives exact solutions in shorter times for the problems whose solutions were found by the MISOCP formulation. Also, larger instances are solved in reasonable time. Likewise, a VNS based approach is proposed to determine an efficient path for a PEV. Speed optimization part of the problem is realized by an MISOCP at final step of the VNS algorithm.

Again, it is seen that VNS algorithm is good at finding optimal or solutions near to optimal in shorter time intervals compared to exact algorithm over large instances. To conclude, computation times and solution quality of the proposed heuristic are acceptable for the problems with previously decided optimal solutions.

As a future work, different instances can be solved using the heuristic methods on hand. Moreover, T- Improvement, E-Improvement and VNS algorithms can be developed in terms of their computational efforts. New neighborhood definitions can be made on speed optimization for a PEV. Moreover, a time dependent problem definition can be made considering the traffic congestion on road segments. In this case, a dynamic programming method can be used to solve the problem. For further applications of PEVs, our problem definition can be modified in order to solve minimum cost path problems for commercial vehicles, which is a valuable research direction.

REFERENCES

[1] Y. Li, L. Zhang, H. Zheng, X. He, S. Peeta, T. Zheng, and Y. Li, "Nonlane-discipline-based car-following model for electric vehicles in transportation-cyber-physical systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 38–47, 2017.

[2] I. IEA, W. UNSD, *et al.*, "Who. tracking sdg 7: The energy progress report 2019," *Washington DC*, 2019.

[3] E. Regulation, "Regulation (eu) 2018/1999 of the european parliament and of the council of 11 december 2018 on the governance of the energy union and climate action, amending regulations (ec) no 663/2009 and (ec) no 715/2009 of the european parliament and of the council," tech. rep., Directives 94/22/EC, 98/70/EC, 2009/31/EC, 2009/73/EC, 2010/31/EU, 2012/27 . . . , 2018.

[4] O. Arslan, B. Yıldız, and O. E. Karaşan, "Minimum cost path problem for plug-in hybrid electric vehicles," *Transportation Research Part E: Logistics and Transportation Review*, vol. 80, pp. 123–141, 2015.

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

[6] M. Baum, J. Dibbelt, L. Hübschle-Schneider, T. Pajor, and D. Wagner, "Speed-consumption tradeoff for electric vehicle route planning," in *14Th workshop on algorithmic approaches for transportation modelling, optimization, and systems*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[7] M. Strehler, S. Merting, and C. Schwan, "Energy-efficient shortest routes for electric and hybrid vehicles," *Transportation Research Part B: Methodological*, vol. 103, pp. 111–135, 2017.

[8] J. Lu, Y. Chen, J.-K. Hao, and R. He, "The time-dependent electric vehicle routing problem: Model and solution," *Expert Systems with Applications*, p. 113593, 2020.

[9] P. Vasant, J. A. Marmolejo, I. Litvinchev, and R. R. Aguilar, "Nature-inspired meta-heuristics approaches for charging plug-in hybrid electric vehicle," *Wireless Networks*, vol. 26, no. 7, pp. 4753–4766, 2020.

[10] C. De Cauwer, J. Van Mierlo, and T. Coosemans, "Energy consumption prediction for electric vehicles based on real-world data," *Energies*, vol. 8, no. 8, pp. 8573–8593, 2015.

[11] X. Wu, X. He, G. Yu, A. Harmandayan, and Y. Wang, "Energy-optimal speed control for electric vehicles on signalized arterials," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2786–2796, 2015.

[12] X. Wu, D. Freese, A. Cabrera, and W. A. Kitch, "Electric vehicles' energy consumption measurement and estimation," *Transportation Research Part D: Transport and Environment*, vol. 34, pp. 52–67, 2015.

[13] C. Archetti and M. G. Speranza, "A survey on matheuristics for routing problems," *EURO Journal on Computational Optimization*, vol. 2, no. 4, pp. 223–246, 2014.

[14] J. G. Villegas, C. Prins, C. Prodhon, A. L. Medaglia, and N. Velasco, "A matheuristic for the truck and trailer routing problem," *European Journal of Operational Research*, vol. 230, no. 2, pp. 231–244, 2013.

[15] R. Kramer, A. Subramanian, T. Vidal, and F. C. Lucídio dos Anjos, "A matheuristic approach for the pollution-routing problem," *European Journal of Operational Research*, vol. 243, no. 2, pp. 523–539, 2015.

[16] A. Franceschetti, D. Honhon, T. Van Woensel, T. Bektaş, and G. Laporte, "The time-dependent pollution-routing problem," *Transportation Research Part B: Methodological*, vol. 56, pp. 265–293, 2013.

[17] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, "The electric vehicle routing problem with nonlinear charging function," *Transportation Research Part B: Methodological*, vol. 103, pp. 87–110, 2017.

[18] M. Bruglieri, F. Pezzella, O. Pisacane, and S. Suraci, "A matheuristic for the electric vehicle routing problem with time windows," *arXiv preprint arXiv:1506.00211*, 2015.

[19] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte, *A matheuristic for the electric vehicle routing problem with capacitated charging stations*. PhD thesis, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la . . . , 2017.

[20] R. Abousleiman and O. Rawashdeh, "Energy-efficient routing for electric vehicles using metaheuristic optimization frameworks," in *MELECON 2014-2014 17th IEEE Mediterranean Electrotechnical Conference*, pp. 298–304, IEEE, 2014.

[21] R. Abousleiman, O. Rawashdeh, and R. Boimer, "Electric vehicles energy efficient routing using ant colony optimization," *SAE International Journal of Alternative Powertrains*, vol. 6, no. 1, pp. 1–14, 2017.

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

[23] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.

[24] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

[25] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Röhler, "Matheuristics: Optimization, simulation and control," in *International Workshop on Hybrid Metaheuristics*, pp. 171–177, Springer, 2009.

[26] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[27] P. Hansen, N. Mladenović, R. Todosijević, and S. Hanafi, "Variable neighborhood search: basics and variants," *EURO Journal on Computational Optimization*, vol. 5, no. 3, pp. 423–454, 2017.

[28] E. W. Dijkstra *et al.*, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[29] J.-M. Belenguer, E. Benavent, P. Lacomme, and C. Prins, “Lower and upper bounds for the mixed capacitated arc routing problem,” *Computers & Operations Research*, vol. 33, no. 12, pp. 3363–3383, 2006.

[30] L. Gouveia, M. C. Mourão, and L. S. Pinto, “Lower bounds for the mixed capacitated arc routing problem,” *Computers & Operations Research*, vol. 37, no. 4, pp. 692–699, 2010.

APPENDIX A

PEVEEP-MISOCP MODEL

(PEVEEP-MISOCP)

$$\text{Minimize} \sum_{i\in\mathcal{N}}\sum_{i\in\mathcal{N}} E_{ij} \tag{A.1}$$

subject to

$$E_{ij} \geq e \times t_{ij} + D_{ij} \times (b \times l_{ij} + a \times f_{ij} + c \times Vij + x_{ij} \times d) \quad \forall (i,j) \in \mathcal{A} \tag{A.2}$$

$$\left\| \begin{matrix} V_{ij} \\ (l_{ij}-1)/2 \end{matrix} \right\| \leq (l_{ij}+1)/2 \quad \forall (i,j) \in \mathcal{A} \tag{A.3}$$

$$\left\| \begin{matrix} l_{ij} \\ (f_{ij}-V_{ij})/2 \end{matrix} \right\| \leq (f_{ij}+V_{ij})/2 \quad \forall (i,j) \in \mathcal{A} \tag{A.4}$$

$$\left\| \begin{matrix} x_{ij} \times \sqrt{D_{ij}} \\ (t_{ij}-V_{ij})/2 \end{matrix} \right\| \leq (t_{ij}+V_{ij})/2 \quad \forall (i,j) \in \mathcal{A} \tag{A.5}$$

$$\sum_{i\in N} x_{1i} - \sum_{i\in N} x_{i1} = 1 \tag{A.6}$$

$$\sum_{i\in N} x_{Ni} - \sum_{i\in N} x_{iN} = -1 \tag{A.7}$$

$$\sum_{i=2}^{N-1} x_{ij} - \sum_{i=2}^{N-1} x_{ji} = 0 \qquad j = 2,\ldots,N-1 \tag{A.8}$$

$$M \times (1 - x_{ij}) \geq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{A.9}$$

$$M \times (x_{ij} - 1) \leq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{A.10}$$

$$V_{ij} \leq U_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{A.11}$$

$$V_{ij} \geq L_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{A.12}$$

$$ea_1 = I \tag{A.13}$$

$$ec_N = 0 \tag{A.14}$$

$$ct_i = A \times y_i + B \times ec_i \qquad \forall i \in \mathcal{S} \tag{A.15}$$

$$ed_i = ea_i + ec_i \qquad \forall i \in \mathcal{N} \tag{A.16}$$

$$ed_i \leq C \qquad \forall i \in \mathcal{N} \tag{A.17}$$

$$ec_i \leq y_i \times (C - m) \qquad \forall i \in \mathcal{S} \tag{A.18}$$

$$y_j \leq \sum_{i=1}^{N-1} x_{ij} \qquad j = 2, \ldots, N-1 \tag{A.19}$$

$$ea_j \leq \sum_{i=1}^{N-1} x_{ij} \times C \qquad j = 2, \ldots, N-1 \tag{A.20}$$

$$ea_j \geq \sum_{i=1}^{N-1} x_{ij} \times m \qquad j = 2, \ldots, N \tag{A.21}$$

$$T \geq \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} + \sum_{i=1}^{N} ct_j \tag{A.22}$$

$$\sum_{i=1}^{N} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{A.23}$$

$$V_{ij}, t_{ij}, E_{ij}, l_{ij}, f_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{A.24}$$

$$ec_i, ed_i, ea_i, ct_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{A.25}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in \mathcal{A} \tag{A.26}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{A.27}$$

APPENDIX B

PEVTEPP-MIP MODEL

(PEVTEPP-MIP)

Minimize $\sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij} + \sum_{i=1}^{N} ct_j$ (B.1)

subject to

$$E_{ij} \geq e \times t_{ij} + D_{ij} \times (a \times V_{ij}^3 + b \times V_{ij}^2 + c \times V_{ij} + x_{ij} \times d) \quad \forall (i,j) \in \mathcal{A} \tag{B.2}$$

$$\sum_{i \in N} x_{1i} - \sum_{i \in N} x_{i1} = 1 \tag{B.3}$$

$$\sum_{i \in N} x_{Ni} - \sum_{i \in N} x_{iN} = -1 \tag{B.4}$$

$$\sum_{i=2}^{N-1} x_{ij} - \sum_{i=2}^{N-1} x_{ji} = 0 \qquad j = 2, \ldots, N-1 \tag{B.5}$$

$$M \times (1 - x_{ij}) \geq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{B.6}$$

$$M \times (x_{ij} - 1) \leq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{B.7}$$

$$V_{ij} \leq U_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{B.8}$$

$$V_{ij} \geq L_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{B.9}$$

$$x_{ij} \times D_{ij} \geq L_{ij} \times t_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{B.10}$$

$$ea_1 = I \tag{B.11}$$

$$ec_N = 0 \tag{B.12}$$

$$ct_i = A \times y_i + B \times ec_i \qquad \forall i \in \mathcal{S} \tag{B.13}$$

$$ed_i = ea_i + ec_i \qquad \forall i \in \mathcal{N} \tag{B.14}$$

$$ed_i \leq C \qquad \forall i \in \mathcal{N} \tag{B.15}$$

$$ec_i \leq y_i \times (C - m) \qquad \forall i \in \mathcal{S} \tag{B.16}$$

$$y_j \leq \sum_{i=1}^{N-1} x_{ij} \qquad j = 2, \ldots, N-1 \tag{B.17}$$

$$ea_j \leq \sum_{i=1}^{N-1} x_{ij} \times C \qquad j = 2, \ldots, N-1 \tag{B.18}$$

$$ea_j \geq \sum_{i=1}^{N-1} x_{ij} \times m \qquad j = 2, \ldots, N \tag{B.19}$$

$$\sum_{i=1}^{N} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{B.20}$$

$$t_{ij}, E_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{B.21}$$

$$ec_i, ed_i, ea_i, ct_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{B.22}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in \mathcal{A} \tag{B.23}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{B.24}$$

APPENDIX C

PEVTEPP-MISOCP MODEL

(PEVTEPP-MISOCP)

$$\text{Minimize } \sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij} + \sum_{i=1}^{N} ct_j \tag{C.1}$$

subject to

$$E_{ij} \geq e \times t_{ij} + D_{ij} \times (b \times l_{ij} + a \times f_{ij} + c \times Vij + x_{ij} \times d) \quad \forall (i,j) \in \mathcal{A} \tag{C.2}$$

$$\left\| \begin{matrix} V_{ij} \\ (l_{ij}-1)/2 \end{matrix} \right\| \leq (l_{ij}+1)/2 \quad \forall (i,j) \in \mathcal{A} \tag{C.3}$$

$$\left\| \begin{matrix} l_{ij} \\ (f_{ij}-V_{ij})/2 \end{matrix} \right\| \leq (f_{ij}+V_{ij})/2 \quad \forall (i,j) \in \mathcal{A} \tag{C.4}$$

$$\left\| \begin{matrix} x_{ij} \times \sqrt{D_{ij}} \\ (t_{ij}-V_{ij})/2 \end{matrix} \right\| \leq (t_{ij}+V_{ij})/2 \quad \forall (i,j) \in \mathcal{A} \tag{C.5}$$

$$M \times (1 - x_{ij}) \geq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{C.6}$$

$$M \times (x_{ij} - 1) \leq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{C.7}$$

$$V_{ij} \leq U_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{C.8}$$

$$V_{ij} \geq L_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{C.9}$$

$$ea_1 = I \tag{C.10}$$

$$ec_N = 0 \tag{C.11}$$

$$ct_i = A \times y_i + B \times ec_i \qquad \forall i \in \mathcal{S} \tag{C.12}$$

$$ed_i = ea_i + ec_i \qquad \forall i \in \mathcal{N} \tag{C.13}$$

$$ed_i \leq C \qquad \forall i \in \mathcal{N} \tag{C.14}$$

$$ec_i \leq y_i \times (C - m) \qquad \forall i \in \mathcal{S} \tag{C.15}$$

$$y_j \leq \sum_{i=1}^{N-1} x_{ij} \qquad j = 2, \ldots, N-1 \tag{C.16}$$

$$ea_j \leq \sum_{i=1}^{N-1} x_{ij} \times C \qquad j = 2, \ldots, N-1 \tag{C.17}$$

$$ea_j \geq \sum_{i=1}^{N-1} x_{ij} \times m \qquad j = 2, \ldots, N \tag{C.18}$$

$$T \geq \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} + \sum_{i=1}^{N} ct_j \tag{C.19}$$

$$V_{ij}, t_{ij}, E_{ij}, l_{ij}, f_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{C.20}$$

$$ec_i, ed_i, ea_i, ct_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{C.21}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{C.22}$$

APPENDIX D

PEVFEEP–MISOCP MODEL

(PEVFEEP-MISOCP)

$$\text{Minimize } \sum_{i\in N}\sum_{i\in N} E_{ij} \tag{D.1}$$

subject to

$$E_{ij} \geq e \times t_{ij} + D_{ij} \times (b \times l_{ij} + a \times f_{ij} + c \times Vij + x_{ij} \times d) \quad \forall (i,j) \in \mathcal{A} \tag{D.2}$$

$$\left\| \begin{matrix} V_{ij} \\ (l_{ij}-1)/2 \end{matrix} \right\| \leq (l_{ij}+1)/2 \quad \forall (i,j) \in \mathcal{A} \tag{D.3}$$

$$\left\| \begin{matrix} l_{ij} \\ (f_{ij}-V_{ij})/2 \end{matrix} \right\| \leq (f_{ij}+V_{ij})/2 \quad \forall (i,j) \in \mathcal{A} \tag{D.4}$$

$$\left\| \begin{matrix} x_{ij} \times \sqrt{D_{ij}} \\ (t_{ij}-V_{ij})/2 \end{matrix} \right\| \leq (t_{ij}+V_{ij})/2 \quad \forall (i,j) \in \mathcal{A} \tag{D.5}$$

$$M \times (1 - x_{ij}) \geq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{D.6}$$

$$M \times (x_{ij} - 1) \leq ea_j - ed_i + E_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{D.7}$$

$$V_{ij} \leq U_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{D.8}$$

$$V_{ij} \geq L_{ij} \times x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{D.9}$$

$$ea_1 = I \tag{D.10}$$

$$ec_N = 0 \tag{D.11}$$

$$ct_i = A \times y_i + B \times ec_i \qquad \forall i \in \mathcal{S} \tag{D.12}$$

$$ed_i = ea_i + ec_i \qquad \forall i \in \mathcal{N} \tag{D.13}$$

$$ed_i \leq C \qquad \forall i \in \mathcal{N} \tag{D.14}$$

$$ec_i \leq y_i \times (C - m) \qquad \forall i \in \mathcal{S} \tag{D.15}$$

$$y_j \leq \sum_{i=1}^{N-1} x_{ij} \qquad j = 2, \ldots, N-1 \tag{D.16}$$

$$ea_j \leq \sum_{i=1}^{N-1} x_{ij} \times C \qquad j = 2, \ldots, N-1 \tag{D.17}$$

$$ea_j \geq \sum_{i=1}^{N-1} x_{ij} \times m \qquad j = 2, \ldots, N \tag{D.18}$$

$$T \geq \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} + \sum_{i=1}^{N} ct_j \tag{D.19}$$

$$V_{ij}, t_{ij}, E_{ij}, l_{ij}, f_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{D.20}$$

$$ec_i, ed_i, ea_i, ct_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{D.21}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{D.22}$$